UNIVERSIDADE FEDERAL FLUMINENSE

ROGÉRIO DA SILVA BATISTA

Implementações eficientes para problemas de caminhos e ciclos em grafos com arestas coloridas

NITERÓI

2014

UNIVERSIDADE FEDERAL FLUMINENSE

ROGÉRIO DA SILVA BATISTA

Implementações eficientes para problemas de caminhos e ciclos em grafos com arestas coloridas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização Combinatória

Orientador:
CARLOS ALBERTO DE JESUS MARTINHON

NITERÓI

2014

ROGÉRIO DA SILVA BATISTA

Implementações eficientes para problemas de caminhos e ciclos em grafos com arestas coloridas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização Combinatória

Aprovada em de .

BANCA EXAMINADORA

Prof. Carlos Alberto de Jesus Martinhon - Orientador, UFF

Prof. Luiz Satoru Ochi, UFF

Prof. Luérbio Faria, UERJ

Niterói

<2014>

Agradecimentos

Primeiramente, agradeço a Deus por ter me permitido alcançar este objetivo.

À minha amável esposa, Marta Lira, por todo apoio e compreensão, principalmente nos momentos em que estive distante. Sou imensamente grato por tudo de bom que você tem proporcionado na minha vida.

Aos meus pais Iberê e Elzimar pelo apoio e compreensão.

Ao meu orientador Carlos Martinhon por me aceitar como seu orientando, por ter acreditado neste trabalho, pelos ensinamentos, pela paciência e pela confiança que sempre me passou durante toda orientação. Muito Obrigado!

Aos colegas de MINTER/DINTER do IFPI, Ely, Rogério Silva, Cláutenis, Thiago, Franciéric, Adalton, Valéria, Elanne, Ney, Jefferson e Ritomar pela força, confiança e estímulo para seguir em frente nos momentos mais difíceis.

Aos colegas da UFF com quem convivi neste mestrado, em especial ao Eyder Rios, pela atenção, disposição, conselhos, e ensinamentos valiosos de programação em C/C++ que obtive. Muito Obrigado!

Aos professores da UFF, especialmente àqueles que fizeram parte do convênio MINTER/DINTER com o IFPI.

Aos professores da banca por aceitarem o convite e participarem deste momento tão especial em minha carreira.

Às secretárias da pós, Teresa e Viviane, pela atenção e apoio.

Aos meus chefes imediatos na Agespisa: Pádua e Valdiná e ao Diretor Administrativo Sr. Nonato Marreiros por proporcionarem minha liberação para conclusão deste Mestrado.

Ao Reitores do IFPI, prof. Santana (anterior) e prof. Paulo Henrique (atual), ao Pró-Reitores de Pesquisa, profa. Valdira (anterior) e prof. Brandim (atual), aos chefes de departamento, prof. Marcos Teixeira (anterior) e prof. Jurandir (atual) e ao coordenador

Eduilson pelo apoio no meu afastamento para conclusão deste Mestrado.

Resumo

Nos últimos anos uma grande quantidade de problemas importantes tem sido modelada por grafos com cores nas arestas. Neste trabalho, consideramos diferentes questões sobre $s-t$ caminhos e trilhas propriamente coloridos. Dizemos que um $s-t$ caminho/trilha é propriamente colorido quando suas arestas sucessivas possuírem cores distintas. Foram demonstrados na literatura que alguns destes problemas tem solução polinomial e se baseiam principalmente na redução de um problema de caminho/trilha propriamente colorido em um grafo G^c com c cores nas arestas em um problema de emparelhamento perfeito em um grafo G não-colorido.

Implementamos algoritmos para determinação do menor e maior caminho/trilha propriamente colorido em um grafo G^c. Na construção do algoritmo polinomial para a determinação do maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c sem ciclos propriamente coloridos foram implementados algoritmos para determinação de vértices de corte separando cores, vértices monocromáticos e pontes em G^c. Quando não há restrição de ciclos/trilhas fechadas propriamente coloridas para determinação do maior $s-t$ caminho/trilha em um grafo G^c qualquer, este problema torna-se NP-Difícil.

No caso geral, para um grafo G^c qualquer apresentamos um modelo de Programação Linear Inteira (PLI) para o problema do maior $s-t$ caminho propriamente colorido. Implementamos um algoritmo que detecta cada sub-ciclo propriamente colorido e elimina-o através da inserção de uma nova restrição ao modelo de PLI construído. Além disso, foi implementado um algoritmo para determinação do menor custo de recoloração de arestas para determinação de $s-t$ caminhos propriamente coloridos, onde fizemos uma adaptação da construção XP-Gadget (proposto na literatura), que nos permitiu uma redução na quantidade de vértices gerados no grafo resultante não-colorido em comparação com a construção original.

Resultados computacionais mostraram, por exemplo, que o algoritmo para obtenção do menor $s-t$ caminho propriamente colorido em um grafo G^c obteve melhores resultados em termos de tempo computacional quando comparado com um algoritmo baseado em um modelo de PLI sem a restrição de eliminação de sub-ciclos.

Palavras-chave: Grafos com Cores nas Arestas; Caminhos e Trilhas Propriamente Coloridos; Emparelhamento Perfeito em Grafos; Custo de recoloração de arestas; Programação Linear Inteira;

Abstract

In the last few years, a great number of important problems have been modeled by edge-colored graphs. In this work, we consider different issues about properly edge-colored $s-t$ paths/trails. We define properly edge-colored $s-t$ paths/trails when any two sucessive edges in these paths/trails differ in color. Some of these problems were shown in the literature to be polynomially solvable and the solutions are based on reducing the properly edge-colored path problem in G^c to a perfect matching problem in a non-colored graph defined appropriately.

We implement algorithms for finding the shortest/longest properly edge-colored path/trail in a graph G^c. In the implementation of the polynomial algorithm for finding a longest $s-t$ properly edge-colored path/trail in a graph G^c with no properly edge-colored cycles we give algorithms for finding cut vertex separating colors, monochromatic vertices and bridges in G^c. When there is no properly edge-colored cycles (closed trails), the problem of finding a longest $s-t$ path/trail in an arbitrary graph G^c becomes NP-Hard.

In the general case, for an arbitrary graph G^c, we present an Integer Linear Programming (ILP) formulation to the longest $s-t$ properly edge-colored path problem. We implemented an algorithm that detects each properly edge-colored sub-cycle and eliminates it by inserting a new constraint to the ILP model. In addition, an algorithm was implemented for finding a shortest edge-recoloring cost to determine properly edge-colored $s-t$ paths, in which we adapt the XP-Gadget building (proposed in literature), which allowed us to reduce the number of vertices in a non-colored generated graph when compared with the original gadget.

Computational results showed, for example, that the algorithm for the shortest $s-t$ properly edge-colored path outperforms the ILP approach in terms of computational time.

Keywords: Edge-Colored Graphs; Properly Edge-Colored Paths/Trails; Perfect Matching in Graphs; Edge-recoloring cost; Integer Linear Programming;

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

SGB	:	Stanford Graph Base;
PLI	:	Programação Linear Inteira;
3DNMR	:	Reconstrução de Caminhos Ótimos em Mapas de Ressonância Magnética Nuclear Tridimensional;
		Programação Linear Inteira;

Sumário

Capítulo 1

Introdução

Nos últimos anos uma grande quantidade de problemas importantes tem sido modelada por grafos com cores nas arestas. Para resolvê-los, conexões interessantes tem sido exploradas entre grafos com arestas coloridas, teoria de caminhos e ciclos em grafos direcionados e não-direcionados, emparelhamento em grafos, fluxo em redes além de outros ramos da teoria de grafos [1, 4, 17, 20, 27]. Para exemplificar algumas destas aplicações podemos citar problemas em biologia molecular [31], em particular no problema da reconstrução de caminhos ótimos em mapas de Ressonância Magnética Nuclear Tridimensional (3D NMR) [33], onde, por exemplo, pretende-se encontrar o maior caminho formado por um conjunto de arestas obedecendo uma determinada ordem de cores. Problemas de criptografia/fluxo em redes [36, 37], ciências sociais [7], conectividade e transporte [15], em particular, nos problemas onde custos de conexão são associados a pares de cores em arestas adjacentes cujo o objetivo é encontrar uma rota onde o custo total de conexão seja minimizado [14, 17, 38] e problemas com custos de recoloração de arestas, cujo o objetivo é obter o menor custo com a troca de cores nas arestas para que uma determinada propriedade seja estabelecida [29].

Várias propriedades interessantes podem ser consideradas nos problemas modelados por grafos com arestas coloridas. Entre elas, podemos citar a determinação de caminhos, trilhas e ciclos propriamente coloridos em grafos e dígrafos. Dizemos que um caminho, trilha ou ciclo é propriamente colorido se arestas sucessivas possuírem cores distintas.

O estudo das propriedades referentes a $s-t$ caminhos e trilhas propriamente coloridos, abordados em diversos trabalhos presentes na literatura, nos motivaram a transformar alguns resultados importantes obtidos nestes trabalhos em um conjunto de algoritmos que integrados a outros *solvers* em um mesmo ambiente, produzissem resultados práticos equivalentes quando aplicados a instancias de problemas representados em um grafo com

cores nas arestas.

A seguir, serão apresentadas as notações e definições de grafos que são utilizadas neste trabalho. Mais especificamente, abordaremos conceitos importantes relacionados a contrações de grafos não-direcionados, sub-grafos induzidos, grafos conexos, caminhos e ciclos em grafos, vizinhança e grau de cores de vértices.

Outro conceito importante abordado neste capítulo é o de Emparelhamento em grafos. Abordamos em especial o problema de emparelhamento perfeito em grafos para o caso geral (surgimento de florações), resolvido inicialmente pelo algoritmo de Edmonds (Blossom I) e tendo como mais recente implementação o Blossom V proposto em [25].

Descrevemos também, uma seção com a representação computacional utilizada em nossa implementação. Detalhamos a estrutura do Stanford Graph Base (SGB) [24] que foi a plataforma baseada em vetor de listas de adjacências que foi utilizada neste trabalho.

1.1 Grafos: Definições e Notação

Um *grafo simples* G é um par ordenado $G = (V, E)$, onde V ou $V(G)$ representa um conjunto finito não-vazio de *vértices* e E ou $E(G)$ representa um conjunto de pares não-ordenados de vértices distintos, denominados *arestas*. Seja $e \in E(G)$, portanto $e = (u, v)$ ou $e = uv$. Neste caso, dizemos que u e v são adjacentes ou vizinhos em G e que a aresta $e = (u, v)$ é incidente a u e a v ou que tem extremos em u, v. Denotamos por $n = |V(G)|$ e $m = |E(G)|$ o número de vértices e de arestas de um Grafo G, respectivamente. No restante deste trabalho, utilizaremos a denominação *grafos* para denotar o que definimos como *grafos simples*.

Denotamos por $N_G(v)$ o conjunto dos vértices que são adjacentes ao vértice v e este conjunto é chamado de *vizinhança* de v.

Um *grafo completo* G é um grafo tal que se $u, v \in V(G)$ onde $u \neq v$, então u e v são adjacentes. Denotamos por K_n um *grafo completo* com n vértices.

O complemento $\overline{G}$ de um grafo G é um grafo com o mesmo conjunto de vértices $V(G)$ e com pares de arestas entre os vértices não-adjacentes em G.

Vamos definir o conceito de *contração* em um grafo não-direcionado. Dado um subgrafo induzido X de um grafo G não-colorido, a contração de X em G consiste em substituir X por um novo vértice denotado por z_X em que cada vértice v em $G - X$ é conectado a z_X por uma aresta, se e somente se, existir uma aresta $uv \in G$ para cada vértice $v \in X$.

A Figura 1.1 ilustra um exemplo de uma contração em um grafo.

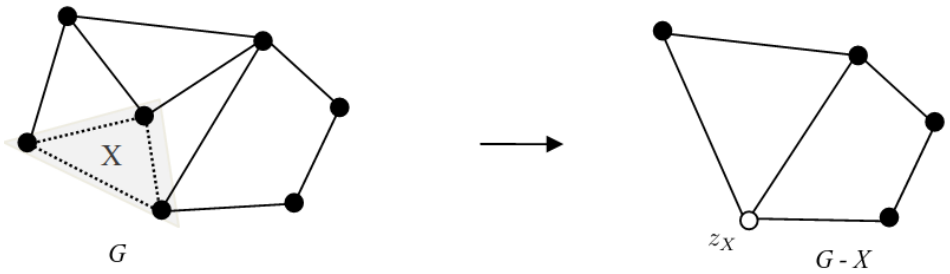

Figura 1.1: Um grafo G e o resultado de sua contração em G - X através do vértice z_X

O grau de um vértice $v \in V(G)$, denotado por $d_G(v)$, é o numero de arestas incidentes ao vértice v denotado por $|E(v)|$. É equivalente a dizer que é igual ao número de vértices adjacentes a v denotado por $N_G(v)$. O grau mínimo de G é denotado por $\delta(G) = min\{d_G(v)|v \in V\}$ e o grau máximo de G é denotado por $\Delta(G) = max\{d_G(v)|v \in V(G)\}$.

Um grafo H é um subgrafo de um grafo G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Seja V' um subconjunto não vazio de V. Um subgrafo de G induzido por V' (denotado por $G[V']$) é um subgrafo H de G tal que $V(H) = V'$ e $E(H)$ é o conjunto das arestas de G que tem ambos os extremos em V'. O subgrafo induzido $G[V \setminus V']$, denotado por $G - V'$, é o subgrafo obtido de G pela remoção dos vértices em V' e todas as suas arestas incidentes. Se $V' = \{v\}$ escrevemos $G - v$ ao invés de $G - \{v\}$.

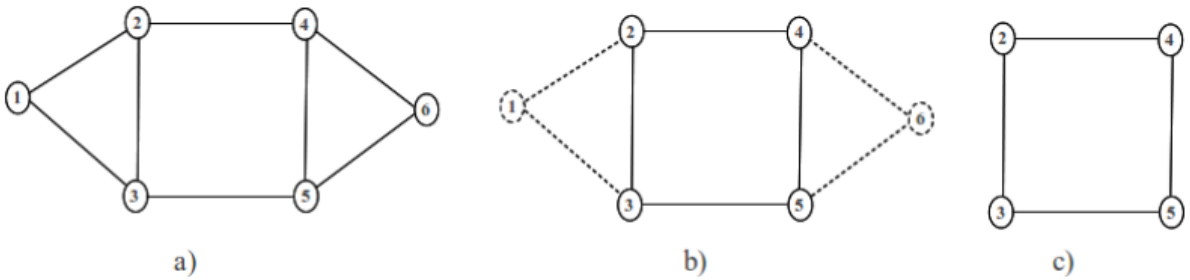

Figura 1.2: a)Um grafo original G; b) Um subconjunto não-vazio $V' = \{1, 6\}$; c) Subgrafo resultante $G - V'$

Seja dois vértices $s, t \in V(G)$. Um *passeio* entre s e t no grafo é uma sequencia alternada finita $\rho = (v_0, e_0, v_1, e_1, v_2, e_2, ..., e_{n-1}, v_n)$ de vértices e arestas do grafo G, onde: $V(\rho) = (v_0, v_1, \ldots, v_n)$, $E(\rho) = (e_0, e_1, \ldots, e_{n-1})$, $v_0 = s$, $v_n = t$ e $e_i = (v_i, v_{i+1})$ para todo $i < n$, onde os vértices e arestas podem eventualmente se repetir. Uma *trilha* é um

passeio onde as arestas são todas distintas e um *caminho* é uma trilha onde os vértices são todos distintos. Os vértices s e t são as extremidades e os demais vértices são os vértices intermediários do caminho. O tamanho de um caminho, trilha ou passeio P é a quantidade de arestas presentes, ou seja, o número de arestas entre o vértice inicial e o vértice final de P. Se ρ é um caminho(resp. trilha) com $v_0 = v_n$, então ρ define um ciclo (resp. trilha fechada).

Um grafo G é *conexo* se para todo par de vértices distintos v e w de G existe um caminho de v a w. Caso contrário, G é denominado *desconexo*. Uma componente conexa de G é um subgrafo maximal conexo de G, ou seja, é um subgrafo conexo que não está estritamente contido em outros subgrafos conexos de G.

Sejam $S, T \subseteq V(G)$ dois subconjuntos de vértices pertencentes ao conjunto de vértices do grafo G e X um subgrafo de G tais que cada caminho $S - T$ em G contém um vértice ou aresta pertencente ao subgrafo X, então podemos afirmar que X separa S e T em G. Em outras palavras, dizemos que X separa G ou X é o conjunto separador em G se $G - X$ possuir mais de um componente. Se o conjunto separador consistir de um único vértice, então temos um vértice de corte (em inglês, *cut vertex*). Se o conjunto separador consistir de uma única aresta, então temos uma ponte (em inglês, *bridge*). A Figura 1.3 ilustra um exemplo de um grafo contendo quatro vértices de corte e uma ponte.

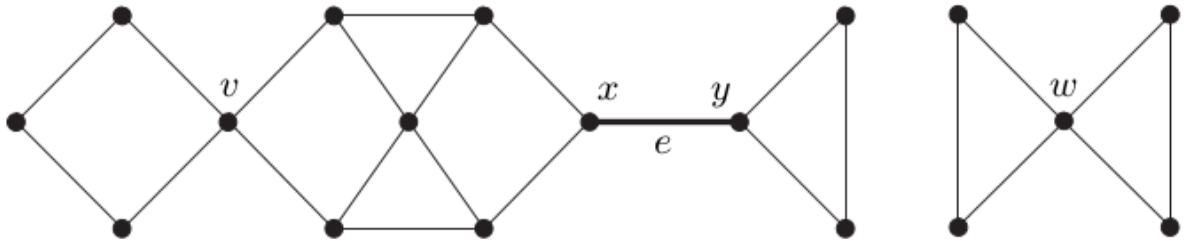

Figura 1.3: Um grafo com vértices de corte v, x, y, w e uma ponte $e = xy$.

Um *grafo orientado* ou *digrafo* $D = (V, \overrightarrow{E})$ é um objeto formado por dois conjuntos: um conjunto de vértices V e um conjunto de arcos $\overrightarrow{E}$. Cada arco é um par ordenado de vértices ($\overrightarrow{E} \subseteq V \times V$). O primeiro vértice do par é a ponta inicial do arco e o segundo é a ponta final do arco. Um arco com ponta inicial u e ponta final v é denotado por (u, v) ou $\overrightarrow{uv}$. Dizemos que um arco $\overrightarrow{uv}$ vai de u a v. Em outras palavras, dizemos que o arco $\overrightarrow{uv}$ sai de u e entra em v, ou que o arco é incidente do vértice u para o vértice v, ou divergente do vértice u e convergente ao vértice v.

O grau de saída de um vértice v é o número de arcos que saem deste vértice e analogamente o grau de entrada de um vértice v é o número de arcos que entram neste vértice.

Dizemos que um vértice v é adjacente a um vértice u se o par (u, v) é um arco, ou seja, se existe um arco que sai de u e entra em v. Nessas mesmas circunstâncias, dizemos também que v é um vizinho de u. A relação de adjacência não é simétrica: v pode ser adjacente a u sem que u seja adjacente a v. Portanto a vizinhança de saída de um vértice v é denotada por $N_D^+(v) = \{y \in V(D) : \overrightarrow{xy} \in E(D)\}$ e a vizinhança de entrada de um vértice v é denotada por $N_D^-(v) = \{y \in V(D) : \overrightarrow{yx} \in E(D)\}$. Desta forma o grau de saída de um vértice será denotado por $d_D^+(v) = |N_D^+(v)|$ e o grau de entrada de um vértice v será denotado por $d_D^-(v) = |N_D^-(v)|$. Um vértice com $d_D^-(v) = 0$ é chamado de fonte uma vez que é a origem de cada uma de suas arestas incidentes. Da mesma forma, um vértice com $d_D^+(v) = 0$ é chamado de sumidouro.

Seja $I_c = \{1, 2, ..., c\}$ um conjunto de cores com $c \geqslant 2$. Neste trabalho, denotamos G^c por um grafo simples, conectado, não-orientado, sem arestas paralelas, onde cada aresta tem uma cor pertencente ao conjunto I_c. Em outras palavras, G^c é um grafo com c cores nas arestas. Os conjuntos de vértices e arestas são denotados por $V(G^c)$ e $E(G^c)$, respectivamente, onde $|V(G^c)| = n$ e $|E(G^c)| = m$. Denotamos por $N_{G^c}(v)$ como o conjunto de todos os vértices vizinhos de v em G^c e por $N_{G^c}^i(v)$ o conjunto de vértices de G^c ligados ao vértice v com uma aresta de cor i. Para um grafo completo com c cores nas arestas de tamanho n denotamos por K_n^c ao invés de G^c. Para uma cor i qualquer, $E^i(G^c)$ denota o conjunto de arestas de G^c com a cor i. Uma aresta entre dois vértices x e y é denotada por xy e sua cor por $c(xy)$. Denotamos por $\chi(x) = \{i : i \in I_c,\ N_{G^c}^i(x) \neq \emptyset\}$ o conjunto de cores das arestas incidentes a um vértice x de G^c. O grau de um vértice v em G^c será denotado por $d_{G^c}(v)$ onde $d_{G^c}(v) = \sum_{i \in I_c} |N_{G^c}^i(v)|$, e o grau máximo de G^c será denotado por $\Delta(G^c)$ onde $\Delta(G^c) = max\{d_{G^c}(v) | x \in V(G^c)\}$.

Analogamente, para um digrafo D^c com c cores nas arestas e dois vértices $u, v \in V(D^c)$, denotamos por $\overrightarrow{uv}$ como um arco de $E(D^c)$ e sua cor por $c(\overrightarrow{uv})$. Adicionalmente, denotamos por $N_{D^c}^+(v) = \{u \in V(D^c) : \overrightarrow{vu} \in E(D^c)\}$ como a vizinhança de saída de um vértice v em D^c (onde o seu respectivo grau de saída é denotado por $d_{D^c}^+(v) = |N_{D^c}^+(v)|$) e por $N_D^-(v) = \{u \in V(D) : \overrightarrow{uv} \in E(D^c)\}$ como a vizinhança de entrada de um vértice v em D^c (onde seu grau de entrada é denotado por $d_D^-(v) = |N_{D^c}^-(v)|$).

Neste trabalho, o termo *propriamente colorido* é utilizado em um grafo G^c para dizer que duas arestas consecutivas possuem cores diferentes. O conceito de caminhos, ciclos, trilhas e trilhas fechadas propriamente coloridos, equivale ao mesmo conceito definido para caminhos, ciclos, trilhas e trilhas fechadas para grafos não-coloridos, com a restrição de que arestas sucessivas tenham cores distintas.

Como fontes de referência para esta seção, indicamos [9, 3, 21].

1.1.1 Emparelhamento em Grafos

O problema do emparelhamento em Grafos pode ser definido da seguinte maneira: Dado um grafo $G = (V, E)$ não-orientado, pretendemos encontrar um subconjunto M de arestas de tal forma que nenhuma aresta pertencente ao conjunto M compartilhe um vértice em comum com outra aresta pertencente a este mesmo conjunto M e que a cardinalidade de M seja máxima, ou seja, o conjunto M deverá conter o maior número de arestas não-adjacentes. Em outras palavras, trata-se de selecionar o maior número de pares de vértices adjacentes do grafo. A Figura 1.4 ilustra um exemplo de um emparelhamento em um grafo G. As arestas em negrito pertencem ao conjunto M.

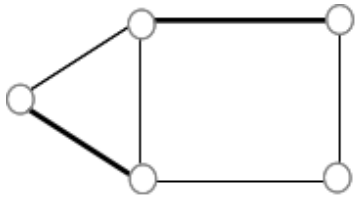

Figura 1.4: Exemplo de um emparelhamento em um grafo

Este problema é clássico no estudo algoritmos e otimização combinatória e possui várias aplicações como: determinação de rotas de veículos, determinação de percursos mínimos, problema do carteiro chines e muitos outros. O precursor no estudo deste problema foi Edmonds [10] quando criou um algoritmo em tempo polinomial $O(n^4)$ para o problema geral de emparelhamento. Implementações mais eficientes surgiram com os trabalhos de Hopcroft e Karp [22] (complexidade $O(m\sqrt{n})$) para o caso bipartido, Micali e Vazirani [30] (complexidade $O(m\sqrt{n})$) para o caso geral.

Seja um grafo $G = (V, E)$ com um emparelhamento $M \subseteq E$, então afirmamos que os vértices extremos de uma aresta $e \in M$ estão cobertos pelo emparelhamento ou *emparelhados* por M, ou simplesmente M-*emparelhados*. Um emparelhamento M *satura* um vértice v e v é denominado M-*saturado* se $e \in M$ é incidente a v, caso contrário, v não é M-*saturado* ou *livre*. Se $M' \subseteq M$ e M é um emparelhamento, então M' também define um emparelhamento. Um emparelhamento M é *máximo* ou de *cardinalidade máxima* em G se M contém o maior número possível de arestas, ou seja, se $|M'| > |M|$, então G não admite emparelhamento. Um emparelhamento M é *perfeito* se M cobre (satura) todos os vértices de G, ou seja, se todo vértice $v \in V$ é incidente a alguma aresta de M. Portanto, todo emparelhamento perfeito é máximo e que num grafo $G = (V, E)$ com

emparelhamento perfeito, $|V|$ é par e possui exatamente $|V|/2$ arestas.

1.1.1.1 Caminhos aumentantes - Teorema de Berge

O problema de determinar um emparelhamento máximo de um grafo, é resolvido pelo estudo de um tipo especial de caminho. Seja M um emparelhamento qualquer (não-vazio), um caminho M-*alternante* é um caminho em G que, para quaisquer duas arestas consecutivas, uma pertence a M e a outra não. Se o primeiro e o último vértice de um caminho M-alternante não são cobertos por M, dizemos que este caminho é M-*aumentante*. Surge então o teorema proposto por Berge [5] :

Teorema 1 *(Berge,1957) Um emparelhamento M tem cardinalidade máxima em G se e somente se G não possui caminho M-aumentante.*

Um algoritmo natural para encontrar um emparelhamento de cardinalidade máxima que decorre do Teorema 1 é começar com um emparelhamento vazio e, repetidamente, aumentar a cardinalidade do emparelhamento corrente através do uso sucessivo de caminhos aumentantes. Cabe ressaltar que a existência de um caminho M-aumentante P define, através da operação diferença simétrica, um novo emparelhamento $M' = M\Delta P$, onde $|M'| = |M| + 1$. Este processo de repetição de sucessivas atualizações na cardinalidades de emparelhamentos termina em um emparelhamento de cardinalidade máxima, uma vez que a cada iteração aumenta em uma unidade a cardinalidade do emparelhamento corrente e a cardinalidade do emparelhamento de cardinalidade máxima é finita. A complexidade do algoritmo acima é função da procura por caminhos aumentantes.

1.1.1.2 Emparelhamentos Perfeitos

Um emparelhamento é perfeito se cada vértice v é incidente a alguma aresta de M. Todo emparelhamento perfeito é um emparelhamento de cardinalidade máxima. Portanto, num grafo $G(V, E)$ com emparelhamento perfeito, $|v|$ é par e M possui exatamente $|V|/2$ arestas. Num emparelhamento perfeito M, todo vértice encontra-se M-saturado. A Figura 1.5 ilustra um exemplo de um emparelhamento perfeito em um grafo G.

O problema da determinação de emparelhamentos perfeitos teve como base o Teorema de *Hall* para grafos bipartidos e foi utilizado em muitas aplicações como por exemplo, o problema dos casamentos e de alocação de tarefas.

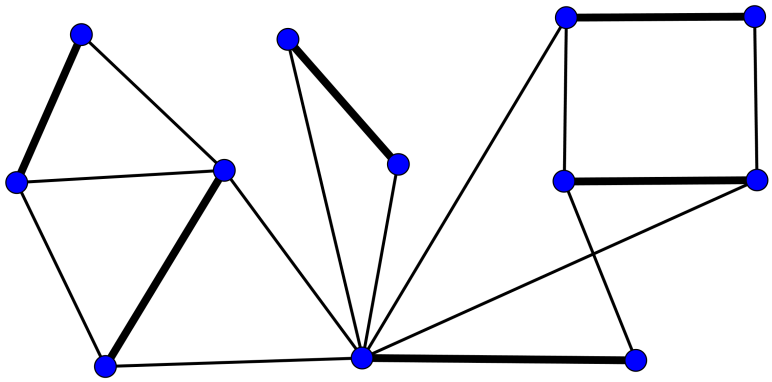

Figura 1.5: Exemplo de um emparelhamento perfeito no grafo G. As arestas em negrito, fazem parte do conjunto M

Uma outra abordagem para o problema do emparelhamento perfeito em grafos bipartidos é descrita pelo Algoritmo Húngaro proposto por Kuhn [26] . Dado um grafo $G = (V, E)$ com $2n$ vértices, bipartido em conjuntos X e Y, com $|X| = |Y| = n$, o Algoritmo Húngaro, em tempo $O(mn)$, ou encontra um emparelhamento que satura todo vértice de X ou encontra um subconjunto de X que viola a condição do Teorema de Hall. Em outras palavras, o Algoritmo Húngaro decide em tempo $O(mn)$ se este grafo admite emparelhamento perfeito.

Em se tratando de emparelhamento perfeito em grafos arbitrários, uma condição necessária e suficiente para que isto ocorra é dada pelo teorema Tutte [35]. Um componente de um grafo é par ou ímpar se este possuir um número par ou ímpar de vértices. Denotamos por $o(G)$ o número de componentes ímpares de G.

Teorema 2 *(Tutte,1947) Um grafo G tem um emparelhamento perfeito se e somente se $o(G - S) \leqslant |S|$ para todo $S \subset V$.*

A Figura 1.6 ilustra um exemplo de um grafo onde a condição proposta pelo Teorema 2 é satisfeita. O conjunto de vértices de S (marcado por uma elipse) possui 3 vértices enquanto que o numero de componentes ímpares do grafo $G - S$ é igual a 1.

Um algoritmo eficiente para o problema do emparelhamento perfeito em grafos arbitrários (não-bipartidos) foi proposto inicialmente por Edmonds [10] em 1965. O algoritmo de Edmonds generalizou o algoritmo para o caso bipartido e mostrou como se podia contrair *florações* e procurar por caminhos aumentantes de modo eficiente. O conceito de floração foi então definido como um ciclo com $2k + 1$ vértices onde k arestas estão emparelhadas.

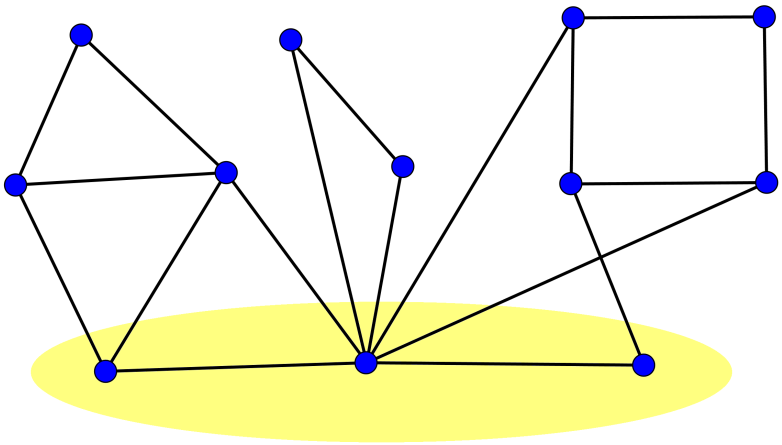

Figura 1.6: Ilustração da condição necessária para a existência de um emparelhamento perfeito. A Figura sugere um grafo G e um conjunto S marcado por uma elipse vértices.

Portanto se um grafo não tem florações em relação ao emparelhamento corrente, então a procura por caminhos aumentantes pode ser realizada como um grafo bipartido, aplicando por exemplo o algoritmo húngaro. O diferencial do algoritmo de Edmonds em relação ao algoritmo húngaro é modifica-lo para que, caso existam florações, estas sejam detectadas e mesmo na presença delas, o algoritmo continue encontrando caminhos aumentantes válidos.

Neste trabalho, utilizamos o algoritmo de emparelhamento perfeito de custo mínimo denominado *Blossom V* desenvolvido por Kolmogorov [25]. Este algoritmo é a mais recente implementação encontrada na literatura baseada no algoritmo *Blossom I* desenvolvido por Edmonds [11] em 1969.

Como fontes de referência para esta seção, indicamos [6, 9, 13].

1.1.2 Representação Computacional

Existem pelo menos três maneiras de representar um grafo no computador. As mais conhecidas são: matriz de adjacência, matriz de incidência e listas de adjacência. Neste trabalho utilizamos listas de adjacências. Na representação de um grafo $G = (V, E)$ através de listas de adjacência, tem-se, para cada vértice u, uma lista de arestas incidentes a u. Desta forma, para cada vértice u o conjunto $E(u)$ é representado por uma lista. Nesta representação o espaço gasto é proporcional a $n+m$, onde n é o número de vértices e m é o número de arestas do grafo. Na Figura 1.7 temos a ilustração de uma lista de adjacência

para um grafo com cinco vértices e sete arestas.

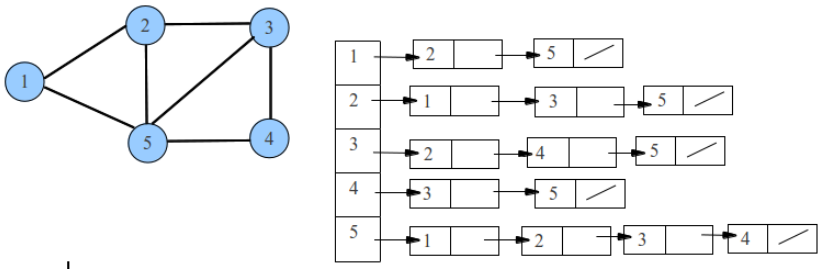

Figura 1.7: Exemplo de grafo e sua representação em lista de adjacências

1.1.2.1 Stanford GraphBase

O Stanford GraphBase (SGB) é uma plataforma para algoritmos combinatórios e estruturas de dados desenvolvida por Knuth [24] na linguagem C. O SGB é um pacote que contém dois tipos de módulos:
1. módulos que geram uma grande variedade de grafos interessantes;
2. módulos que resolvem certos problemas combinatórios sobre alguns dos grafos gerados pelos módulos do tipo 1.

Dentre os diversos módulos do SGB, utilizamos em nosso trabalho o módulo GB_GRAPH. Este módulo define uma estrutura de dados padrão para grafos além de incluir rotinas para gerenciamento de alocação de dados.
A representação de um grafo utiliza estruturas (*structs*) para vértices, arcos e grafos, além de funções básicas para manipular estas estruturas. As principais estruturas são descritas a seguir:

1. **Vertex (Vértice)**: Cada vértice é representado no SGB através de uma estrutura do tipo **Vertex** com dois campos padrão e seis campos de utilidade geral (do tipo util). Os campos padrão são:

- *arcs*: Ponteiro para um **Arc**. Armazena uma lista de arestas;
- *name*: Ponteiro para uma cadeia de caracteres. Identifica simbolicamente cada vértice.

Se v é do tipo **Vertex**, denotamos como $v \rightarrow arcs$ como a lista de adjacência (onde cada aresta incidente a v é uma estrutura do tipo **Arc**) de v. Em outras palavras, $v \rightarrow arcs$

representa o conjunto de arestas incidentes a v. Se v não tem arestas incidentes, dizemos que $v \rightarrow arcs$ é NULL.

2. **Arc (Arco)**: Cada arco é representado por uma estrutura do tipo **Arc**. Cada **Arc** tem três campos padrão e dois campos para uso geral. Os campos padrão são:

- *tip*: Um ponteiro para um **Vertex**;
- *next*: Um ponteiro para um **Arc**;
- *len*: Um inteiro (long) representado o peso do arco;

Se a aponta para um **Arc** em uma lista de arcos saindo de v, significa dizer que que a representa um arco de peso $a \rightarrow len$ indo de v até $a \rightarrow tip$ e o próximo arco saindo de v na lista é representado por $a \rightarrow next$. Os campos para uso geral são chamados a e b. Estes campos foram utilizados na nossa implementação para representar respectivamente a cor e o nome (identificador) da aresta.

3. **Grafo (Graph)**: Esta estrutura pode ser passada para um algoritmo que trabalha sobre grafos. Uma estrutura do tipo **Graph** tem sete campos padrão e seis campos para uso geral. Os campos utilizados neste trabalho foram:

- *vertices*: Um ponteiro para um vetor de **Vertex**;
- n: O número total de vértices;
- m: O número total de arcos;
- *id*: Um identificador simbólico;

Os campos para uso geral são uu, vv, ww, xx, yy, zz. Os campos uu e vv foram utilizados na implementação do algoritmo correspondente ao gadget XP-Gadget proposto por Gutin et al. em [20]. Os campos ww, xx e yy foram utilizados na implementação da transformação de uma trilha s-t propriamente colorida em um caminho s-t propriamente colorido e o campo zz foi utilizado na implementação da verificação da existência de um ciclo propriamente colorido em um Grafo com arestas coloridas G^c, ambas propostas em [1].
Este trabalho foi implementado apenas para grafos não-direcionados, portanto cada criação de uma nova aresta xy significa na verdade a criação de dois novos arcos: $x \rightarrow y$ e $y \rightarrow x$. A Figura 1.8 ilustra o exemplo de um grafo e sua representação no SGB.

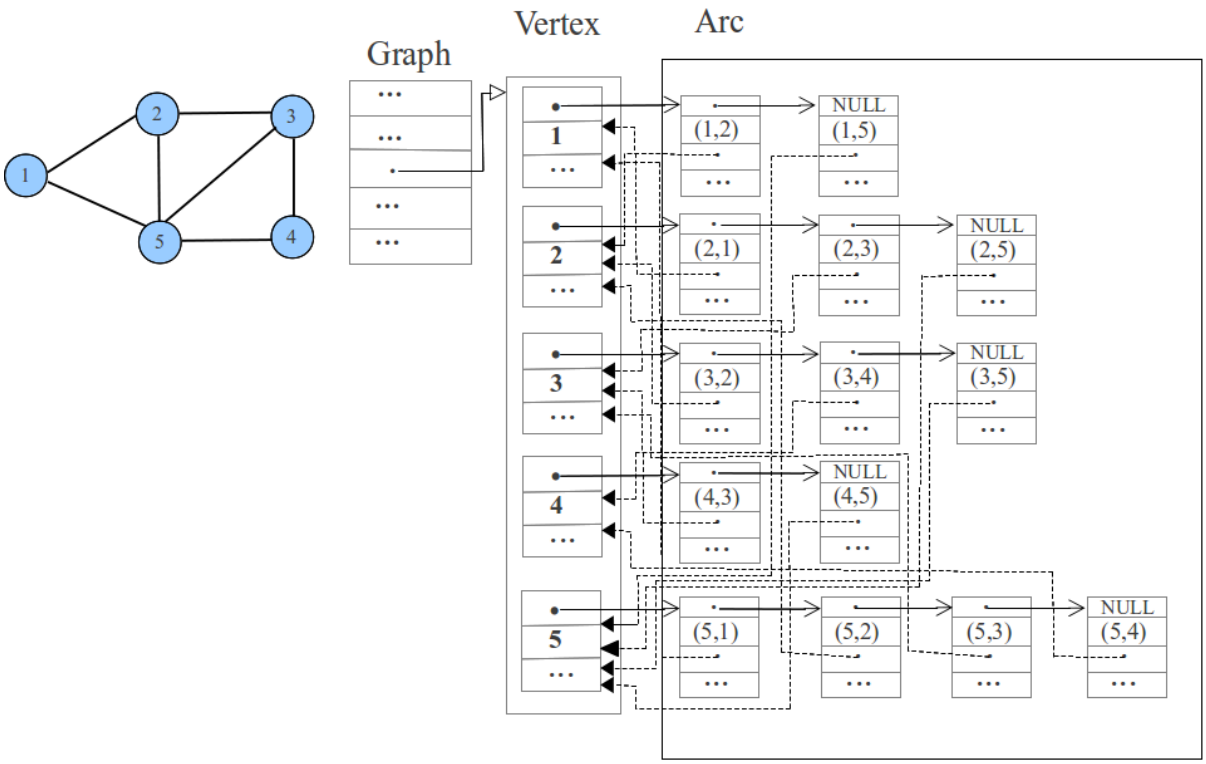

Figura 1.8: Exemplo de um grafo não-orientado e sua representação no SGB

O objetivo geral desta dissertação é propor uma metodologia para resolução de problemas relacionados a grafos com cores nas arestas tais como a determinação de caminhos, trilhas, ciclos e trilhas fechadas através da implementação de algoritmos eficientes.

Os objetivos específicos desta dissertação são os seguintes:

- Fazer um levantamento bibliográfico relacionado ao problema em questão.
- Implementar um sistema computacional na linguagem C/C++, incorporando *solvers* relacionados à resolução de emparelhamento perfeito de custo mínimo em grafos e problemas de programação linear inteira, baseado nas metodologias utilizadas para resolver o problema em questão.
- Criar e resolver um conjunto de problemas-teste dos problemas em questão.
- Desenvolver um modelo matemático para representar um problema de programação linear inteira, quando for necessário.

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2, introduzimos conceitos importantes a respeito da teoria de caminhos e ciclos, mais especificamente sobre caminhos e ciclos propriamente coloridos, além de apresentar os trabalhos relacionados referentes ao tema desta dissertação. No Capítulo 3, introduzimos os conceitos sobre construções P-Gadgets e descrevemos em seguida o algoritmo XP-Gadget (que foi escolhido para ser utilizado em nossas implementações), além da descrição de todos os algoritmos implementados. No Capítulo 4, descrevemos o modelo de Programação Linear Inteira que foi criado para solucionarmos o problema do maior caminho/trilha propriamente colorido quando há a presença de ciclos/trilhas fechadas propriamente coloridas na solução inicial encontrada. No Capítulo 5, descrevemos os experimentos computacionais realizados, comparando e analisando os algoritmos para o problema do menor caminho propriamente colorido em um grafo G^c. Finalmente, o Capítulo 6 contém as conclusões deste trabalho juntamente com as sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica e Aplicações

Neste capítulo, revisaremos conceitos e características importantes relacionadas ao trabalho proposto. Mais especificamente, trataremos de problemas relacionados a caminhos, ciclos, trilhas e trilhas fechadas propriamente coloridas encontrados na literatura. Nos trabalhos relacionados descreveremos os problemas de caminhos e trilhas com custo de reconexão simétricos e caminhos com recoloração de arestas, mais especificamente, a determinação do menor custo de recoloração nas arestas de um Grafo G^c para a construção de um $s-t$ caminho propriamente colorido. Em seguida, descreveremos uma aplicação na área de biologia molecular que trata da reconstrução de caminhos ótimos em mapas de ressonância magnética nuclear tridimensional através da formulação de grafos com cores nas arestas.

2.1 Caminhos/Ciclos/Trilhas/Trilhas Fechadas Propriamente Coloridos(as) em um Grafo G^c

Considerando grafos com arestas coloridas, nosso foco de estudo concentra-se em encontrar subgrafos com um padrão específico de cores, ou seja, subgrafos onde pares de arestas adjacentes possuam cores diferentes ou mais especificamente, *subgrafos propriamente coloridos*.

Um caminho ou trilha em um grafo G^c, é denominado *propriamente colorido* se todos os pares de arestas sucessivas possuírem cores diferentes (veja exemplos na Figura 2.1 b) c) d)) . Vale ressaltar que as arestas pertencentes a uma trilha propriamente colorida, podem não formar um subgrafo propriamente colorido, uma vez que a condição de termos arestas sucessivas de cores diferentes não implica que teremos um subgrafo com pares de

arestas adjacentes de cores diferentes (veja exemplo na Figura 2.1 e)). Uma variação de caminhos e trilhas propriamente coloridos ocorre quando seus vértices terminais coincidem ($s = t$) e quando as arestas inicial e final possuem cores diferentes. Neste caso estamos falando de caminhos fechados (também chamados de ciclos) propriamente coloridos(veja exemplo na Figura 2.1) e trilhas fechadas propriamente coloridas.

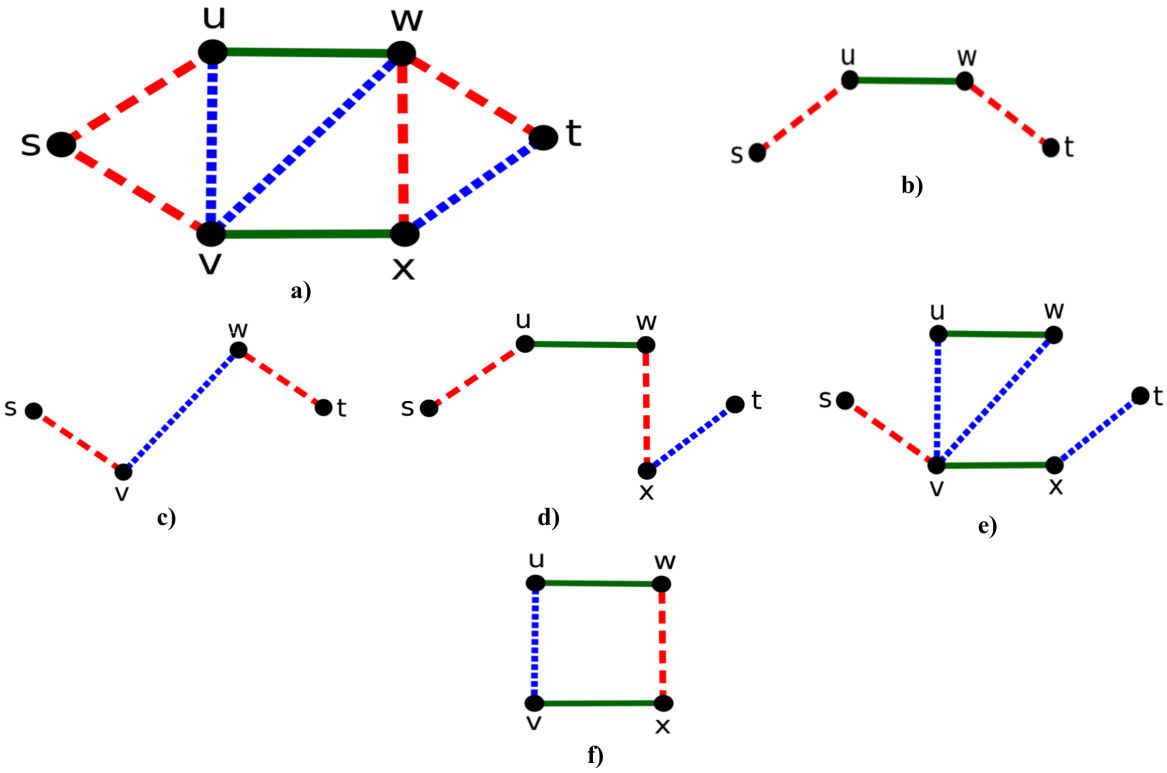

Figura 2.1: a)Um Grafo com 3-cores nas arestas; b),c),d): $s-t$ caminhos propriamente coloridos; e)$s-t$ trilha propriamente colorida; f) ciclo propriamente colorido

O problema da determinação de $s-t$ caminhos propriamente coloridos foi inicialmente resolvido por Edmonds através do Lemma a seguir citado em [28].

Lemma 1 *(J.Edmonds) Seja s,t dois vértices em um grafo com 2 cores nas arestas de ordem n. Existe um algoritmo $O(n^{2.5})$ para encontrar um caminho $s-t$ alternante em que a primeira e ultima arestas tenham a mesma cor.*

Manoussakis [28] mostra que encontrar um caminho alternante ou propriamente colorido para um grafo com 2 cores nas arestas é similar a encontrar um caminho aumentante no algoritmo de Edmonds entre dois vértices não-saturados, associando por exemplo as arestas pertencentes ao emparelhamento M a uma cor i e as arestas não pertencentes

ao emparelhamento M a uma cor j. A complexidade $O(n^{2.5})$ é justamente a complexidade do melhor algoritmo na época para calcular um emparelhamento máximo em grafos [12]. Mais tarde Szeider [34] estendeu a resolução do problema da determinação de $s-t$ caminhos propriamente coloridos para um grafo com qualquer número de cores.

Uma importante contribuição para caminhos $s-t$ propriamente coloridos em grafos G^c foi proposta por Szeider [34] no seguinte teorema:

Teorema 3 *(Szeider, 2003) Seja s e t dois vértices em um grafo G^c com c arestas coloridas para $c \geq 2$. Portanto, ou encontramos um caminho $s-t$ propriamente colorido ou então decidimos que tal caminho em G^c não existe em tempo polinomial para o tamanho do grafo.*

A idéia central da prova deste teorema baseia-se na ideia inicial de Edmonds (proposta no Lemma 1) e consiste em reduzir o problema de caminhos propriamente coloridos em grafos G^c para o problema de calcular o emparelhamento perfeito em um grafo não-colorido equivalente. Na Seção 3.1 referente aos P-Gadgets, abordamos os principais tipos de construções presentes na literatura e neste trabalho optaremos pela construção XP-Gadget definida nas expressões 3.5 e 3.6.

Para discutirmos a caracterização de grafos contendo ciclos propriamente coloridos, explicaremos o conceito de *vértice de corte separando cores (em inglês, cut vertex separating colors)*. Um vértice v é um *vértice de corte separando cores* se nenhum componente de $G^c \setminus v$ é ligado a v com pelo menos duas arestas de cores diferentes. O exemplo da Figura 2.2 ilustra um grafo onde os vértices v e w são vértices de corte separando cores.

Em se tratando de ciclos propriamente coloridos em grafos G^c, Yeo [39] generalizou o problema proposto por Grossman and Häggkvist [19] para grafos com 2 ou mais cores nas arestas, através do seguinte teorema:

Teorema 4 *(Yeo, 1997) Seja G^c um grafo com c cores nas arestas com $c \geq 2$, tal que cada vértice seja incidente em pelo menos duas arestas de cores diferentes. Então, ou G^c tem um ciclo propriamente colorido ou G^c tem um vértice de corte separando cores.*

Este resultado proposto no Teorema 4 pode ser utilizado para se determinar se um grafo G^c possui ou não ciclos propriamente coloridos. Yeo [39] propõe um algoritmo onde todos os *vértices de corte separando cores* são retirados até que o conjunto de arestas seja vazio (neste caso G^c não contém ciclos propriamente coloridos) ou o grafo resultante

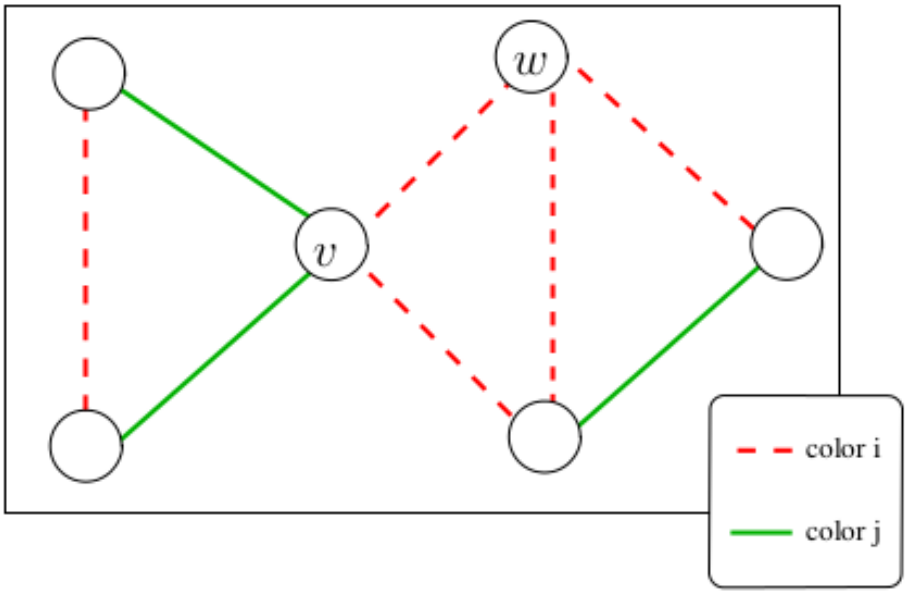

Figura 2.2: Um grafo com 2 vértices de corte separando cores: v e w [27]

contenha vértices com pelo menos duas arestas incidentes com cores diferentes (neste caso, G^c contém ciclos propriamente coloridos. Descreveremos este procedimento com mais detalhes no Capítulo 3.

Para o problema da determinação de uma $s-t$ trilha propriamente colorida em um grafo G^c, Abouelaoualim [1] provou que este problema pode ser reduzido à resolução de um $s-t$ caminho propriamente colorido sobre um novo grafo (denominado grafo trilha-caminho) equivalente $p-H^c$, onde $p \geqslant 2$, onde para um Grafo G^c com n vértices, o número máximo de visitas para um vértice intermediário $v \in V(G^c) \setminus \{s,t\}$ é dado pela fórmula: $p = \lfloor \frac{n-1}{2} \rfloor$. Para $p = 2$, o grafo trilha-caminho se resume a apenas H^c ao invés de $p-H^c$. Esta relação entre o grafo G^c contendo uma $s-t$ trilha e um grafo trilha-caminho H^c contendo um $s-t$ caminho, foi descrita no Teorema a seguir:

Teorema 5 *(Abouelaoualim et al., 2008) Dados dois vértices s e t em G^c, então existe uma $s-t$ trilha propriamente colorida em G^c, se e somente se, existir um $s-t$ caminho propriamente colorido em H^c*

A determinação do grafo $p-H^c$ contendo um $s-t$ caminho propriamente colorido correspondente à $s-t$ trilha propriamente colorida em G^c é descrita a seguir:

Seja G^c um grafo com c arestas coloridas e um inteiro $p \geqslant 2$, um novo grafo com arestas coloridas denominado $p-H^c$ (grafo trilha-caminho) é obtido de G^c da seguinte

forma:
Cada vértice $x \in G^c$ é substituído por p novos vértices $x_1, x_2, \cdots, x_p$. Para cada aresta $xy \in G^c$ com uma cor j, adiciona-se dois novos vértices v_{xy} e u_{xy}. Adiciona-se as arestas $x_i v_{xy}, u_{xy} y_i$, onde $i = 1, 2, \cdots, p$, com a cor j da aresta original xy. Em seguida, adiciona-se a aresta $v_{xy}u_{xy}$ com uma cor $j' \in \{1, 2, \cdots, c\}$ com $j \neq j'$.

O subgrafo com arestas coloridas $p-H^c$ induzido pelos vértices x_i, v_{xy}, u_{xy}, y_i (*para* $i = 1, 2, \cdots, p$) associado com a aresta xy de G^c é denotado por H^c_{xy}. Para $p = 2$, o subgrafo $p - H_c$ se reduz a H^c e neste caso determinaremos a menor trilha propriamente colorida em G^c.

As Figuras 2.3 e 2.4 definidas em [1] ilustram, respectivamente, exemplos de transformação para um subconjunto $E^i(G^c)$ e para uma trilha com 2 arestas coloridas.

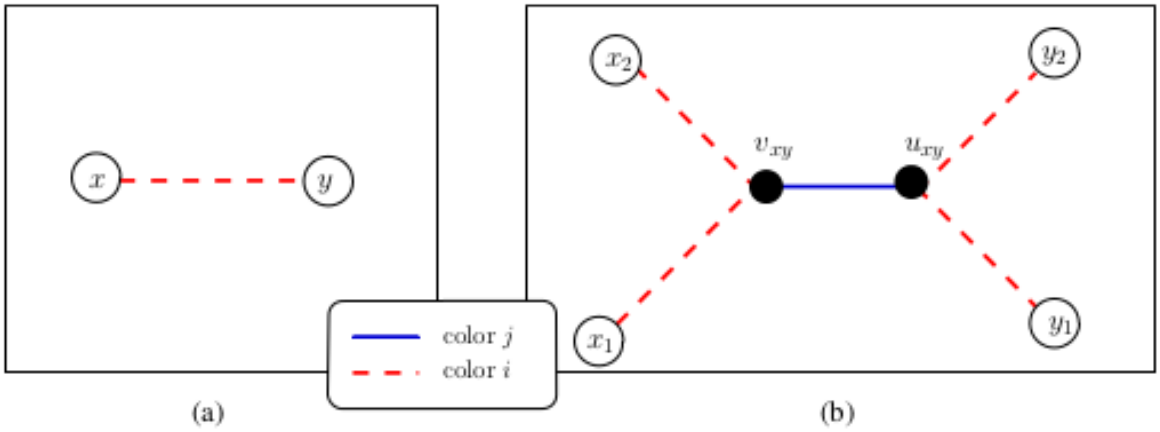

Figura 2.3: (a) Aresta $xy \in E^i(G^c)$. (b) Subgrafo H^c_{xy} *associado com* xy

Abouelaoualim [1], generalizou o teorema proposto por Yeo [39] para resolução de trilhas fechadas propriamente coloridas, através do seguinte teorema:

Teorema 6 *(Abouelaoualim et al., 2008) Seja G^c um grafo com c arestas coloridas tal que todo vértice de G^c é incidente com pelo menos duas arestas de cores diferentes. Então, ou G^c tem uma ponte ou G^c tem uma trilha fechada propriamente colorida.*

Relembrando que uma ponte é uma aresta cuja remoção aumenta o número de componentes conexas no grafo original G^c. Na Figura 2.5, temos um exemplo de um grafo com uma ponte. Perceba que se removermos esta aresta, teremos duas componentes conexas no grafo resultante.

Analogamente ao que foi mencionado para ciclos propriamente coloridos, eliminações

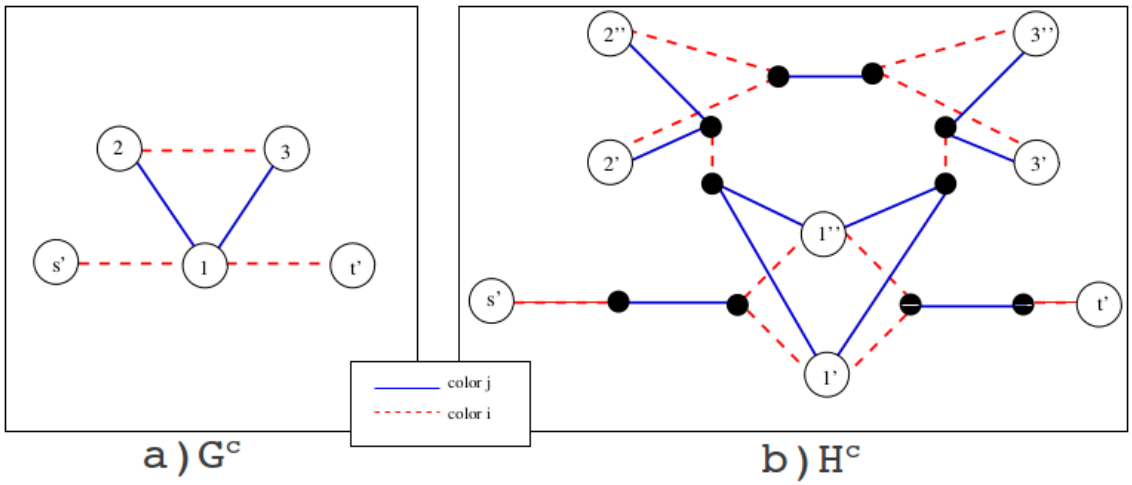

Figura 2.4: Transformação de uma $s-t$ trilha com 2 arestas coloridas (a) em um $s-t$ caminho propriamente colorido (b).

sucessivas de pontes e vértices monocromáticos podem detectar ou não a presença de trilhas fechadas propriamente coloridas.

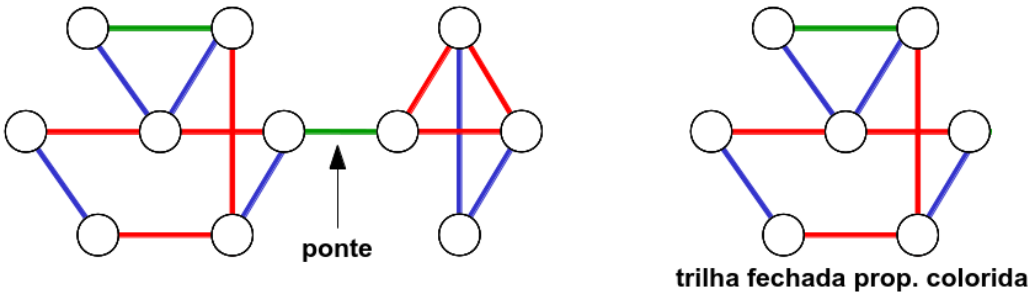

Figura 2.5: Um grafo com uma ponte

2.2 Trabalhos Relacionados

A caracterização de grafos G^c com c cores nas arestas contendo ciclos propriamente coloridos foi inicialmente abordada por Yeo em [39] sendo mais tarde generalizada em [1] para trilhas fechadas propriamente coloridas. Vários problemas a respeito de caminhos/trilhas entre dois vértices s, t em um grafo G^c com c cores nas arestas são abordados em Abouelaoualim et al. [1]. Mais especificamente, algoritmos polinomiais são estabelecidos para determinar problemas como: menor $s-t$ caminho/trilha propriamente colorido(a), menor $s-t$ ciclo/trilha fechada propriamente colorido(a) e maior $s-t$ caminho/trilha para uma particular classe de instâncias. Os autores em [1] provaram ainda que para um inteiro $k \geqslant 2$, decidir se existem k $s-t$ caminhos/trilhas propriamente coloridos disjuntos

em vértices/arestas em um grafo G^c é **NP**-Completo mesmo para $k = 2$ e $c = \Omega(n^2)$. Além disso, os autores em [1] também provaram que para um k arbitrário, estes problemas permanecem **NP**-Completos para grafos G^c sem ciclos/trilhas fechadas propriamente coloridos(as) e $c = \Omega(n)$.

2.2.1 Caminhos e trilhas com custos de reconexão

Em Gourvès et al. [17], os autores abordam problemas relacionados a $s - t$ caminhos, trilhas e passeios onde o objetivo é minimizar o custo total de reconexão desta rota. Um custo de reconexão é calculado da seguinte forma: cada vez que um vértice v é visitado por um passeio, trilha ou caminho é associado um custo não-negativo de reconexão $r_{i,j}$, onde i, j denotam respectivamente as cores das arestas sucessivas desta rota. Mais especificamente, os autores consideram o seguinte modelo: Dado grafo G^c, um par de vértices $s, t \in V(G^c)$, uma matriz $c \times c$ $R = [r_{i,j}]$ com i, j pertencendo a um conjunto de cores $I_c = \{0, 1, \ldots, c\}$ associada a um custo não-negativo para cada par de cores e um caminho/trilha/passeio $\rho = (v_0, e_0, v_1, e_1, \ldots, e_k, v_{k+1})$ entre vértices s, t, então o custo de reconexão de ρ é definido como:

$$minimizar\ (\rho) = \sum_{j=0}^{k-1} r_{c(e_j),c(e_{j+1})} \tag{2.1}$$

Em outras palavras, o objetivo é encontrar um caminho/trilha/passeio ρ entre s, t com custo de reconexão mínimo. O menor $s - t$ caminho de reconexão (resp. trilha de reconexão) também está relacionado com o problema de decidir se um grafo G^c tem um $s - t$ caminho propriamente colorido (resp. $s - t$ trilha propriamente colorida). Por exemplo, se for atribuído o custo de reconexão $r_{i,i} = 1$ e $r_{i,j} = 0$ para $i, j \in I_c$ com $i \neq j$ então existe um $s-t$ caminho (resp. trilha) com custo de reconexão igual a 0 se e somente se existir um $s - t$ caminho propriamente colorido (resp. trilha propriamente colorida) em G^c. Analogamente, existe um $s - t$ caminho monocromático se e somente se existir um $s - t$ caminho em G^c com $r_{i,i} = 0$ e $r_{i,j} = 1$.

A escolha de um $s-t$ passeio ao invés de um $s-t$ caminho pode significar uma redução no custo total de reconexão em um dado grafo G^c. A Figura 2.6 a seguir ilustra dois exemplos diferentes de $s - t$ passeios: $\rho 1 = (s, e_1, v_1, e_3, t)$ e $\rho 2 = (s, e_1, v_1, e_2, v_2, e_2, v_1, e_3, t)$ com custos de reconexão $r_{i,j}$ definidos na matriz de custos R para um conjunto de cores $I_c = \{1, 2, 3\}$.

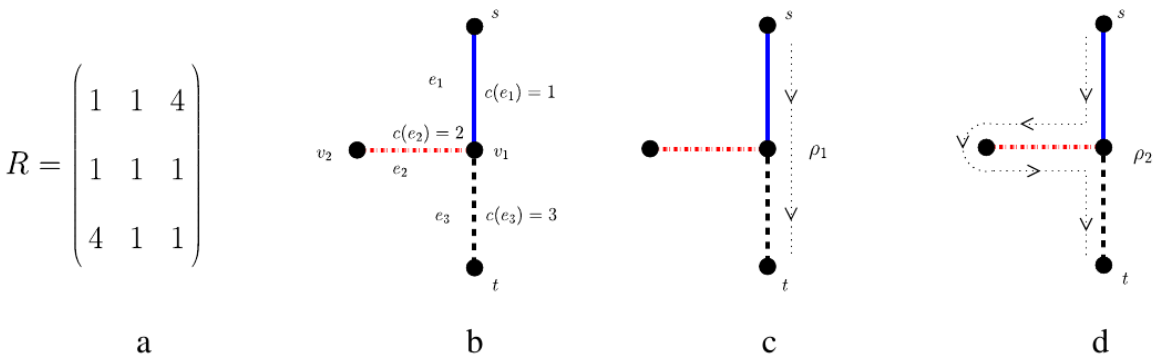

Figura 2.6: a) Uma matriz R com os custos de reconexão associados ao conjunto de cores $I_c = \{1, 2, 3\}$; b) Grafo G^c original com 3 cores nas arestas; c) Passeio $\rho 1$ com custo de reconexão $r(\rho 1) = 4$; d)Passeio $\rho 2$ com custo de reconexão $r(\rho 2) = 3$. [17]

Analisando a Figura 2.6, o custo de reconexão do passeio $\rho 1$ é menor que o custo de reconexão do passeio $\rho 2$, onde $\rho 2$ também é um caminho. Uma vez que todo passeio é uma trilha e toda trilha é um caminho, um $s-t$ passeio com custo de reconexão mínimo é menor ou igual a uma $s-t$ trilha com custo de reconexão mínimo que é menor ou igual a um $s-t$ caminho com custo de reconexão mínimo.

Os autores em [17] provaram que o problema da $s-t$ trilha com custo de conexão mínimo pode ser resolvido em tempo polinomial para $c \geqslant 2$ e uma matriz simétrica R com custos não-negativos. Também foi provado que o problema do $s-t$ caminho com custo de reconexão mínimo pode ser resolvido em tempo polinomial para $c = 2$ mantendo-se a desigualdade triangular ou se G^c tiver grau máximo igual a 3. Para $c \geq 3$ mantendo-se a desigualdade triangular o problema torna-se **NP**-Difícil e permanece **NP**-Difícil para $c \geq 4$ mantendo-se a desigualdade triangular mesmo para grafos planares com grau máximo igual a 4.

Para o problema da $s-t$ trilha com custo de reconexão mínimo, os autores em [17], mostraram que este problema pode ser reduzido ao problema de emparelhamento perfeito de custo mínimo em um grafo não-colorido G.

Dados dois vértices $s, t \in V(G^c) = \{v_1, \ldots, v_n\}$, $W = V(G^c) \setminus \{s, t\}$. Para cada vértice $v_i \in W$, são definidos subgrafos G_i, com seus conjuntos de vértices e arestas definidos da seguinte forma:

$$V(G_i) = \{v_{i,j}, v'_{i,j} : v_j \in N_{G^c}(v_i)\} \cup \{p^i_{j,k}, q^i_{j,k} : j < k \text{ e } v_j, v_k \in N_{G^c}(v_i)\} \tag{2.2}$$

$$E(G_i) = \{[v_{i,j}, v'_{i,j}] : v_j \in N_{G^c}(v_i)\} \cup$$

$$\{[v'_{i,j}, p^i_{j,k}], [p^i_{j,k}, q^i_{j,k}], [q^i_{j,k} v'_{i,k}] : j < k \ e \ v_j, v_k \in N_{G^c}(v_i)\} \tag{2.3}$$

O grafo resultante $G = (V', E')$ não-colorido com pesos nas arestas denotado por w é definido da seguinte forma:

$$V' = \{s', t'\} \cup (\bigcup_{v_i \in W} V(G_i)\} \tag{2.4}$$

$$E' = \{ \ [v_{i,j}, v_{x,y}] : j = x \ e \ i = y\} \cup$$

$$\{[s', v_{i,j}] : v_j = s \ e \ [v_i, v_j] \in E(G^c)\} \ \cup \ \{[v_{i,j}, t'] : v_j = t \ e \ [v_i, v_j] \in E(G^c)\} \tag{2.5}$$

$$w([v'_{i,j}, p^i_{j,k}]) = {}^{r_{c([v_i,v_j]),c([v_i,v_k])}}/_2, \ \ w([v'_{i,k}, q^i_{j,k}]) = {}^{r_{c([v_i,v_k]),c([v_i,v_j])}}/_2 \tag{2.6}$$

Todas as arestas restantes tem peso igual a 0. A Figura 2.7 a seguir ilustra a construção de um subgrafo G_i para um dado vértice v_i de G^c.

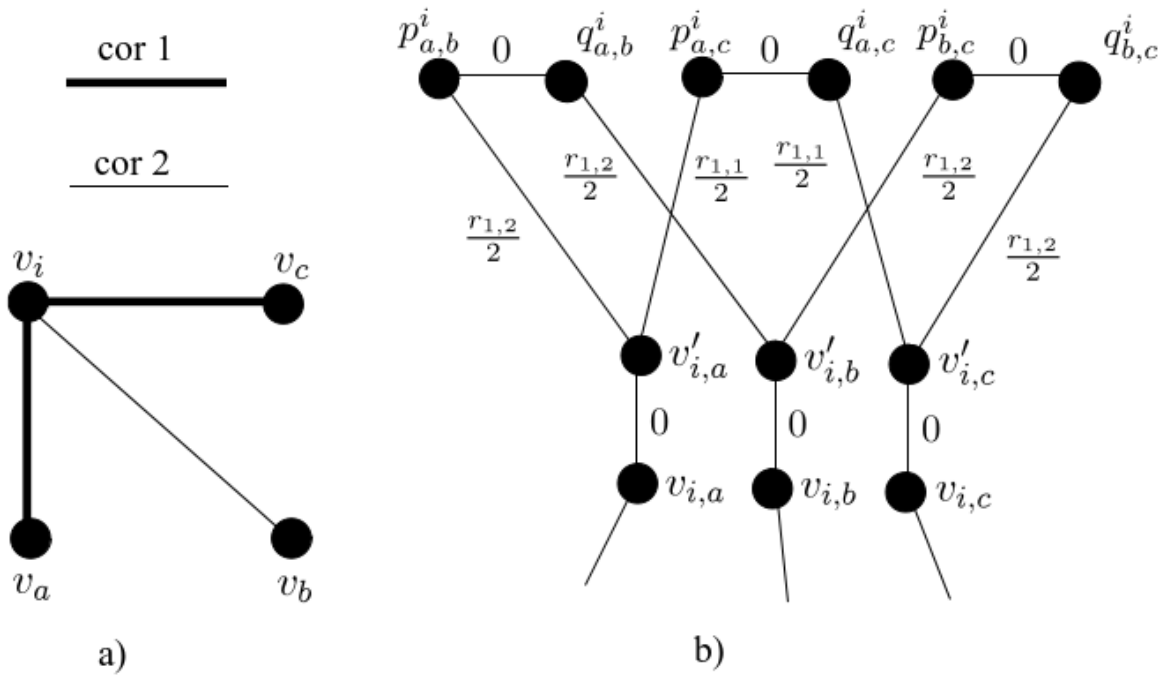

Figura 2.7: Redução do problema de uma s-t trilha com custo de reconexão mínimo em um emparelhamento perfeito de custo mínimo. a) v_i e sua vizinhança em G^c; b) Subgrafo G_i correspondente em um grafo não-colorido G.[17]

Após a construção do grafo G, é necessário encontrar um emparelhamento perfeito de custo mínimo M^* em G. O conjunto M^* encontrado terá um conjunto de arestas não-adjacentes que sejam incidentes a todos os vértices em G. O custo total do emparelhamento M^* é dado pela fórmula: $w(M^*) = \sum_{e \in M^*} w(e)$.

Como sabemos que o cálculo do emparelhamento perfeito de custo mínimo é polinomial (veja Seção 1.1.1), Formalmente:

Teorema 7 *(Gourvès, 2010) Dado um grafo G^c conectado com c cores nas arestas e um par de vértices $s, t \in G^c$ o problema da $s-t$ trilha com custo de reconexão simétrico mínimo pode ser calculado em tempo polinomial.*

Para a prova deste teorema, os autores em [17] precisaram provar apenas que uma $s-t$ trilha com custo de reconexão mínimo equivale a um emparelhamento perfeito de custo mínimo ($w(M) = r(\rho_c)$).

Dado um grafo $G = (V', E')$ não-direcionado. $G = (V', E')$ pode ser polinomialmente construído a partir de um grafo G^c, de acordo com a construção mostrada na Figura 2.7. Uma vez definido um emparelhamento perfeito de custo mínimo M em G denotado por $w(M) = \sum_{e \in M} w(e)$, a $s-t$ trilha com custo de reconexão mínimo equivalente, pode ser obtida após a contração de todos os subgrafos G_i em G e pela associação das arestas restantes não-coloridas em G com as arestas coloridas em G^c. Considerando a matriz de custos simétrica e a definição dos pesos das arestas na equação 2.6, então:

$w([v'_{i,j}, p^i_{j,k}]) + w([v'_{i,k}, q^i_{j,k}]) = r_{c([v_i,v_j]),c([v_i,v_k])}$

Portanto, pode-se facilmente afirmar que: $w(M) = r(\rho_c)$.

De uma outra maneira, dado uma $s-t$ trilha $\rho_c \in G^c$ com custo de reconexão $r(\rho_c)$, um emparelhamento perfeito de custo mínimo associado, é construído da seguinte forma:

- Para cada vértice $v_i \in G^c$, onde $v_i \notin \rho_c$, todas as arestas com peso 0 em G_i são escolhidas.
- Para cada vértice $v_i \in G^c$, onde $v_i \in \rho_c$, se ρ_c contém uma sequencia: (v_a, e, v_i, e', v_b). com $e \neq e'$ e $a < b$, as arestas $[v'_{i,a}, p^i_{a,b}]$ e $[q^i_{a,b}, v'_{i,b}$ são escolhidas em G_i.
- Todas as arestas restantes com custo 0 em G são escolhidas a fim de obter o emparelhamento perfeito de custo mínimo em G.

Portanto é fácil afirmar que: $w(M) = r(\rho_c)$.

Um exemplo completo de um grafo G^c e seu grafo equivalente G não-colorido é ilustrado na Figura 2.8.

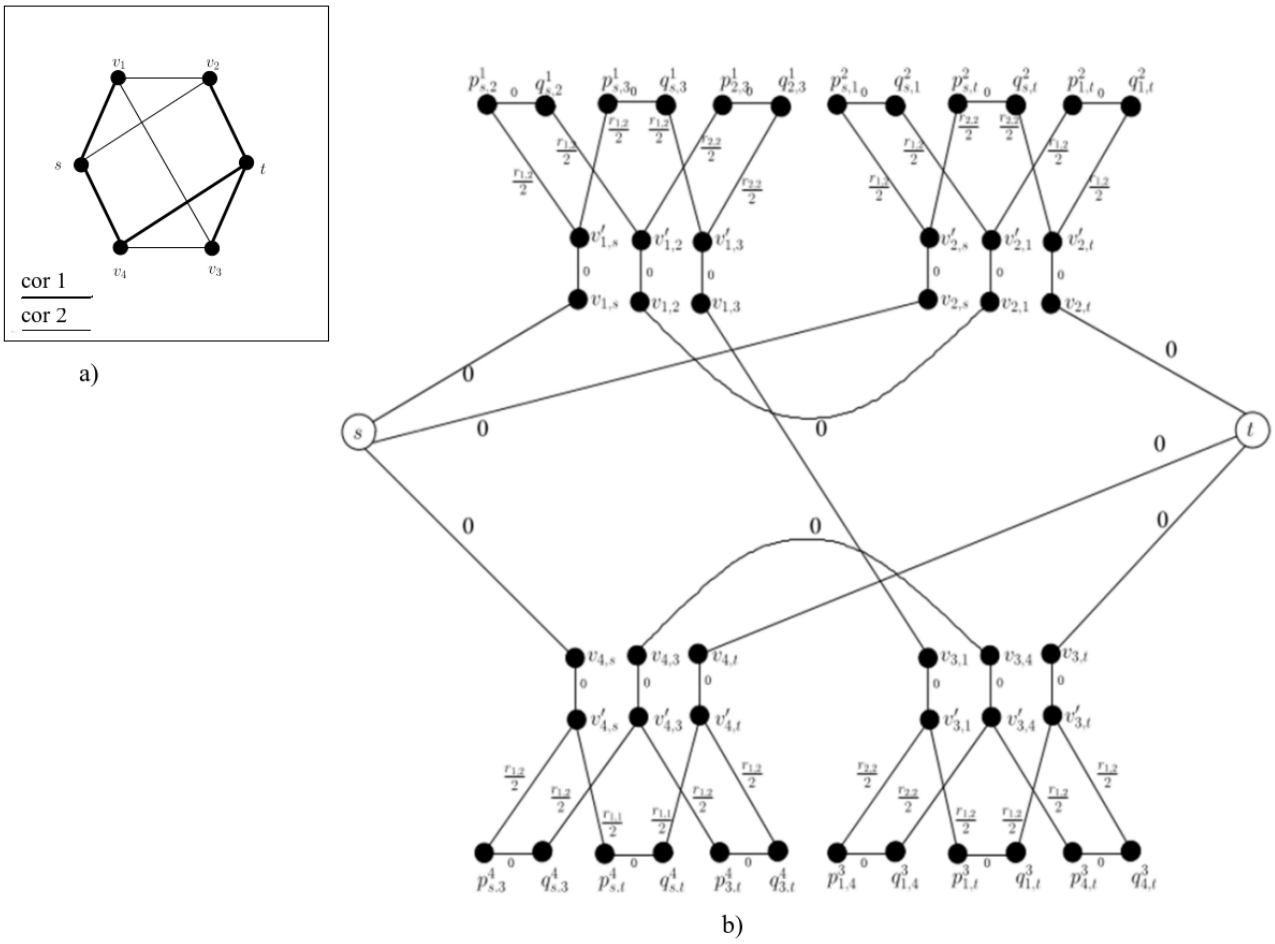

Figura 2.8: Exemplo de um Grafo G^c com 2 cores nas arestas (a) e seu grafo G não-colorido equivalente (b) [17]

2.2.2 Recoloração de arestas

Em Martinhon & Faria [29], problemas de custo de recoloração de arestas são resolvidos para construção de caminhos, trilhas e ciclos propriamente coloridos em um grafo G^c com c cores nas arestas. Dado uma propriedade π, um grafo G^c não satisfazendo a propriedade π e uma matriz de custo de recoloração $R = [r_{i,j}]$ $c \times c$, onde $r_{i,j} \geq 0$ denota o custo da mudança de uma cor i da aresta e para uma cor j, a idéia é trocar as cores de uma ou mais arestas em G^c a fim de obter um novo grafo $G^{c'}_{new}$ com c' cores nas arestas, com $c' \leqslant c$, tal que o custo total de recoloração nas arestas seja minimizado e a propriedade π seja satisfeita. Em [29], esta propriedade se refere especificamente a caminhos/trilhas/ciclos

propriamente coloridos ou monocromáticos em grafos ou digrafos.

Se G^c não contém um $s-t$ caminho propriamente colorido, é proposto o seguinte resultado:

Teorema 8 *(Martinhon & Faria,2013) O problema de determinar o custo mínimo de recoloração nas arestas necessário para construir um $s-t$ caminho propriamente colorido pode ser resolvido em tempo polinomial.*

Dados um grafo G^c, dois vértices $s, t \in V(G^c)$ e uma matriz de custo de recoloração R, o objetivo é construir um grafo G não-colorido com pesos nas arestas onde um emparelhamento perfeito de custo mínimo $M*$ em G (que sempre existirá) determine o valor do custo de recoloração nas arestas mínimo necessário para construir um $s-t$ caminho propriamente colorido em um novo grafo gerado denotado por G^c_{new}.

Para a construção do grafo G são definidos os seguintes passos:

Inicialmente um subgrafo G_x é definido para cada $x \in V(G^c)$ com a seguinte definição do conjunto de vértices e arestas:

$V(G_x) = (\bigcup_{i \in I_c}\{x_i, x'_i\}) \cup \{x''_a, x''_b\}$

$E(G_x) = \{x''_a, x''_b\} \cup (\bigcup_{i \in I_c}(\{x_i x'_i\} \cup (\bigcup_{j=a,b}\{x'_i x''_j\})))$

Onde $w(e) = 0 \forall e \in E(G_x)$, ou seja, é atribuído custo 0 para todas as arestas de $E(G_x)$.

No segundo passo, a construção de G é definida da seguinte forma:

Se xy é uma aresta de $E^j(G^c)$, para cada cor $i \in I_c$, dois novos vértices são inseridos em G e consequentemente são criadas 3 novas arestas: $x_i x_i^y$, $x_i^y y_i^x$ e $y_i^x y_i$ com os pesos: $w(x_i^y y_i^x) = 0$ e $w(x_i x_i^y) = w(y_i^x y_i) = {}^{r_{ji}}/_2$.

Em seguida os subgrafos G_s e G_t são contraídos e substituídos respectivamente, pelos vértices s' e t', como é ilustrado nas Figuras 2.9 e 2.10 a seguir.

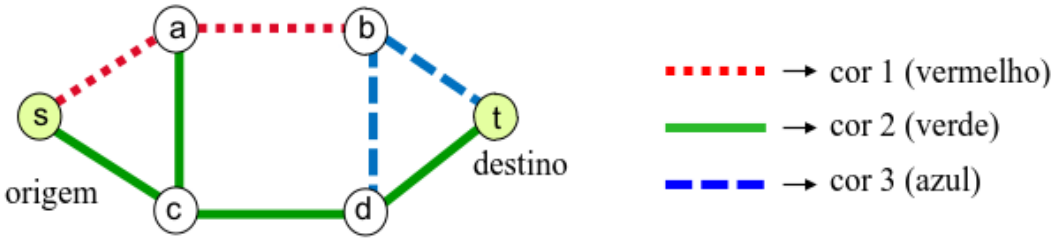

Figura 2.9: Exemplo de um Grafo G^c com 3 cores nas arestas que não contém um $s-t$ caminho propriamente colorido [29]

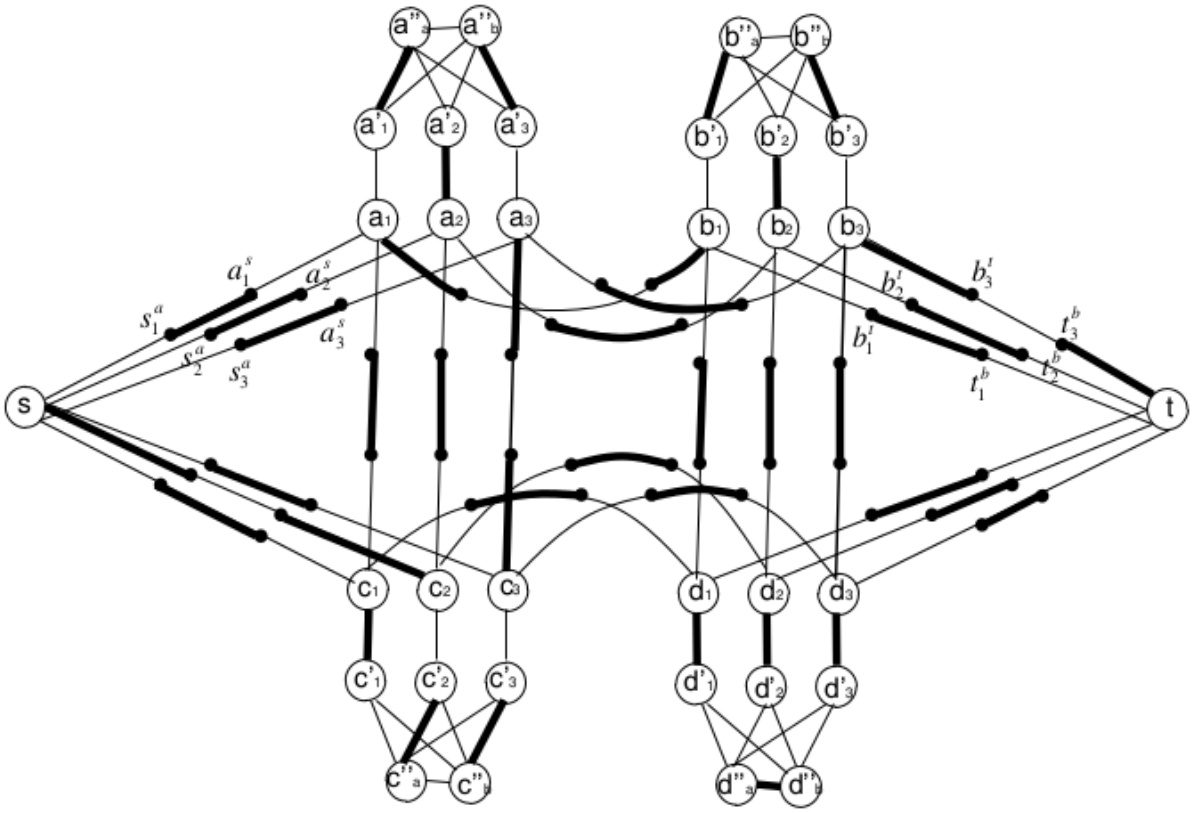

Figura 2.10: Exemplo de um Grafo G com emparelhamento perfeito M^* equivalente ao grafo G^c da Figura 2.9 [29]

A equivalência entre o emparelhamento perfeito em G com o $s-t$ caminho propriamente colorido em G^c mostra que, se M^* é um emparelhamento perfeito de custo mínimo, sempre que algum par de arestas $x_i x_i^y$ e $y_i^x y_i$ (ambas com pesos $r_{ji}/2 > 0$) pertencente ao emparelhamento M^*, a cor associada da aresta $xy \in E^j(G^c)$ é trocada da cor j para a cor i com custo $r_{ji} > 0$ permitindo a construção do grafo G^c_{new} com a nova aresta xy onde $c(xy) = i$. Se $j = i$, a cor da aresta xy não precisa ser trocada. Se as arestas $x_i^y y_i^x$ com

$i \in I_c$ pertencerem ao emparelhamento M^* então a aresta xy não pertence a qualquer $s-t$ caminho propriamente colorido. Se o custo total do emparelhamento M^* for igual a 0 ($w(M^*) = 0$) então G^c já contém um $s-t$ caminho propriamente colorido e não é necessário recolorir nenhuma aresta do grafo.

Na Figura 2.10, o custo total do emparelhamento dado em função da matriz de custos vai determinar (a)s aresta(s) que devem mudar da cor inicial j para a nova cor i. Supondo que $custo(M^*) = r_{23}$, ou seja, todas as arestas de M^* tem custo 0 com exceção das arestas entre os vértices c_3 e a_3 que tem custo igual a $r_{23} > 0$, então conclui-se que a cor da aresta ac em G^c deve ser trocada da cor verde (cor 2) para a cor azul (cor 3).

A complexidade deste procedimento é determinada pela complexidade do problema do emparelhamento perfeito de custo mínimo em G e é igual a $O(N^{2.5})$, onde N é o número total de vértices em G. Uma vez que cada gadget $G_x, \forall x \in V(G^c) \setminus \{s,t\}$ contém $2|I_c|+2$ vértices, a complexidade deste procedimento resume-se a $O(n|I_c|^{2.5})$.

2.2.3 Reconstrução de caminhos ótimos em mapas de ressonância magnética nuclear tridimensional

Em Szachniuk et. al. [33], os autores descrevem o problema da reconstrução de caminhos ótimos em mapas de Ressonância Magnética Nuclear Tridimensional (3D NMR) através da formulação em grafos com cores nas arestas.

O mapeamento 3D NMR está analogamente relacionado ao mapeamento de uma região de cadeia de montanhas onde estamos interessados inicialmente em seus picos, onde cada pico é referenciado por sua latitude, longitude e altitude. O objetivo é traçar a maior rota possível neste mapa onde cada pico seja visitado no máximo uma vez, seguindo repetidamente alguns passos pré-definidos, como por exemplo: *(i)* A partir de um pico p^i vá para um pico p^{i+1} localizado na mesma latitude e longitude. *(ii)* A partir do pico p^{i+1} vá para o pico p^{i+2} localizado na mesma altitude de p^{i+1}.

Experimentos NMR geram uma série de dados coletados na forma espectral. O problema da atribuição de caminhos 3D, analisa os mapas 3D obtidos de moléculas de RNA. Esta informação é um registro das interações NMR que ocorrem entre átomos RNA envolvidos no experimento. Cada interação é representada como um pico cruzado no mapa NMR, onde cada pico cruzado é caracterizado por sua localização no mapa através de suas coordenadas x, y e z representando latitude, longitude e altitude, seu tamanho e o valor de seu sinal de intensidade. Portanto, a atribuição de picos cruzados em mapas 3D NMR

para átomos correspondentes de RNA resulta da construção de caminhos particulares entre esses picos cruzados.

A construção desses caminhos inicia-se a partir de um pico cruzado $p^i(x^i, y^i, z^i)$ no mapa onde x^i, y^i, z^i são as coordenadas de p^i. Dependendo do tipo do caminho, os seguintes passos são repetidos:

1. Para caminhos entre núcleos de mesmo tipo (homonucleares)
   (1) Vá para p^{i+1} se $x^{i+1} = x^i$, $y^{i+1} = y^i$ e $z^{i+1} \neq z^i$
   (2) Vá para p^{i+2} se $x^{i+2} \neq x^{i+1}$, $y^{i+2} = y^{i+1}$ e $z^{i+2} = z^{i+1}$
   (3) Vá para p^{i+3} se $x^{i+3} = x^{i+2}$, $y^{i+3} \neq y^{i+2}$ e $z^{i+3} = z^{i+2}$

2. Para caminhos entre núcleos de tipos diferentes (heteronucleares)
   (1) Vá para p^{i+1} se $x^{i+1} = x^i$, $y^{i+1} \neq y^i$ e $z^{i+1} \neq z^i$
   (2) Vá para p^{i+2} se $x^{i+2} \neq x^{i+1}$, $y^{i+2} \neq y^{i+1}$ e $z^{i+2} = z^{i+1}$ (3) Vá para p^{i+3} se $x^{i+3} \neq x^{i+2}$, $y^{i+3} = y^{i+2}$ e $z^{i+3} \neq z^{i+2}$

O problema pode ser modelado em um grafo $G = (V, E)$ não-direcionado com cores nas arestas. Cada vértice $V^i \in V$ representa um pico cruzado p^i do mapa 3D NMR e $|V| = n$, onde n é o número de picos cruzados. Cada conexão entre dois picos cruzados que podem ter uma ou duas coordenadas em comum é representado por uma aresta $e \in E$ no grafo G, onde m é a quantidade de todas as possíveis conexões do mapa 3D NMR e $|E| = m$. Cada aresta $e(v^i, v^j) \in E$ possui uma cor de um conjunto de $c = 6$ cores $C = \{0, 1, 2, 3, 4, 5\}$ e cada cor representa um tipo de relacionamento entre as coordenadas dos correspondentes picos cruzados. O conjunto E é então particionado em seis subconjuntos a seguir:

$E_0 = \{(v^i, v^j) \in E : x^i \neq x^j, y^i = y^j, z^i = z^j\}$ (1)
$E_1 = \{(v^i, v^j) \in E : x^i = x^j, y^i \neq y^j, z^i = z^j\}$ (2)
$E_2 = \{(v^i, v^j) \in E : x^i = x^j, y^i = y^j, z^i \neq z^j\}$ (3)
$E_3 = \{(v^i, v^j) \in E : x^i = x^j, y^i \neq y^j, z^i \neq z^j\}$ (4)
$E_4 = \{(v^i, v^j) \in E : x^i \neq x^j, y^i \neq y^j, z^i = z^j\}$ (5)
$E_5 = \{(v^i, v^j) \in E : x^i \neq x^j, y^i = y^j, z^i \neq z^j\}$ (6)

A Figura 2.11 ilustra o mapa 3D NMR e seu correspondente grafo G^c com c cores nas arestas representando todas os tipos de relacionamentos entre as coordenadas dos picos cruzados.

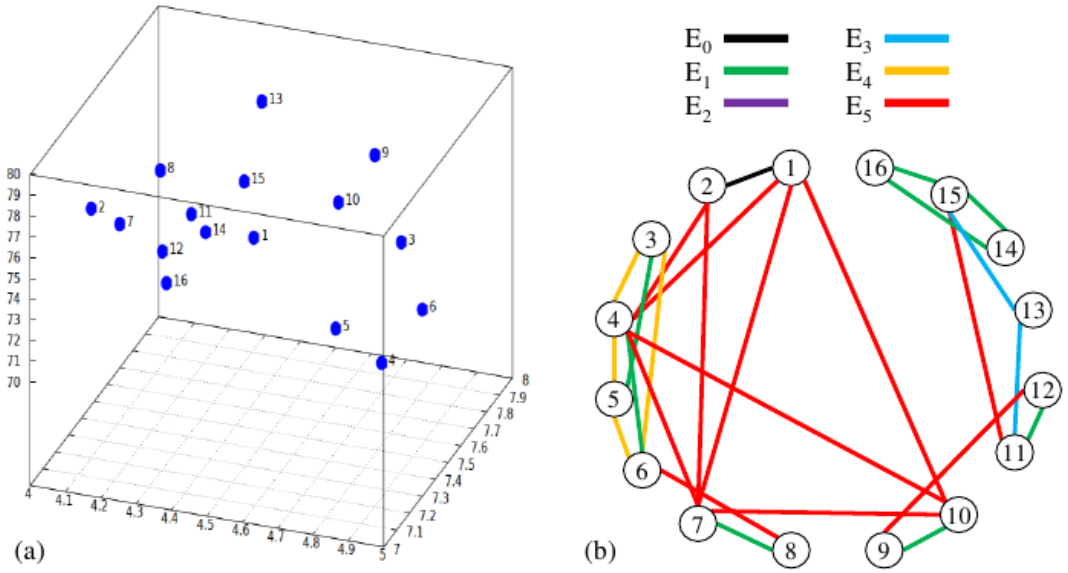

Figura 2.11: Exemplo de um mapa 3DNMR (a) e seu grafo G^c com c cores nas arestas correspondente (b) [33]

Três tipos de mapas 3D NMR (homonuclear, heteronuclear e misto) são definidos e estão representados a seguir por diferentes grafos G^c com c cores nas arestas.

1. Se $G = (V, E)$ representa um mapa homonuclear, $c = 3$, $E = E_0 \cup E_1 \cup E_2$
2. Se $G = (V, E)$ representa um mapa heteronuclear ou misto, $c = 6$, $E = E_0 \cup E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$

A seguir são definidos dois tipos de atribuição de caminhos (homonuclear e heteronuclear) que podem ser reconstruídos em um grafo G^c representando um mapa 3D NMR.

a. Problema de atribuição de caminhos homonucleares (em inglês, homonuclear assignment pathway problem - $3D - APP_{HO}$): Dado um grafo $G^c = (V, E)$ não-direcionado com c cores nas arestas onde V é o conjunto de vértices representando os picos cruzados do mapa 3D NMR e E o conjunto de arestas coloridas particionados em c subconjuntos ($c \in \{3, 6\}$), encontrar o maior caminho elementar $P = \{e_0, e_1, \cdots, e_k\}$ em $E_0 \cup E_1 \cup E_2$ tal que se $e_i \in E_w$ então $e_{i+1}, e_{i+2} \notin E_w$ e as arestas de P estão alternadamente em E_0, E_1, E_2.

b. Problema de atribuição de caminhos heteronucleares (em inglês, heteronuclear assignment pathway problem - $3D-APP_{HE}$): Dado um grafo $G^c = (V, E)$ não-direcionado

com c cores nas arestas onde V é o conjunto de vértices representando os picos cruzados do mapa 3D NMR e E o conjunto de arestas coloridas particionados em $c = 6$ subconjuntos, encontrar o maior caminho elementar $P = \{e_0, e_1, \cdots, e_k\}$ em E tal que se $E_i \in E_w$ então $e_{i+1} \notin E_w$, e as arestas de P estão alternadamente em E_0, E_3 ou $E_1.E_5$ ou E_2, E_4.

Resumindo, o problema da reconstrução de caminhos ótimos em mapas 3D NMR é equivalente ao problema do maior caminho colorido ordenado, onde o caminho é ordenadamente uma sequencia de 3 (caminhos homonucleares) ou 2 (caminhos heteronucleares) cores.

Capítulo 3

Algoritmos Propostos

Neste capítulo, descreveremos as implementações realizadas para determinação de caminhos/ciclos/trilhas/trilhas fechadas propriamente coloridas em um grafo G^c.

Inicialmente, para a resolução de problemas de caminhos e ciclos em grafos com arestas coloridas, analisamos construções **P-Gadgets** existentes na literatura e em seguida descrevemos a implementação da construção **XP-Gadget**, que foi utilizada neste trabalho por ser mais vantajosa em comparação com as outras construções. Descrevemos em seguida, a implementação do menor $s-t$ caminho propriamente colorido em um grafo G^c. Na implementação do algoritmo para determinação da menor $s-t$ trilha propriamente colorida, explicamos em detalhes a transformação de um grafo G^c contendo uma trilha em um grafo H^c associado contendo um caminho propriamente colorido de acordo com o *Teorema 5* explicado na Seção 2.1. Além disto, comparamos a construção P-Gadget utilizada neste algoritmo com a construção utilizada no algoritmo para o menor $s-t$ caminho propriamente colorido.

Para a obtenção do maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c, precisamos garantir a não existência de ciclos/trilhas fechadas propriamente coloridas em G^c, situação onde este problema pode ser resolvido em tempo polinomial. Para isto, descrevemos inicialmente os algoritmos para detecção de ciclos/trilhas fechadas propriamente coloridos, que são construídos em função de algoritmos para detecção de vértices de corte separando cores, vértices monocromáticos e pontes em G^c (veja Seção 2.1).

Para obtenção do maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c qualquer, descrevemos um algoritmo que utiliza um modelo de Programação Linear Inteira(PLI) com restrição de eliminação de subciclos propriamente coloridos. Descreveremos este modelo de PLI e sua implementação no Capítulo 4. No final deste capítulo

descrevemos o formato dos arquivos de entrada utilizados em nossas implementações.

3.1 P-Gadgets

As construções P-Gadgets [20] são ferramentas úteis que permitem transformações de grafos com arestas coloridas em grafos não-coloridos. Elas serão utilizadas na solução de problemas de caminhos e ciclos em grafos com arestas coloridas.

Seja $\chi(x)$ o conjunto de cores das arestas incidentes a um vértice x. Seja G^c um grafo com arestas coloridas e $G' = G^c - \{x \in V(G^c) : |\chi(x)| = 1\}$. Para cada $x \in V(G')$, seja G_x um grafo arbitrário (não-colorido) com as seguintes propriedades:

P1: $\{x_q : q \in \chi(x)\} \subseteq V(G(x))$;

P2: G_x tem um emparelhamento perfeito.

P3: Para cada $p \neq q \in \chi(x)$, se o grafo $G_x - \{x_p, x_q\}$ for não vazio, então este grafo contem um emparelhamento perfeito.

P4: Para cada conjunto $L \subseteq \chi(x)$ com pelo menos 3 elementos; Se o grafo $G_x - \{x_l : l \in L\}$ for não-vazio, então este grafo não contem emparelhamento perfeito.

Cada G_x satisfazendo as propriedades P1-P4 é chamado P-Gadget.

Os três P-Gadgets mais conhecidos na literatura são:

1. **SP-Gadget** [34], onde:

$$V(G_x) = \{x_i, {x'}_i : i \in \chi(x)\} \cup \{{x''}_a, {x''}_b\} \tag{3.1}$$

$$E(G_x) = \{{x''}_a {x''}_b, {x'}_i x''_a, {x'}_i {x''}_b : i \in \chi(x)\} \cup \{x_i x'_i : i \in \chi(x)\} \tag{3.2}$$

2. **BJGP-Gadget** [4], onde:

$$V(G_x) = \{x_j : j \in \chi(x)\} \cup \{y_j : j \in \chi(x) \backslash \{m, M\}\} \tag{3.3}$$

$$\text{onde, } m = min\ \chi(x), M = max\ \chi(x)$$

$$E(G_x) = \{x_j y_k : j \in \chi(x), k \in \chi(x) \backslash \{m, M\}\} \cup \{x_j x_k : j \neq k \in \chi(x)\} \tag{3.4}$$

3. **XP-Gadget** [20], onde:

$$V(G_x) = \{x_j : j \in \chi(x)\} \ \cup \ \{y_j : j \in \chi(x) \backslash \{m, M\}\} \tag{3.5}$$

$$E(G_x) = \{x_m x_M\} \cup \{x_j y_j, x_m y_j, x_M y_j : j \in \chi(x) \setminus \{m, M\} \tag{3.6}$$

Seja $z = \chi(x)$. **SP-Gadget** gera $2z + 2$ vértices e $3z + 1$ arestas, **BJGP-Gadget** gera por sua vez $2z - 2$ vértices e $z(3z - 5)/2$ arestas enquanto que **XP-Gadget** gera $2z - 2$ vértices e $3z - 5$ arestas. Portanto, a transformação **XP-Gadget** é mais vantajosa pois, com o menor numero de vértices e arestas gerados, se torna menos custosa para a implementação do algoritmo.

A construção do grafo G^*, resultante da interligação dos vértices s, t de G^c com a interligação entre todos os subgrafos G_x, se dá a seguir:

Seja G^c um grafo com arestas coloridas e G_x um P-Gadget onde $x \in V(G')$. O Grafo G^* é definido como:

$V(G^*) = \{s, t\} \ \cup \ (\bigcup_{x \in V(G')} V(G_x))$

$E(G^*) = \{\bigcup_{i \in I_c} \{sx_i | sx \in E^i(G^c)\} \cup \{x_i t | xt \in E^i(G^c)\} \cup \{x_i y_i | xy \in E^i(G^c)\}\} \cup \{\bigcup_{x \in V(G')} E(G_x)\}$.

Na Figura 3.1 temos uma ilustração de um vértice x com 3 cores nas arestas e seus G_x correspondentes aos P-Gadgets descritos.

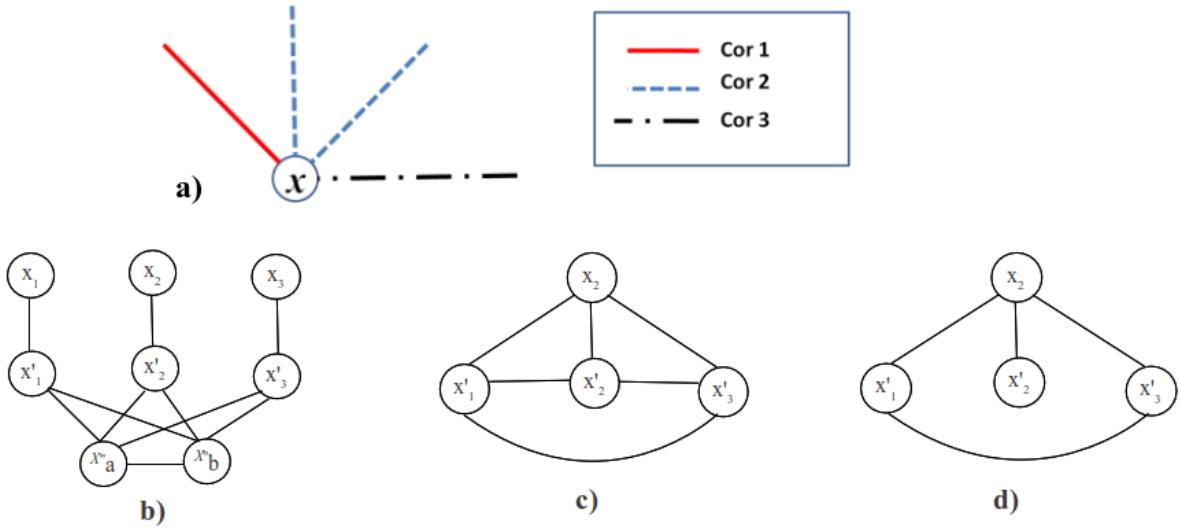

Figura 3.1: a)Exemplo da representação de um vértice x com 3-cores nas arestas; b) Representação G_x utilizando a construção SP-Gadget;c) Representação G_x utilizando a construção BJGP-Gadget; d) Representação G_x utilizando a construção XP-Gadget

Na Figura 3.2, temos um exemplo de um grafo com arestas coloridas para três cores onde serão gerados os grafos baseados nas construções **SP-Gadget** de Szeider [34] e **XP-Gadget** de Gutin et al. [20] ilustrados respectivamente nas Figuras 3.2 e 3.3.

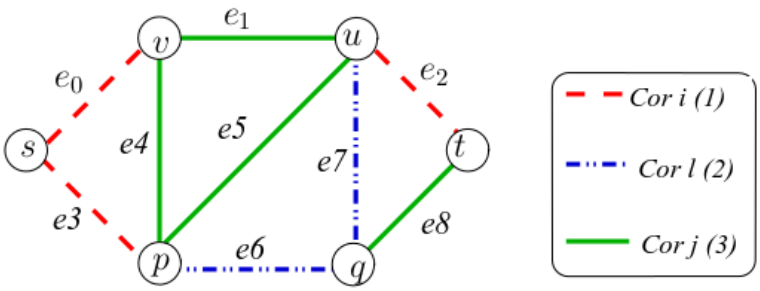

Figura 3.2: Exemplo de um grafo com arestas coloridas com 3 cores.[27]

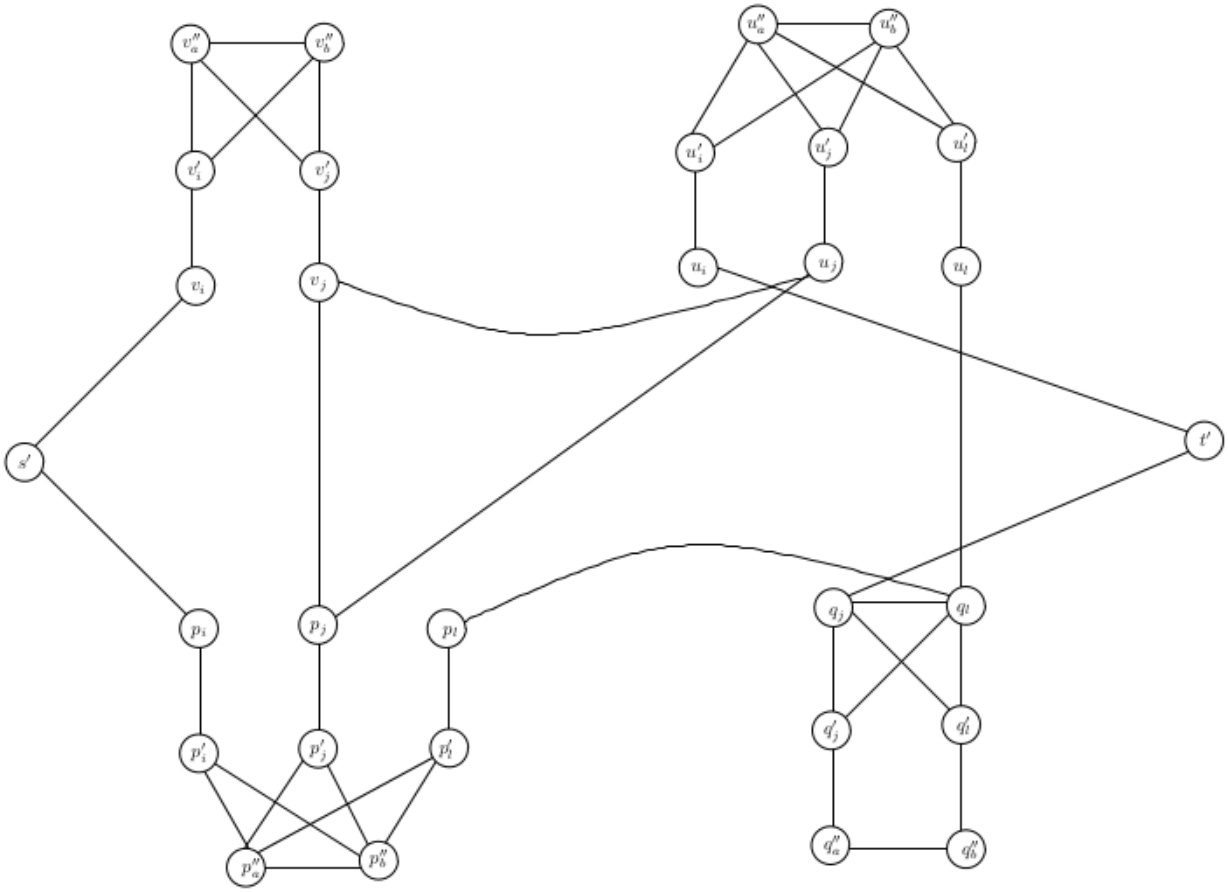

Figura 3.3: Um grafo não-colorido G resultante da construção SP-Gadget no grafo original da Figura 3.2.[27]

Podemos observar visualmente que o grafo gerado pela construção XP-Gadget ilustrado na Figura 3.4 é menor e mais simples que o grafo gerado pela construção SP-Gadget

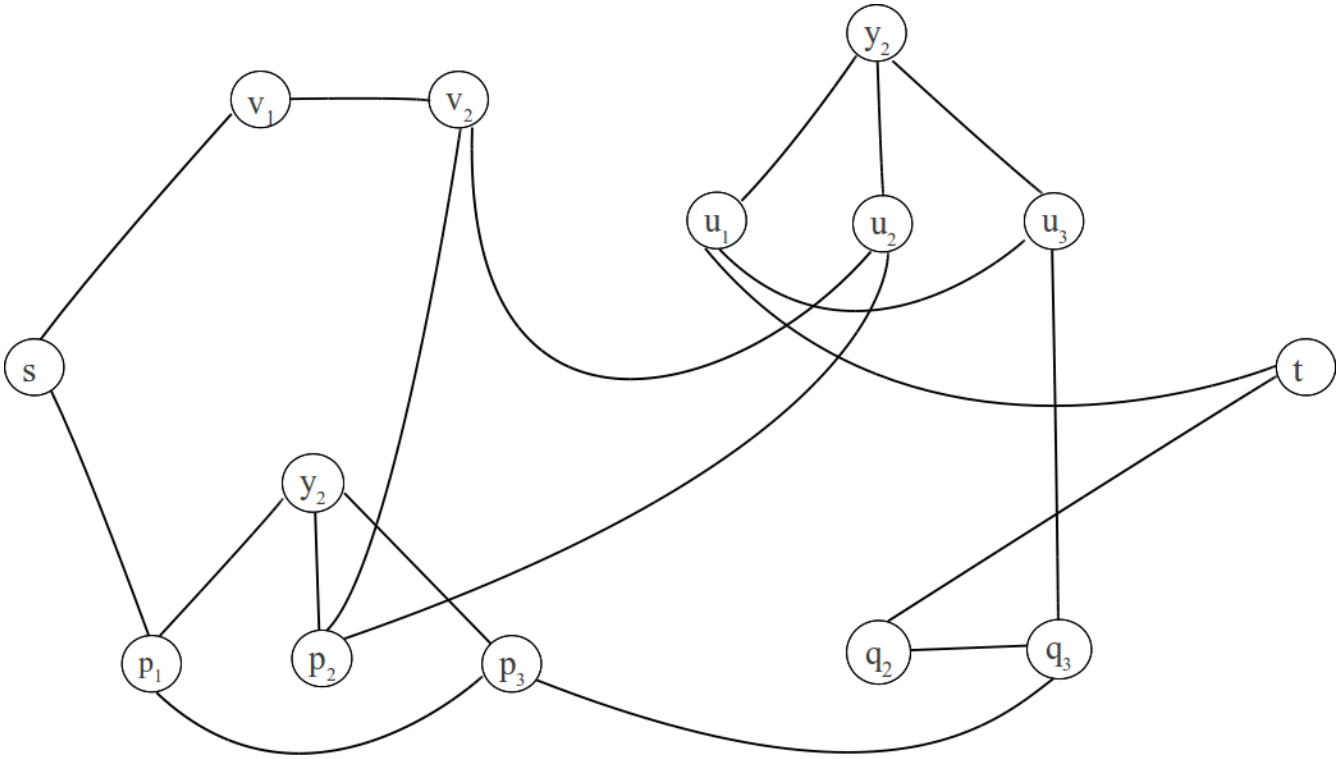

Figura 3.4: Um grafo não-colorido G resultante da construção XP-Gadget no grafo original da Figura 3.2.

ilustrado na Figura 3.3, fato este que pode ser comprovado pelo numero de vértices do conjunto G_x e número de arestas do conjunto E_x de cada grafo gerado.

Estes exemplos gerados servirão de base para a determinação de caminhos $s - t$ que serão discutidos no decorrer deste trabalho.

3.2 Implementação de Grafos com a Construção XP-Gadget

As implementações desenvolvidas neste trabalho fazem uso de um algoritmo de emparelhamento perfeito para obtenção de $s-t$ caminhos, trilhas e ciclos propriamente coloridos. Para isto, é necessária a utilização de algum tipo de construção P-Gadget, uma vez que o algoritmo de emparelhamento tem como entrada um grafo não-colorido. Optamos por desenvolver a construção XP-Gadget pelas razões explicadas na seção anterior.

A implementação da construção XP-Gadget está descrita no **Algoritmo 1**, a seguir:

Algoritmo 1 Obter um Grafo não-colorido G a partir de um Grafo G^c com c arestas coloridas utilizando a construção **XP-Gadget**

Entrada: Um Grafo G^c com c arestas coloridas

Saída: Um Grafo G não-colorido correspondente ao grafo G^c

1: **Inicio**
2: Seja um conjunto: $W = V(G^c) \setminus \{s, t\}$
3: **para todo** $x \in W$ **faça**
4: construa o conjunto $V(G_x)$ conforme a expressão 3.5
5: construa o conjunto $E(G_x)$ conforme a expressão 3.6
6: **fim para**
7: Criar as arestas correspondentes às arestas do conjunto W entre cada G_x criado no grafo G
8: Criar as arestas correspondentes às arestas dos vértices s, t de G^c entre os G_x e os vértices s, t de G.
9: **Fim**

No primeiro passo do **Algoritmo 1**, definimos um conjunto W formado pelos vértices do grafo original G^c, excluindo os vértices s, t. Nas linhas de 3 a 6, geramos os subgrafos G_x formados pelos vértices $V(G_x)$ definido na expressão 3.5 e pelas arestas $E(G_x)$ definido na expressão 3.6. Após a execução das linhas 7 e 8 do algoritmo, a montagem do grafo G está completa e um exemplo deste resultado pode ser visto na Figura 3.4.

3.3 Obtenção do menor $s-t$ caminho propriamente colorido em um Grafo G^c

A implementação para o problema do menor $s-t$ caminho propriamente colorido é baseada no algoritmo proposto por Abouelaoualim et. al. em [1], e está descrita no **Algoritmo 2**.

Algoritmo 2 Obter menor caminho s-t propriamente colorido de um grafo G^c

Entrada: Um grafo G^c com arestas coloridas com vértices $s, t \in V(G^c)$

Saída: Caso exista, o menor caminho s-t propriamente colorido em um Grafo G^c

1: **Inicio**
2: $G \leftarrow$ XP-Gadget(G^c)
3: Defina um conjunto de arestas $E' = \bigcup_{x \in W} E(G_x)$
4: **para todo** $pq \in E(G) \backslash E'$ **faça**
5: $\quad custo(pq) \leftarrow 1$;
6: **fim para**
7: **para todo** $pq \in E'$ **faça**
8: $\quad custo(pq) \leftarrow 0$;
9: **fim para**
10: Se existir, encontre o emparelhamento perfeito de custo mínimo M no Grafo G
11: Contraindo cada gadget G_x ($\forall x \in W$), use M para construir um caminho $s-t$ propriamente colorido P em G^c
12: **Fim**

O primeiro passo do **Algoritmo 2**, definido na linha 2, é chamar a construção **XP-Gadget** (implementada no **Algoritmo 1**) para gerar um grafo não-colorido G a partir de um Grafo de entrada G^c com c arestas coloridas. A seguir, na linha 3, são identificadas todas as arestas pertencentes aos subgrafos G_x para separar estas arestas das arestas que interligam os subgrafos G_x.

Nas linhas de 4 a 9 do **Algoritmo 2**, a idéia foi penalizar todas as arestas de G associadas com as arestas do grafo original G^c. Desta forma o algoritmo de emparelhamento perfeito irá minimizar o número de arestas de G associadas a arestas coloridas em G^c. Uma vez que as arestas de $E(G_x)$ estão com os pesos com valor zero, o caminho P resultante será formado pelas arestas pertencentes ao emparelhamento perfeito M de $E(G) \backslash E'$. Estas arestas definirão o $s-t$ caminho associado a um $s-t$ caminho propriamente colorido no grafo original G^c.

Na linha 10 do algoritmo, utilizamos algoritmo de emparelhamento perfeito de custo mínimo denominado "Blossom V" proposto em [25]. O conjunto M é gerado do conjunto de arestas do emparelhamento perfeito gerado, excluindo as arestas de custo zero. Por fim, para obtermos o passo descrito na linha 11, contraímos cada conjunto de vértices de G_x de maneira que cada G_x seja representado por um vértice apenas. Estes novos vértices são interligados às arestas do conjunto M modificando o grafo G anteriormente gerado. Desta forma, conseguimos associar este grafo G com o grafo original G^c para obtermos o menor $s-t$ caminho propriamente colorido no grafo original G^c.

A Figura 3.5 representa o resultado da aplicação do algoritmo de emparelhamento perfeito "Blossom V" [25] no grafo não-colorido G ilustrado na Figura 3.4. As arestas pontilhadas representam as arestas do emparelhamento perfeito M. O conjunto de arestas de $G_x : \{v_1v_2, y_2p_3, y_2u_3, q_2q_3\}$ possuem peso zero (linha 8 do algoritmo) e consequentemente estão fora do cálculo do custo mínimo total. O caminho ρ em G^c associado com M (linha 11 do algoritmo) é: $\rho = (s, e_3, p, e_5, u, e_2, t)$ e está ilustrado na Figura 3.6. O resultado ilustrado na Figura 3.6 representa o menor $s-t$ caminho propriamente colorido associado ao grafo original G^c ilustrado na Figura 3.2.

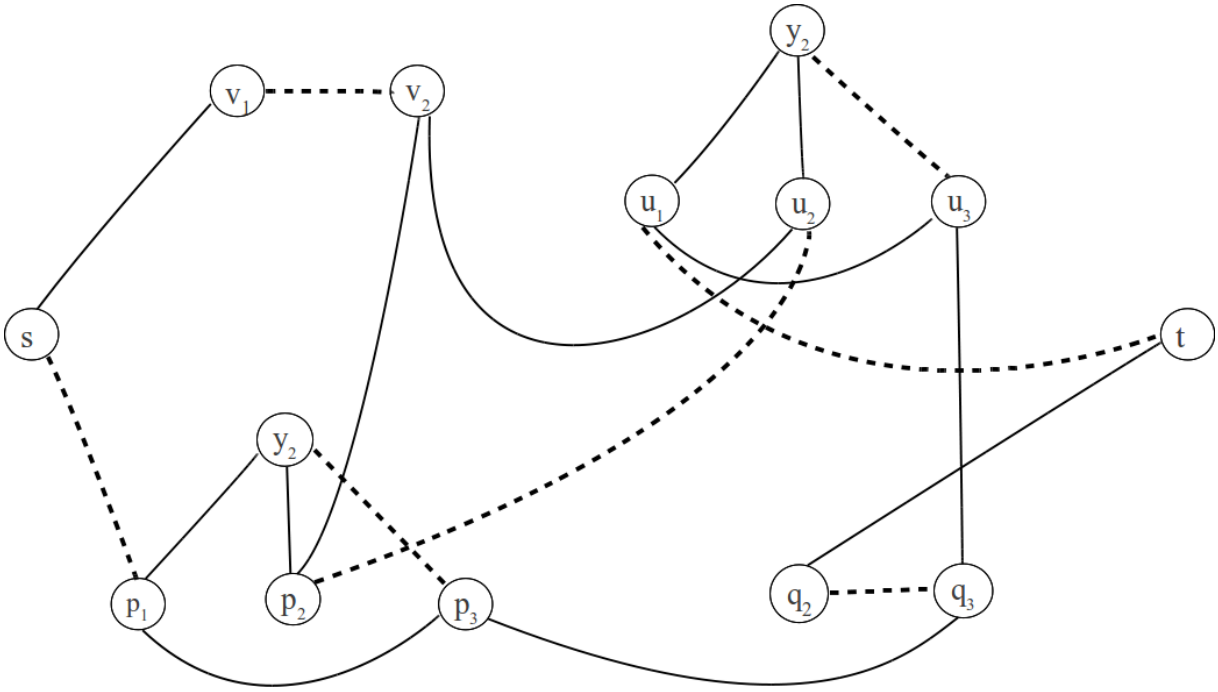

Figura 3.5: Um emparelhamento perfeito em G

Notamos que nem todos os vértices $x \in V(G^c)$ fazem parte do caminho propriamente colorido ρ. Se existir um emparelhamento perfeito M' entre os vértices de um conjunto $V(G_x)$ no grafo G, então o vértice x em G^c não faz parte do caminho propriamente colorido ρ em G^c.

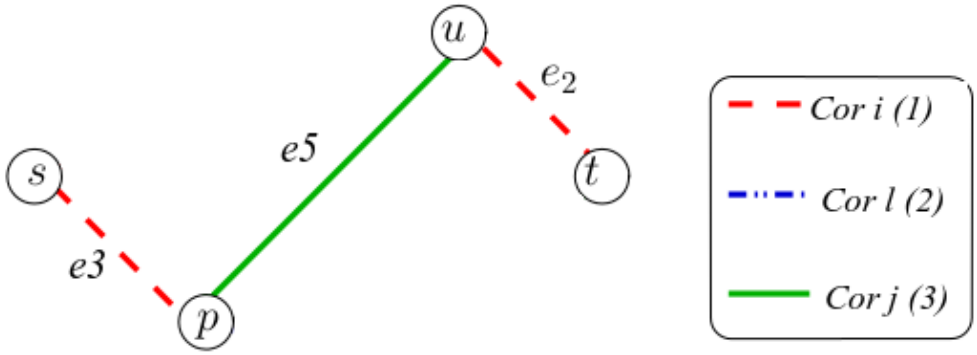

Figura 3.6: Menor $s-t$ caminho propriamente colorido obtido do grafo original G^c da Figura 3.2

3.4 Obtenção da menor $s-t$ trilha propriamente colorida em um grafo G^c

Em Abouelaoualim et al. [1], os autores provaram que encontrar trilhas propriamente coloridas em G^c é equivalente a encontrar caminhos propriamente coloridos em um grafo trilha-caminho $p-H^c$ (onde $p \geqslant 2$).

Como consequência do *Teorema 5* (veja Seção 2.1), para encontramos uma trilha propriamente colorida, basta aplicarmos uma construção P-Gadget (XP-Gadget, por exemplo) no grafo H^c que não leve em consideração os vértices v_{xy} e u_{xy} e aplicarmos o algoritmo de emparelhamento perfeito (Blossom V) no grafo não-colorido resultante desta construção P-Gadget. A implementação dessa construção P-Gadget difere um pouco da implementação descrita no **Algoritmo 1** no que se refere ao conjunto dos vértices de entrada no grafo e consequentemente na quantidade de vértices gerados no grafo resultante. Neste caso, o conjunto de entrada seria: $W = V(G_c) \setminus \{s,t,v_{xy},u_{xy}\} \forall xy \in E(G^c)$. A quantidade de vértices resultantes seria dada pela fórmula:

$\sum_{i=1}^{n-2}(2z_{x_i}-2)+\sum v_{xy}+\sum u_{xy}+2$, onde $x = V(G) \setminus \{s,t,v_{xy},u_{xy}\}$ e $z_x = \chi(x)$.

De acordo com a formula, o conjunto de cores das arestas incidentes aos vértices v_{xy} e u_{xy}, denotados respectivamente por $z_{v_{xy}}$ e $z_{u_{xy}}$ não são calculados fazendo com que o grafo resultante tenha uma economia de 2 vértices para cada par (u_{xy},v_{xy}) gerado. A solução deste problema foi implementada em tempo polinomial e como pré-requisito foi criado um algoritmo para gerar um grafo H^c correspondente ao grafo original G^c. A seguir, temos o algoritmo completo:

A implementação da menor $s-t$ trilha propriamente colorida em G^c (baseada no *Teorema 5* para $p=2$) está descrita no **Algoritmo 3**.

Algoritmo 3 Obter, caso exista, a menor $s-t$ trilha propriamente colorida em um grafo G^c

---

Entrada: Um grafo G^c com c arestas coloridas

Saída: Caso exista, a menor $s-t$ trilha propriamente colorida em G^c

1: **Inicio**
2: Gere o grafo trilha-caminho H^c conforme descrito na Seção 2.1
3: Aplique o **Algoritmo 2** no grafo H^c
4: **se** H^c contem um caminho $s-t$ propriamente colorido P **então**
5: Use P para construir uma $s-t$ trilha propriamente colorida correspondente em G^c
6: **fim se**
7: **Fim**

Percebemos que o **Algoritmo 3** nada mais é do que uma reutilização do **Algoritmo 2** definido anteriormente. Ou seja, para obtermos uma $s-t$ trilha propriamente colorida é necessário que se tenha um $s-t$ caminho propriamente colorido, proposto no **Algoritmo 2**, aplicado em um grafo H^c contendo um $s-t$ caminho correspondente a uma trilha no grafo G^c com c arestas coloridas. A Figura 3.7, mostra o resultado da equivalência de um $s-t$ caminho propriamente colorido e uma $s-t$ trilha propriamente colorida. A equivalência, proposta na linha 5 do algoritmo, se dá através do processo inverso ao realizado para geração do grafo trilha-caminho $p-H^c$.

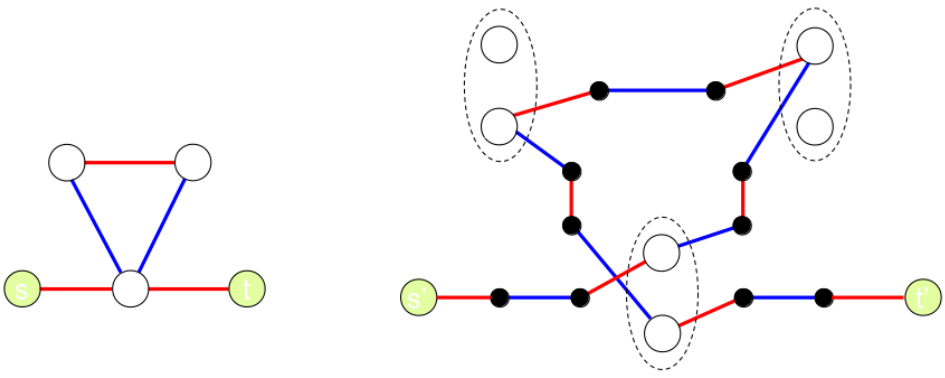

Figura 3.7: Equivalencia entre um caminho s-t e uma trilha s-t propriamente coloridos

3.5 Verificação de um Ciclo/Trilha Fechada Propriamente Colorido(a) em um Grafo G^c

Vários problemas importantes podem ser resolvidos em tempo polinomial caso G^c não contenha ciclos propriamente coloridos. Em Abouelaoualim et al. [1], os autores apresentaram procedimentos em tempo polinomial para o problema do maior $s-t$ caminho (resp. trilha) para grafos não contendo ciclos (resp. trilhas fechadas) propriamente coloridos.

Em Gourvès et al.[18], os autores provaram que se G^c não contém trilhas fechadas propriamente coloridas, o problema de encontrar uma $s-t$ trilha propriamente colorida visitando um determinado subconjunto de vértices pode ser resolvido em tempo polinomial (este problema se torna NP-Completo se for restrito a grafos não contendo ciclos propriamente coloridos [7]).

O *Teorema 4*, aborda o conceito de *vértice de corte separando cores*, onde um vértice $v \in G^c$ é um *vértice de corte separando cores* , se e somente se, nenhum componente de $G^c - v$ é ligado a v por pelo menos duas arestas de cores diferentes. Em outras palavras, cada componente conexa de $G^c - v$ só pode estar ligado a v por arestas de uma cor apenas.

Como consequência do *Teorema 4*, podemos verificar a existência de um ciclo propriamente colorido em tempo polinomial proposta em [27]. Para isso, devemos excluir todos os *vértices de corte separando cores* do Grafo G^c. Se o conjunto E de arestas resultante for não-vazio, então o grafo G^c contém um ciclo propriamente colorido. A implementação da solução deste problema foi dividida em duas partes (algoritmos). Na primeira parte, descreveremos um algoritmo para a determinação de um *vértice de corte separando cores*.

O **Algoritmo** 4 faz uma varredura em todos os vértices do grafo com o objetivo de retornar um *vértice de corte separando cores* em G^c. Na implementação do **Algoritmo 4** precisamos criar um grafo auxiliar G' formado pelo grafo original G retirado o vértice v e suas arestas incidentes (linha 3). A seguir (linha 4), calculamos o número de componentes conexas do grafo G' resultante. Este cálculo foi realizado através da chamada de uma rotina adaptada de Sedgewick em [32] de complexidade $O(n+m)$, onde n é o numero de vértices e m é o número de arestas do grafo. Na linha 5, são definidos os k conjuntos de vértices correspondentes às k componentes conexas de G'. Para os k conjuntos de vértices W_j, *onde* $j \in \{1..k\}$ referentes às k componentes conexas de G' definimos $\chi(W_j)$ como o conjunto das cores das arestas dos vértices de cada conjunto W_j incidentes ao vértice v (linha 6). A definição de $\chi(W_j)$ foi necessária para obtermos a quantidade de cores de cada componente conexa ligada ao vértice v e para que v seja um vértice de corte separando

Algoritmo 4 Determinação de um vértice de corte separando cores em um grafo G^c, caso exista.

Entrada: Um grafo G^c
Saída: Um vértice de corte separando cores, caso exista.

```
1:  Inicio
2:  para todo v ∈ G^c faça
3:     Construir um grafo auxiliar G' = G^c \ v
4:     Determinar k como o numero de componentes conexas de G'
5:     Construir os conjuntos de vértices W_j(G') ∀j ∈ {1..k} das k componentes conexas
       de G'
6:     Definir χ(W_j) como o conjunto das cores das arestas dos vértices de W_j ligados ao
       vértice v
7:     VCSP ← VERDADEIRO;
8:     para todo W_j, onde j ∈ {1..k} faça
9:        se |χ(W_j)| > 1 então
10:          VCSP ← FALSO
11:       fim se
12:    fim para
13:    se VCSP então
14:       Retorne v
15:    fim se
16: fim para
17: Fim
```

cores, todas as componentes conexas de $G^c \setminus v$ tem que estar ligadas a v por uma mesma cor (linhas: 8 a 12). Em outras palavras, temos um *vértice de corte separando cores* se os vértices adjacentes a v incidentes nas arestas de mesma cor pertencerem a uma mesma componente conexa.

Na segunda parte (**Algoritmo** 5), utilizaremos o **Algoritmo** 4, onde a cada iteração, um novo grafo G^c com um vértice a menos é gerado caso o exista um *vértice de corte separando cores* no Grafo G^c anterior. Ao final saberemos se o grafo G^c original possui ou não, um ciclo propriamente colorido.

Algoritmo 5 Determinação de um ciclo propriamente colorido em um grafo G^c, caso exista.

Entrada: Um grafo G^c

Saída: Ciclo propriamente colorido em um grafo G^c, caso exista.

1: **Início**
2: **repita**
3: Aplique o **Algoritmo** 4 no grafo G^c
4: **se** $\exists\ v$ **então**
5: $G^c = G^c \setminus v$
6: **fim se**
7: **até** $\nexists\ v \in G^c$ onde v seja um *vértice de cor separando cores*
8: **se** G^c for não-vazio **então**
9: O grafo G^c contém um ciclo propriamente colorido
10: **senão**
11: O grafo G^c não contém um ciclo propriamente colorido
12: **fim se**
13: **Fim**

Para o caso da obtenção de uma trilha fechada propriamente colorida, citamos o *Teorema 6* proposto em [1] (veja Seção 2.1).

Com base no *Teorema 6* , foram desenvolvidos dois algoritmos: um para detectar vértices monocromáticos no grafo G^c e um outro (adaptado do **Algoritmo** 4) para detectar uma ponte ou aresta de corte em um grafo G^c.

No Algoritmo para detecção de vértices monocromáticos, é feita uma varredura em todos os vértices do grafo G^c com o objetivo de (em caso de sucesso) retornar o primeiro vértice monocromático encontrado. Para isto, é definido o conjunto de cores $\chi(v)$ da vizinhança do vértice v do seu conjunto de arestas $E(v)$. Se este conjunto retornar apenas um elemento, todas as arestas incidentes a este vértice possuem a mesma cor, significando que este vértice v é um vértice monocromático.

Apesar de todo vértice monocromático ser um *vértice de corte separando cores* não podemos utilizar diretamente o Algoritmo para detecção de vértices monocromáticos na retirada de vértices de corte. Um exemplo claro disso está ilustrado na Figura 2.2 da Seção 2.1, onde pela definição, v e w são dois vértices de corte separando cores. Se aplicarmos o Algoritmo para detecção de vértices monocromáticos, apenas o vértice w seria retirado do grafo G^c.

No Algoritmo para detecção de pontes, percorremos todas as arestas de G^c e calculamos para cada aresta e, o número de componentes conexas do grafo G^c e do grafo $\overline{G}^c$ (formado pelo grafo G^c sem a aresta e). Se o número de componentes conexas do grafo original $nc(G^c)$ for menor que o número de componentes conexas do grafo G^c sem a aresta e ($nc(\overline{G}^c)$), então e é uma ponte ou aresta de corte no grafo G^c.

Descreveremos no **Algoritmo 6** o algoritmo completo para detecção de uma trilha fechada propriamente colorida em G^c

Algoritmo 6 Determinação de uma trilha fechada propriamente colorida em um grafo G^c, caso exista.

Entrada: Um grafo G^c

Saída: Uma trilha fechada propriamente colorida em G^c, caso exista.

1: **Início**
2: $loop \leftarrow VERDADEIRO$
3: **enquanto** *loop* **faça**
4: **se** $\exists$ vértice monocromatico v obtido de G^c **então**
5: $G^c = G^c \setminus v$
6: **senão se** $\exists$ ponte e obtida de G^c **então**
7: $G^c = G^c \setminus \{e\}$
8: **senão**
9: $loop \leftarrow FALSO$
10: **fim se**
11: **fim enquanto**
12: **se** $E(G^c) \neq \varnothing$ **então**
13: O Grafo G^c contem uma trilha fechada propriamente colorida
14: **fim se**
15: **Fim**

Na implementação do **Algoritmo 6**, temos um *loop* onde o grafo original G^c vai sendo diminuído de um vértice v ou uma aresta e cada vez que é encontrado um vértice de corte separando cores ou uma ponte, respectivamente. Ou seja, quando não houverem mais vértices monocromáticos ou pontes no grafo G e o conjunto de arestas do grafo ao final do *loop* for não-vazio então o grafo original G^c contém uma trilha fechada propriamente colorida.

3.6 Obtenção do maior $s-t$ caminho/trilha prop. colorido em G^c sem ciclos/trilhas fechadas prop. coloridas

Encontrar o maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c arbitrário é um problema NP-Difícil. No entanto, para grafos sem ciclos/trilhas fechadas propriamente coloridos, Abouelaoualim et. al. [1] propos um procedimento em tempo polinomial, baseado em emparelhamento perfeito de custo máximo, para encontrar o maior $s-t$ caminho/trilha fechada propriamente colorido.

Iniciaremos esta seção apresentando o seguinte Teorema provado por Abouelaoualim [1]:

Teorema 9 *(Abouelaoualim et al., 2008) Assuma que um grafo G^c não tenha um ciclo propriamente colorido. Portanto podemos sempre encontrar em tempo polinomial, o maior $s-t$ caminho propriamente colorido ou este caminho não existe em G^c.*

Seguindo o *Teorema 9*, esta implementação primeiramente verifica através da utilização do **Algoritmo 5** se o Grafo G^c não possui um ciclo propriamente colorido. O restante da implementação foi adaptada do **Algoritmo 2** (obtenção do menor s-t caminho propriamente colorido em um Grafo G^c) e diferencia-se basicamente na penalização dos pesos das arestas geradas do conjunto $E(G)\backslash E'$ (veja a definição de G e E' no **Algoritmo 2** definido na Seção 3.3). Atribuímos o valor -1 para o conjunto de arestas não pertencentes aos subgrafos G_x e desta forma, utilizando o mesmo algoritmo de emparelhamento perfeito de custo mínimo, produzimos o efeito contrário, ou seja, maximizamos o numero de arestas de $E(G)\backslash E'$ uma vez que os pesos das arestas $E(G) \setminus E'$ são negativos e G^c não contém ciclos de custo negativo.

Uma vez que as arestas de $E(G)\backslash E'$ estão com os pesos com valor negativo, o caminho P resultante será formado pelas arestas pertencentes ao emparelhamento perfeito M de $E(G)\backslash E'$. Estas arestas definirão o caminho $s-t$ associado a um caminho propriamente colorido $s-t$ no grafo original G^c.

Algoritmo 7 Obter maior $s-t$ caminho propriamente colorido de um grafo G^c sem ciclos propriamente coloridos

Entrada: Um grafo G^c com arestas coloridas com vértices $s, t \in V(G^c)$

Saída: Caso exista, o maior caminho s-t propriamente colorido em um Grafo G^c

1: **Inicio**
2: **se** $\nexists$ um ciclo propriamente colorido através da execução do **Algoritmo 5** no grafo G^c **então**
3: Aplique o algoritmo da construção XP-Gadget (**Algoritmo 1**) no grafo G^c
4: Defina um conjunto de arestas $E' = \bigcup_{x \in W} E(G_x)$
5: **para todo** $pq \in E(G) \backslash E'$ **faça**
6: $custo(pq) \leftarrow -1$;
7: **fim para**
8: **para todo** $pq \in E'$ **faça**
9: $custo(pq) \leftarrow 0$;
10: **fim para**
11: Se existir, encontre o emparelhamento perfeito de custo mínimo M no Grafo G
12: Use M para construir um caminho $s-t$ propriamente colorido P em G^c
13: **fim se**
14: **Fim**

As figuras 3.8, 3.9 e 3.10 ilustram respectivamente, um exemplo de um grafo G^c sem ciclos propriamente coloridos, o maior $s-t$ caminho encontrado e o grafo G não-colorido (resultante da aplicação do gadget **XP-Gadget** em G^c) com um emparelhamento perfeito de custo máximo.

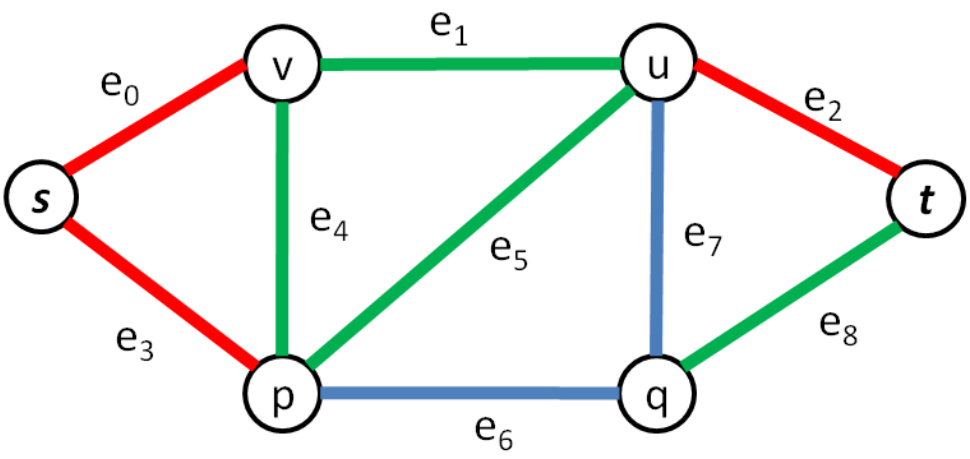

Figura 3.8: Um grafo G^c sem ciclos propriamente coloridos

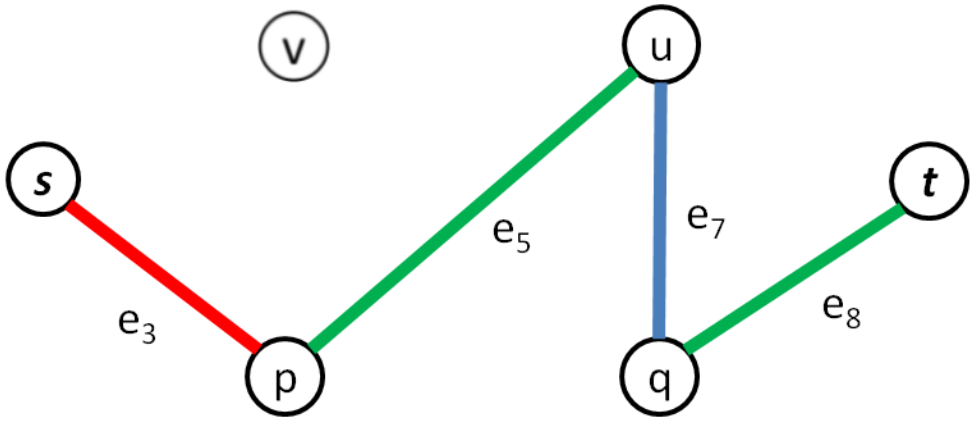

Figura 3.9: Maior $s-t$ caminho propriamente colorido em G^c

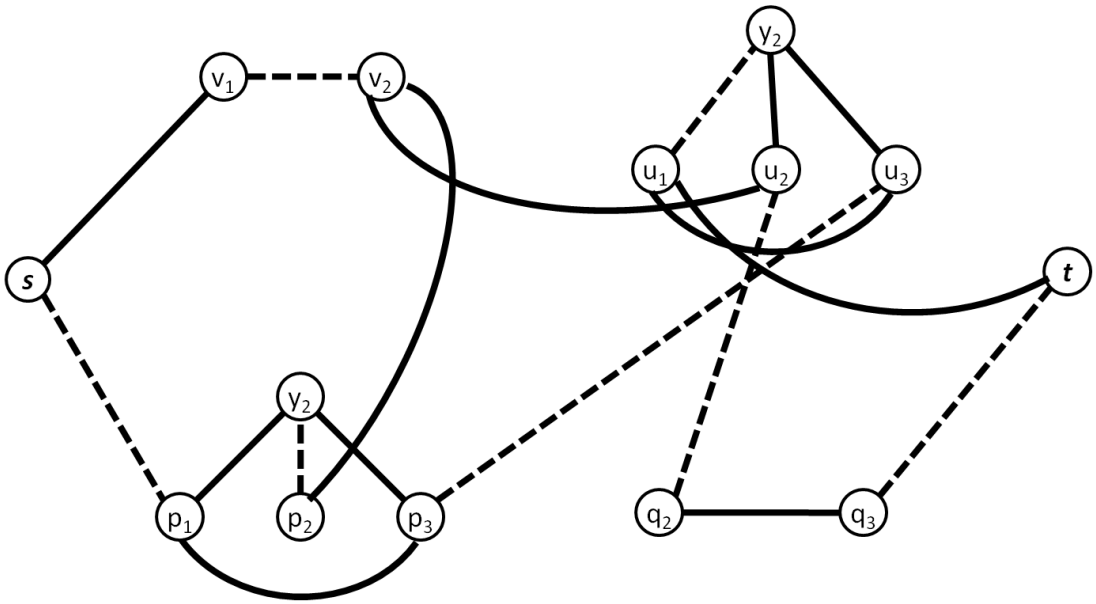

Figura 3.10: Grafo G não-colorido com emparelhamento perfeito de custo máximo

Para o caso da obtenção da maior $s-t$ trilha propriamente colorida, apresentamos abaixo o seguinte Teorema proposto por Abouelaoualim et. al. em [1].

Teorema 10 *(Abouelaoualim et al., 2008) Seja G^c um grafo com c arestas coloridas e que não possua uma trilha fechada propriamente colorida e dois vértices $s,t \in V(G^c)$. Portanto, podemos sempre encontrar em tempo polinomial a maior $s-t$ trilha propriamente colorida em G^c, caso exista.*

Na implementação do **Algoritmo 8**, Seguimos a definição do *Teorema 10* e utilizamos o **Algoritmo 6** para verificarmos se o grafo G^c não contem uma trilha fechada

propriamente colorida. Aplicando o *Teorema 5*, onde podemos gerar um $s-t$ caminho no grafo trilha-caminho $p-H^c$ correspondente. Em seguida, aplicamos o **Algoritmo 7** para encontrarmos o maior $s-t$ caminho propriamente colorido P e utilizamos este grafo resultante para gerarmos a maior $s-t$ trilha propriamente colorida T correspondente ao caminho gerado.

Algoritmo 8 Obter a maior $s-t$ trilha propriamente colorida de um grafo G^c sem trilhas fechadas prop. coloridas

Entrada: Um grafo G^c com arestas coloridas com vértices $s,t \in V(G^c)$
Saída: Caso exista, a maior $s-t$ trilha propriamente colorida em um Grafo G^c

1: **Inicio**
2: **se** $\nexists$ uma trilha fechada propriamente colorida através da execução do **Algoritmo 6** no grafo G^c **então**
3: Gere o grafo trilha-caminho $p-H^c$ conforme descrito na Seção 2.1
4: No grafo $p-H^c$, Aplique o algoritmo **Algoritmo 7** para obter o maior $s-t$ caminho propriamente colorido P
5: Use P para construir a maior $s-t$ trilha propriamente colorida T em G^c
6: **fim se**
7: **Fim**

3.7 Obtenção do maior $s-t$ caminho/trilha propriamente colorido(a) em um grafo G^c qualquer

A solução para o maior $s-t$ caminho/trilha propriamente colorido(a) em um grafo G^c qualquer é um problema NP-Difícil uma vez que este problema generaliza o problema do caminho hamiltoniano em grafos não-coloridos [1]. Descreveremos nesta seção a implementação de um pré-processamento em cima do grafo G^c original de forma a tratarmos todas as possibilidades da não-existência de $s-t$ caminhos propriamente coloridos e da existência de ciclos propriamente coloridos antes de resolvê-lo diretamente através de um modelo de programação linear inteira.

Na primeira parte do **Algoritmo 9**, procuramos simplificar o grafo original G^c retirando todos os vértices $v \in V(G^c)\{s,t\}$ monocromáticos. O Grafo resultante deste processo torna a sua construção XP-Gadget menos custosa uma vez que o grafo G^c contem apenas vértices cujo conjunto de cores de sua vizinhança seja maior ou igual a dois, ou seja, $\chi(v) \geqslant 2$. Em seguida, aplicamos um algoritmo de emparelhamento perfeito de custo mínimo para verificar se há um $s-t$ caminho propriamente colorido em G^c. Em caso afirmativo, chegamos a uma solução G^c_{sol} que pode ser a solução ótima caso o grafo resultante não contenha ciclos propriamente coloridos. Isto significa que não existe um

Algoritmo 9 Obter o maior $s-t$ caminho propriamente colorido de um grafo G^c qualquer

Entrada: Um grafo G^c com c arestas coloridas e vértices $s, t \in V(G^c)$
Saída: Caso exista, o maior $s-t$ caminho propriamente colorido em um Grafo G^c

Inicio
enquanto $\exists$ v monocromatico $\in G^c$, onde $v \notin \{s,t\}$ **faça**
  $G^c = G^c \setminus \{v\}$
fim enquanto
Aplique o algoritmo da construção XP-Gadget (**Algoritmo 1**) no Grafo G^c
Defina um conjunto de arestas $E' = \bigcup_{x \in W} E(G_x)$
para todo $pq \in E(G) \setminus E'$ **faça**
  $custo(pq) \leftarrow -1$;
fim para
para todo $pq \in E'$ **faça**
  $custo(pq) \leftarrow 0$;
fim para
se $\nexists$ emparelhamento perfeito de custo mínimo no grafo G **então**
  Não existe um $s-t$ caminho propriamente colorido em G^c
senão
  Seja G^c_{sol} o grafo solução com um $s-t$ caminho do grafo G^c
  se G^c_{sol} não contém ciclos propriamente coloridos **então**
    Retorne G^c_{sol}
  senão
    Aplique um modelo de PLI em G^c
  fim se
fim se
Fim

ciclo propriamente colorido na solução G^c_{sol} e portanto esta contém o maior $s-t$ caminho propriamente colorido.

Caso G^c_{sol} contenha ciclos propriamente coloridos então esta solução não é ótima. Para obtermos a solução ótima, precisamos aplicar um modelo de PLI que contenha restrições para eliminação de possíveis subciclos encontrados da solução para que encontremos a solução ótima, ou seja, o maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer.

As figuras 3.11 e 3.12 ilustram um exemplo de um grafo G^c qualquer (caso geral) onde a solução inicial encontrada contém sub-ciclos propriamente coloridos e neste caso o problema é resolvido com o modelo de PLI descrito no Capítulo 4.

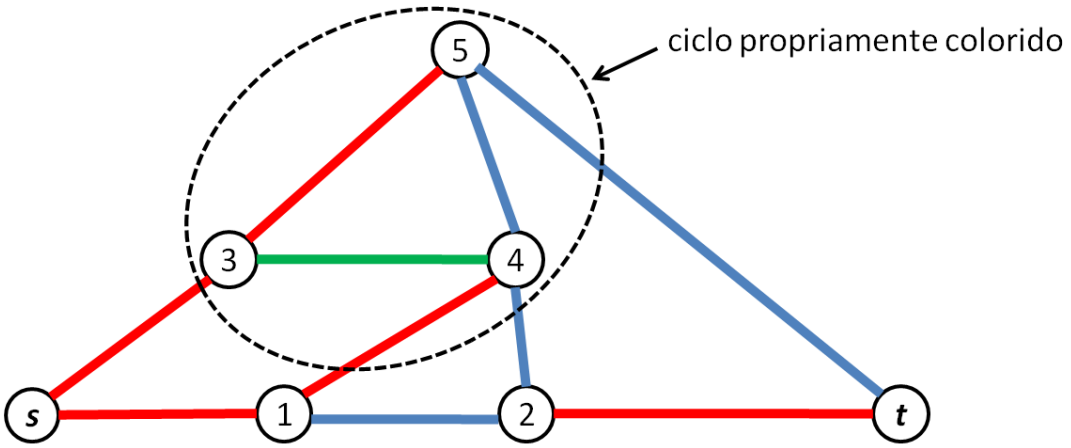

Figura 3.11: Grafo G^c contendo um ciclo propriamente colorido

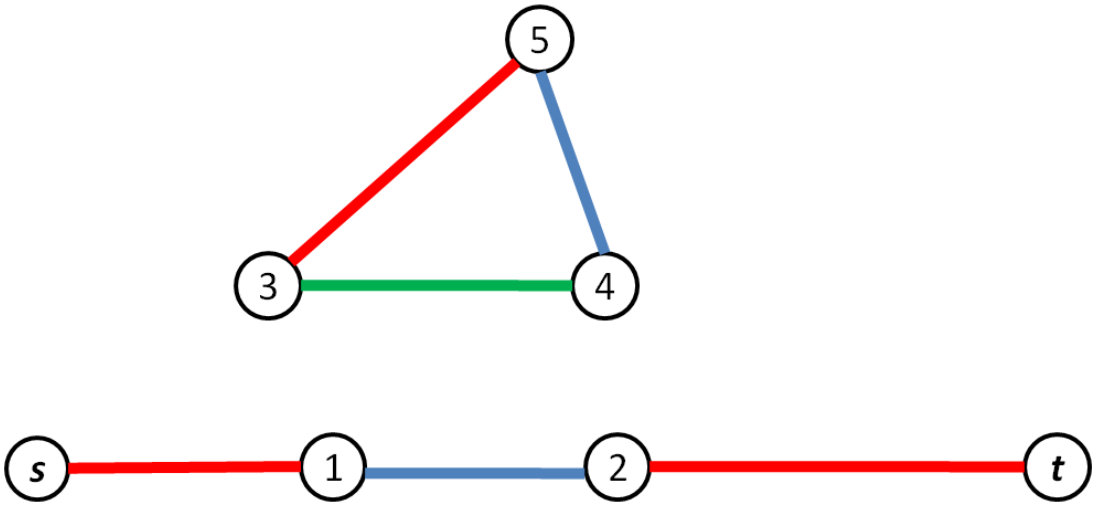

Figura 3.12: Solução contendo um sub-ciclo propriamente colorido

Para determinarmos uma $s-t$ trilha propriamente colorida em um Grafo G^c, Abouelaoualim et al. [1] provou que equivale a determinarmos um $s-t$ caminho propriamente colorido em um grafo trilha-caminho $p-H^c$ *com* $2 \leqslant p \leqslant \lfloor \frac{n-1}{2} \rfloor$, onde p representa o número máximo possível de visitas que um vértice $v \in V(G^c) \setminus \{s,t\}$ pode ter e é dado por $p = \lfloor \frac{d_{G^c}(v)}{2} \rfloor$, onde $d_{G^c}(v)$ representa o grau do vértice v em G^c.

Utilizamos a descrição do grafo-trilha $p-H^c$ descrita na Seção 2.1 para um vértice $v \in V(G^c) \setminus \{s,t\}$ intermediário com $p = \lfloor \frac{n-1}{2} \rfloor$, pois estamos interessados em determinar a maior $s-t$ trilha propriamente colorida em G^c. A Figura 3.13 ilustra um grafo G^c e a transformação de 3 arestas $xy \in E(G^c) \mid x,y \in V(G^c) \setminus \{s,t\}$ em subgrafos em $p-H^c$.

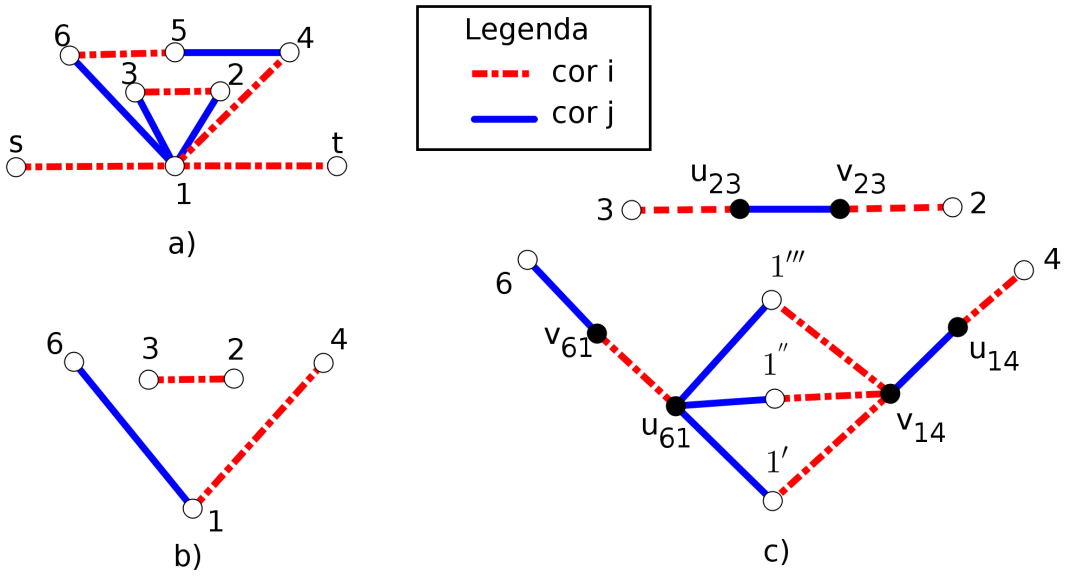

Figura 3.13: a)Um grafo G^c com uma $s-t$ trilha propriamente colorida com $p = \lfloor \frac{n-1}{2} \rfloor$ para um vértice $v \in G^c \setminus \{s,t\}$. b) 3 arestas $xy \in G^c$. c) subgrafos equivalente às arestas xy em $p - H^c$

A Solução obtida do maior $s-t$ caminho propriamente colorido em $p-H^c$ na chamada do **Algoritmo 9** implica na solução da maior $s-t$ trilha propriamente colorida em um grafo G^c qualquer.

Algoritmo 10 Obter a maior $s-t$ trilha propriamente colorida de um grafo G^c qualquer

Entrada: Um grafo G^c com c arestas coloridas e vértices $s,t \in V(G^c)$

Saída: Caso exista, a maior $s-t$ trilha propriamente colorida em um Grafo G^c Qualquer

1: **Inicio**
2: Gere o grafo trilha-caminho $p - H^c$ conforme descrito na Seção 2.1
3: Aplique o **Algoritmo 9** em $p-H^c$ para encontrar o maior $s-t$ caminho propriamente colorido P
4: Use P para construir a maior $s-t$ trilha propriamente colorida T equivalente em G^c
5: **Fim**

Ao analisarmos os **Algoritmos 9 e 10**, concluímos que a obtenção do maior $s-t$ caminho/trilha propriamente colorido(a) em um grafo G^c qualquer pode não ser determinada diretamente caso a solução encontrada contenha ciclos propriamente coloridos na solução. No entanto, podemos chegar a uma solução ótima através da aplicação de um modelo de Programação Linear Inteira (PLI) a partir do grafo Original G^c. A aplicação de métodos exatos será discutida mais adiante no Capítulo 4.

3.8 Obtenção do menor custo de recoloração de arestas para encontrar um $s-t$ caminho prop. colorido em G^c

De acordo com o resultado proposto em [29] (veja o *Teorema 8* na seção 2.2), descreveremos, nesta seção, um algoritmo polinomial para o calcular o custo de recoloração mínimo necessário para obtermos um $s-t$ caminho propriamente colorido em um grafo G^c, que não contenha inicialmente $s-t$ caminhos propriamente coloridos. Neste problema, dados dois vértices $s, t \in V(G^c)$ e uma matriz de custo de recoloração R, o objetivo é determinar um novo grafo G^c_{new} contendo um $s-t$ caminho propriamente colorido com o menor custo de recoloração de arestas.

Em nossa implementação, utilizamos uma adaptação do P-gadget **XP-Gadget** (veja Seção 3.1) para construção dos subgrafos G_x. Para o restante da construção do grafo G, seguimos o mesmo procedimento proposto em [29]. A definição do subgrafo G_x para cada $x \in V(G^c)$ com a seguinte definição do conjunto de vértices e arestas é proposta a seguir:

$$V(G_x) = \{x_j : j \in I_c\} \cup \{x'_j : j \in I_c \setminus \{m, M\} \tag{3.7}$$

$$E(G_x) = \{x_m x_M\} \cup \{x_j x'_j, x_m x'_j, x_M x'_j : j \in I_c \setminus \{m, M\}\} \tag{3.8}$$

A mudança proposta na definição dos conjuntos $V(G_x)$ e $E(G_x)$ citada nas equações 3.7 e 3.8 em relação aos conjuntos $V(G_x)$ e $E(G_x)$ respectivamente citados nas equações 3.5 e 3.6 da Seção 3.1, está na criação dos vértices x_j, onde j agora se refere ao conjunto de cores I_c e não mais em relação à vizinhança de cores do vértice. Em outras palavras, para cada cor j do conjunto de cores I_c, temos um vértice x_j associado. A geração da quantidade de vértices x_j igual à cardinalidade do conjunto I_c se faz necessária para garantir que o menor custo de recoloração (que é gerado pelo cálculo do emparelhamento perfeito de custo mínimo realizado sobre o grafo não-colorido G correspondente) no grafo G^c seja obtido.

O grafo G construído a partir do grafo G^c da Figura 3.15 e com base nas novas definições dos subgrafos G_x e é ilustrado na Figura 3.16 seguir:

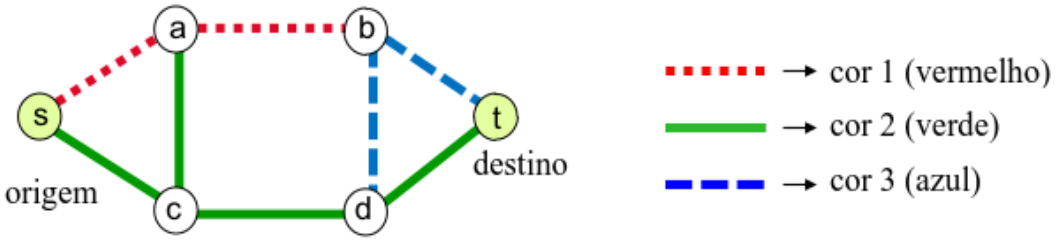

Figura 3.14: Exemplo de um Grafo G^c com 3 cores nas arestas que não contém um $s-t$ caminho propriamente colorido [29]

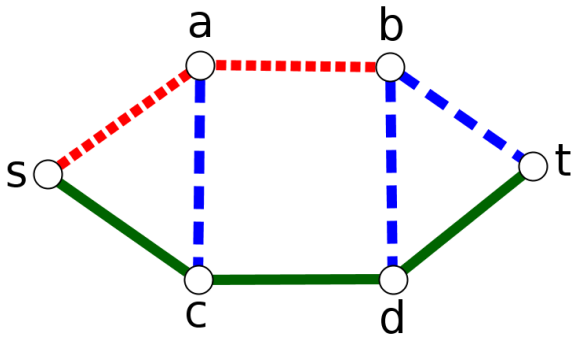

Figura 3.15: Exemplo de um Grafo G^c_{new} com 3 cores nas arestas com um $s-t$ caminho propriamente colorido após a execução do Algoritmo 11 no grafo G^c

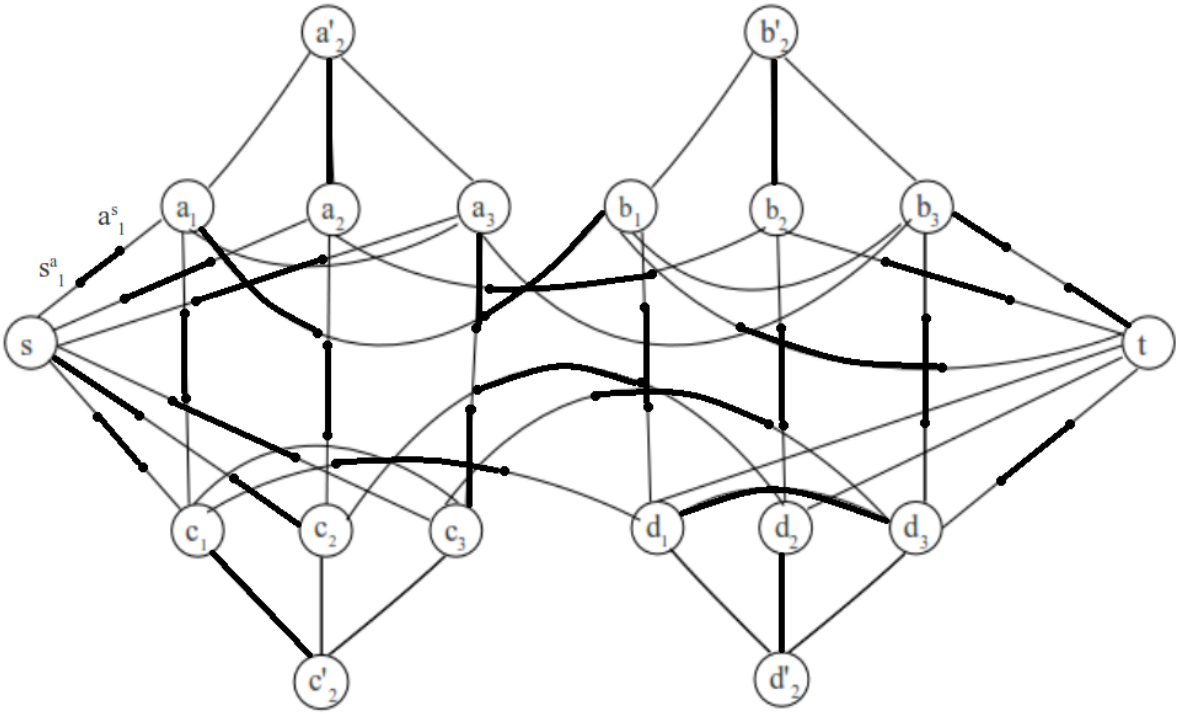

Figura 3.16: Exemplo de um Grafo G com emparelhamento perfeito M^* equivalente ao grafo G^c da Figura 3.15

As novas arestas $e \in E(G) \setminus E(G_x)$ no grafo G da Figura 3.16 têm os mesmos custos definidos para as arestas na construção do grafo G definido em [29].

Para cada vértice $v_i \in G^c \setminus \{s,t\}$, temos o conjunto $V(G_x)$ formado pelos vértices x_i representando cada cor i no conjunto de cores I_c e pelos vértices x'_i onde $i \in I_c \setminus \{m, M\}$, ou seja, os vértices x_i representam as cores do conjunto I_c com exceção da cor inicial m e da cor final M. Embora a complexidade seja a mesma do procedimento utilizado em [29] ($O(n|I_c|^{2.5})$), quantidade de vértices gerados com este procedimento gera um grafo G com uma quantidade de vértices $2|I_c| - 2$ contra $2|I_c| + 2$ do grafo G proposto em [29] para cada G_x gerado.

A matriz de custos de recoloração R para o conjunto de cores I_c do grafo G^c satisfaz a desigualdade triangular como proposta em [29] para permitir que uma aresta seja trocada de cor no máximo uma vez. Por exemplo, se o custo de recoloração de uma aresta xy de uma cor i para uma cor j fosse maior que o custo de recoloração desta mesma aresta da cor i para a cor k e da cor j para a cor k, teríamos que fazer duas trocas de cores para esta aresta. Para isto, para cada cor i do conjunto I_c foi gerado aleatoriamente uma coordenada cartesiana (x, y) e assim os custos de recoloração para cada par de cores do conjunto I_c foram relacionados às distancias euclidianas dos pontos gerados no plano

cartesiano.

Após a definição da matriz de custos de recoloração R, definimos o Gadget denominado **R-XP-Gadget** para a construção do grafo não-colorido G associado ao grafo original G^c, baseado nas definições dos subgrafos G_x definidas nas equações 3.7 e 3.8 acima e nas regras definidas em [29] (veja seção 2.2) para a construção do restante do grafo. A quantidade de vértices gerados no grafo G é dada pela seguinte fórmula:

$$|V(G)| = 2 \times |I_c| \times m + |V(G_x)| \times |V(G^c) \setminus \{s, t\}|,$$

onde $|I_c|$ é a cardinalidade do conjunto de cores, m é a quantidade de arestas do grafo G^c original, $|V(G_x)|$ é a cardinalidade dos subgrafos G_x que é definido como $|V(G_x)| = 2 \times |I_c| - 2$ e $|V(G^c) \setminus \{s, t\}|$ é a cardinalidade do conjunto dos vértices do grafo original G^c com exceção dos vértices inicial s e final t.

Com o grafo G construído, a seguir é obtido o emparelhamento M^* resultante da aplicação do solver "Blossom V" definido em [25] no grafo G. Se o custo do emparelhamento M^* for 0, significa que o grafo original G^c já possui um $s-t$ caminho propriamente colorido e nenhuma aresta de G^c deverá mudar de cor. Caso contrário, o custo de M^* está associado às arestas do conjunto M^* com pesos obtidos da matriz de custos de recoloração R maiores que 0. Com isto, um novo grafo denominado G^c_{new} é construído com base em G^c da seguinte forma: as arestas de G^c que não estão associadas com as arestas de M^* com custo 0 permanecem inalteradas enquanto que as arestas de G^c associadas às arestas de M^* terão suas cores alteradas de acordo com os pesos destas arestas de M^* baseados na matriz de custos de recoloração R. Este procedimento está definido no **Algoritmo 11**:

3.9 Detalhes da implementação: Descrição do formato do arquivo de entrada

Nas implementações utilizadas neste trabalho, os grafos de entrada para os algoritmos foram representados por um arquivo no formato padrão DIMACS (*Center for Discrete Mathematics for Computer Science*) [23]. O arquivo de entrada contém todas as informações a respeito do grafo para a resolução do problema. A seguir descreveremos as principais características do formato deste arquivo de entrada:

Comentários: linhas de comentário trazem informações adicionais sobre o conteúdo do arquivo e são ignoradas pelo programa. Podem aparecer em qualquer lugar do arquivo

Algoritmo 11 Obtenção do menor custo de recoloração de um grafo G^c para obtenção um $s-t$ caminho propriamente colorido em G^c_{new}

Entrada: Um grafo G^c com c arestas coloridas e vértices $s, t \in V(G^c)$
Saída: Um novo grafo G^c_{new} com menor custo de recoloração contendo o $s-t$ caminho propriamente colorido e a relação das arestas com suas cores a serem trocadas

1: **Inicio**
2: Leia a matriz de custo de recoloração R associado ao conjunto de cores I_c
3: $G \leftarrow$ R-XP-Gadget(G^c)
4: Encontre o emparelhamento perfeito de custo mínimo M^* no grafo G
5: **se** custo(M^*)=0 **então**
6: O grafo G^c já contém um $s-t$ caminho propriamente colorido.
7: **senão**
8: Defina o grafo G^c_{new} com base no custo(M^*)
9: Retorne as arestas com suas cores a serem trocadas baseadas na diferença entre $E(G^c_{new})$ e $E(G^c)$
10: **fim se**
11: **Fim**

e iniciam com a letra **c** em minúsculo. Exemplo:

```
c Exemplo de uma linha de comentário
```

Linha do problema: Só há uma linha de problema por arquivo de entrada. Para o nosso estudo, esta linha descreve informações gerais do grafo como: quantidade de vértices, quantidade de arestas e quantidade de cores das arestas e precisa estar localizada antes de qualquer linha descritora de arestas do grafo. Esta linha tem o seguinte formato:

```
p FORMAT NODES EDGES COLORS
```

A letra minúscula **p** é obrigatória e identifica a linha do problema no arquivo de entrada. O campo `FORMAT` é utilizado para manter a compatibilidade com formatos anteriores e contém a palavra "edge". O campo `NODES` contém um valor inteiro especificando o numero de vértices n do grafo. O campo `EDGES` contém um valor inteiro representando a quantidade m de arestas do grafo. O campo `COLORS` contém a quantidade c de cores das arestas do grafo.

Descritores de Arestas: Para cada aresta do grafo há uma linha descritora de aresta no arquivo. Para o nosso estudo cada linha descreve as seguintes informações: identificação do vértice inicial, identificação do vértice final, peso e cor de cada aresta. Esta linha tem o seguinte formato:

```
e SOURCE TARGET WEIGHT COLOR
```

A letra minúscula **e** é obrigatória e identifica a linha descritora de aresta no arquivo de

entrada. Os campos `SOURCE` e `TARGET` contém um inteiro com a identificação dos vértice de origem e destino da aresta. O campo `WEIGHT` contém um valor inteiro representando o peso da aresta e o campo `COLOR` contém um valor inteiro representando a cor da aresta dentro do conjunto de cores definido. Cada aresta só é definida uma única vez no arquivo, ou seja, uma vez definida uma aresta (u, v), onde u representa o vértice `SOURCE` e v representa o vértice `TARGET` a aresta (v, u) não pode aparecer na entrada do arquivo.

Um exemplo completo da representação de um grafo com 6 vértices, 6 arestas e 2 cores é mostrado no exemplo abaixo:

```
   p edge 6 6 2
e 0 1 1 1
e 1 2 1 2
e 2 3 1 1
e 3 4 1 2
e 4 1 1 1
e 1 5 1 2
```

Capítulo 4

Modelo de PLI para determinação do maior $s - t$ caminho propriamente colorido

Neste capítulo descreveremos com detalhes um modelo de Programação Linear Inteira(PLI) na determinação do maior $s - t$ caminho propriamente colorido restrito a um grafo G^c qualquer. Antes, porém, abordaremos alguns conceitos de Modelagem Matemática, necessários para melhor contextualização. Este modelo foi implementado na linguagem C++ utilizando a API do software CPLEX (Solver de programação matemática de alto desempenho para a programação linear, programação inteira mista e programação quadrática).

4.1 Modelagem Matemática em Otimização

Segundo Goldbarg e Luna [16], modelos são representações simplificadas da realidade que preservam, para determinadas situações e enfoques, uma equivalência adequada. A partir da observação de fenômenos, processos ou sistemas, que podem ser físicos, químicos, biológicos, econômicos, buscam-se leis que os regem. Essas leis, se passíveis de serem descritas por relações matemáticas, dão origem aos modelos matemáticos. Modelagem matemática é uma técnica de representação quantitativa de processos e problemas reais. Também pode ser descrita como o conjunto de equações matemáticas ou de relações lógicas desenvolvidas para descrever um sistema real.

No Processo de modelagem consideraremos alguns elementos básicos a seguir:

- Parâmetros do modelo: São todos os dados conhecidos do processo, utilizados como valores de entrada do modelo. Por exemplo, em um problema de corte de materi-

ais, precisamos saber a quantidade dos itens em estoque, a dimensão dos itens em estoque, os dados da demanda, etc.

- Variáveis de Decisão: São as incógnitas do modelo. Seus valores interferem na performance do sistema e o modelo opera como um mecanismo de melhora dentro de um intervalo operacional. Como exemplo para os problemas de corte, as dimensões dos objetos de estoque escolhidas para corte, os padrões de corte, as repetições e o sequenciamento dos padrões são variáveis de decisão

- Restrições: Estabelecem condicionantes entre as variáveis de decisão e a dinâmica do sistema, indicando as limitações físicas, disponibilidades de material e as necessidades a serem atendidas pelo processo a ser modelado. Para o nosso exemplo, temos como restrição as características dos cortes possíveis.

- Função Objetivo: É a representação matemática do objetivo a ser atendido pelo modelo. A função objetivo mede a eficiência de cada solução possível.

A programação matemática faz uso dos modelos matemáticos para tratar de problemas de decisão em espaços de dimensões finitas. Variáveis são definidas e são estabelecidas relações matemáticas entre essas variáveis de forma a descrever o comportamento do sistema [2].

Formalmente um modelo matemático é escrito da seguinte maneira:

Maximizar/Minimizar $f(x)$

sujeito a:

$g(x) \geqslant a$

$h(x) \leqslant b$

$s(x) = c$

$l \leqslant \ x \ \leqslant \ u$

Em que f(x) é uma função, linear ou não linear, a ser otimizada (maximizada ou minimizada) sujeito às restrições (lineares ou não lineares) que indicam uma necessidade a ser satisfeita (“maior ou igual”),uma disponibilidade a ser respeitada (“menor ou igual”) ou um condicionante a ser satisfeito de maneira exata (“igual”).

Para uma melhor classificação dos problemas, Goldbarg e Luna [16] subdividem a programação matemática em algumas sub-áreas, que são:

- Programação Linear: Quando todas as variáveis são contínuas e a função objetivo e as restrições apresentam comportamento linear
- Programação Não-Linear: Quando ocorre algum tipo de não-linearidade, seja na função objetivo ou em qualquer uma de suas restrições
- Programação Inteira: Quando qualquer variável não puder assumir valores contínuos, ficando condicionada a assumir valores discretos.
- Programação Linear Inteira Mista: Neste caso há uma mistura da programação linear com a programação inteira, ou seja, algumas das variáveis assumem valores discretos, mas apresentam comportamento linear.

Estas sub-áreas baseiam-se na representação dos objetivos e restrições de um processo para o qual se deseja descobrir soluções otimizadas, ou seja, o modelo é escrito segundo uma técnica que permite encontrar a melhor solução ou política de ação para os condicionantes representados e podem ser representados por métodos exatos ou aproximados.

Os métodos exatos são capazes de garantir se a solução encontrada é ótima ou não. Uma solução é dita ótima se, sob um certo critério de otimização, não existir nenhuma outra solução do problema com um valor melhor que ela. Os principais métodos exatos são os métodos de enumeração. Estes métodos consistem em organizar uma busca no conjunto de soluções viáveis do problema (soluções que não desrespeitam nenhuma restrição do problema) de maneira que este conjunto é todo vasculhado e uma solução ótima é armazenada. Os métodos aproximados produzem uma solução de "boa qualidade" e são indicados quando a obtenção da melhor solução implica em um esforço computacional considerável. Portanto, às vezes soluções tão próximas da solução ótima são mais interessantes se levarmos em consideração o tempo reduzido disponível. Neste trabalho, utilizamos a Programação Linear Inteira para formulação e resolução dos problemas relacionados ao maior caminho e maior trilha propriamente colorido(a) em um grafo qualquer G^c com c cores nas arestas.

4.2 Formulação Matemática: Obtenção do maior $s-t$ caminho prop. colorido em um Grafo G^c qualquer

Para resolução do problema da obtenção do maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer, temos os seguintes parâmetros de entrada:

- n: Número de vértices do grafo G^c;
- $V(G^c)$: Conjunto de vértices do grafo G^c, onde $v \in \{0..n-1\}$
- S: Subconjunto de vértices do grafo G^c.
- I_c: Conjunto de cores k do grafo G^c;
- d: Custo(peso) de cada arco. Neste problema, assumimos que todos os custos são constantes e iguais a 1;
- $E(G^c)$: Conjunto de arcos do Grafo G^c, onde cada arco é formado por uma tupla $< i, j, k >$, onde i é o vértice origem, j é o vértice destino e k é sua cor.

Variáveis de Decisão:

- x_{ij}^k: Arcos de G^c, onde i representa o vértice origem, j representa o vértice destino e k representa a cor da aresta, onde $i, j \in V(G^c), k \in I_c$

$$x_{i,j}^k = \begin{cases} 1, & \text{se o arco } (i,j) \text{ na cor } k \text{ pertencer ao caminho} \\ 0, & \text{caso contrário.} \end{cases}$$

Considerando o nó s como nó origem, onde apenas arcos saindo de s são permitidos e t como nó destino, onde apenas arcos chegando em t são permitidos temos o seguinte modelo de PLI:

$$Maximizar \sum_{(i,j)\in E(G^c),\ k\in I_c} x_{i,j}^k \tag{4.1}$$

Sujeito a:

$$\sum_{j\in V(G^c)\ e\ k\in I_c} x_{sj}^k \ = \ 1 \tag{4.2}$$

$$\sum_{i\in V(G^c)\ e\ k\in I_c} x_{it}^k \ = \ 1 \tag{4.3}$$

$$\sum_{k\in I_c}\sum_{i\in V(G^c)} x_{ij}^k \ - \ \sum_{k'\in I_c}\sum_{l\in V(G^c)} x_{jl}^{k'} = 0; \qquad \forall j \in V(G^c)\setminus\{s,t\} \tag{4.4}$$

$$\sum_{i\in V(G^c)} x_{ij}^k \ + \ \sum_{l\in V(G^c)} x_{jl}^k <= 1, \qquad \forall j \in V(G^c)\setminus\{s,t\}, k \in I_c \tag{4.5}$$

$$\sum_{k\in I_c}\sum_{i\in V(G^c)} x_{ij}^k <= 1,\ \forall j \in V(G^c)\setminus\{s\} \tag{4.6}$$

$$\sum_{k\in I_c}\sum_{l\in v(G^c)} x_{jl}^k <= 1,\ \forall j \in V(G^c)\setminus\{t\} \tag{4.7}$$

$$\sum_{k\in I_c}\sum_{(i,j)\in E(S)} x_{ij}^k <= |S| \ - \ 1,\ \forall S \subset V(G^c)\setminus\{s,t\},\ 2 \leqslant |S| \leqslant n-2. \tag{4.8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V(G^c),\ k \in I_c. \tag{4.9}$$

O modelo de PLI descrito resume-se a obter o maior número de arcos propriamente coloridos entre um vértice origem s e um vértice destino t sem que estes arcos formem ciclos.

Um $s-t$ caminho propriamente colorido pode ser representado por um conjunto de variáveis binárias onde cada variável x_{ij}^k está diretamente associada a um respectivo arco de um grafo G^c qualquer. Cada variável assume o valor 1 se existe um arco de cor k entre os vértices i,j, indicando que este arco faz parte do $s-t$ caminho propriamente colorido no grafo G^c e, caso contrário, tal variável assume o valor 0, indicando que esta aresta do grafo G^c não pertence ao conjunto solução.

A função objetivo representada na Equação 4.1, maximiza o número de arcos existentes no caminho percorrido do vértice origem s até o vértice destino t. Contudo, não queremos simplesmente obter o maior número de arcos entre s e t e sim o maior número de arcos entre s e t na condição de que arcos adjacentes sejam de cores diferentes. Para isto, precisamos definir restrições no modelo de PLI para a obtenção do maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer.

As restrições correspondentes às Equações 4.2 e 4.3 garantem que exatamente um arco deve sair do vértice origem s e exatamente um arco deve chegar no vértice destino t de acordo com a definição do grafo de entrada. A Equação 4.4 garante o equilíbrio de fluxo dos nós do grafo com exceção dos nós s e t, ou seja, a quantidade de arcos que entram em um vértice v tem que ser igual à quantidade de arcos que saem de v. A Equação 4.5 garante uma outra importante propriedade, que é a de que arcos adjacentes tenham cores diferentes, em outras palavras, garantir um caminho propriamente colorido. Como estamos tratando de caminhos, as Equações 4.6 e 4.7 garantem que um nó só poderá ser visitado no máximo uma vez. E finalmente a última restrição, correspondente à Equação 4.8, garante a eliminação de subciclos na solução gerada. A presença de subciclos influencia negativamente na obtenção da melhor solução para o problema e deve ser evitada a fim de obtermos uma solução ótima do problema abordado. Existem várias formulações de restrições de eliminação de subciclos para o problema do caixeiro viajante existentes na literatura e optamos pela formulação definida em [8] para aplicarmos ao nosso problema. Esta formulação leva um conta um subconjunto S de vértices, onde se a soma dos arcos representados pela variáveis x_{ij}^k onde $x_{ij}^k \in S$ deve ser menor que a quantidade de vértices de S. A Equação 4.9 garante a integralidade e não-negatividade da variável x_{ij}^k.

Em nossa implementação, construímos um procedimento baseado nesta formulação, onde uma vez detectado um subciclo na solução, uma nova restrição com este subciclo é adicionada ao modelo em tempo de execução, sendo este novamente submetido ao cálculo pelo solver CPLEX. Este processo se repete até que a solução encontrada não contenha mais subciclos, neste caso, teremos encontrado uma solução ótima do problema, ou seja, encontramos o maior $s-t$ caminho (hamiltoniano ou não) propriamente colorido em um grafo G^c.

Capítulo 5

Experimentos Computacionais

Neste capítulo descreveremos em detalhes os testes computacionais realizados para o problema da determinação do menor e do maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer. Para os testes do menor $s-t$ caminho propriamente colorido, comparamos o desempenho do Algoritmo baseado na construção XP-Gadget e emparelhamento perfeito (Algoritmo 2, veja Seção 3.3), que a partir deste ponto denominaremos MIN-PECPATH, com um Algoritmo baseado no modelo matemático definido na Seção 4.2 com a função objetivo alterada para minimização, que a partir deste ponto denominaremos MIN-PECPATH-LP. Nos testes da determinação do maior $s-t$ caminho propriamente colorido para um grafo G^c qualquer, comparamos o desempenho do **Algoritmo 9** (veja Seção 3.7), considerando os casos para um grafo particular (sem ciclos propriamente coloridos) e para um grafo qualquer (geralmente com a presença de ciclos propriamente coloridos). Para a primeira parte do Algoritmo 9 onde o maior $s-t$ caminho propriamente colorido é resolvido polinomialmente (solução sem ciclos propriamente coloridos), denominaremos MAX-PECPATH. Para a segunda parte do Algoritmo 9 onde o maior $s-t$ caminho propriamente colorido é resolvido através da implementação do modelo matemático definido na Seção 4.2, denominaremos MAX-PECPATH-LP. A seguir, são descritos o ambiente de execução dos algoritmos e as instâncias utilizadas nos testes e em seguida fazemos uma análise dos resultados obtidos.

5.1 Ambiente de Execução

Os algoritmos testados neste trabalho foram implementados na linguagem C++ e compiladas com o g++ versão 4.6.3. Além disso, os modelo matemáticos implementados nos algoritmos MIN-PECPATH-LP e MAX-PECPATH-LP foram construídos utilizando

a API do solver CPLEX. Os testes computacionais foram realizados em um computador equipado com processador Intel®Core™i5 - 2450 M CPU @ 2.50 GHz x 4 com 4GB de memória RAM e Sistema Operacional Linux Ubuntu versão 12.04 32-bit kernel 3.2.0-56-generic-pae.

5.2 Instâncias Utilizadas

As instâncias utilizadas neste trabalho foram criadas randomicamente e foram classificadas em dois tipos de grafos: esparsos e densos. Para cada grafo gerado com n vértices foram criadas m arestas aleatórias formadas por pares de vértices (x, y) onde x e y estão compreendidos entre 0 e $n-1$ e $x \neq y$. O número de arestas em grafos esparsos é definido como uma função linear do número de vértices e o número de arestas em grafos densos é definido como uma função quadrática do número de vértices.

Nos testes para o menor $s-t$ caminho propriamente colorido, os grafos esparsos foram gerados em instâncias cuja quantidade n de vértices variam no conjunto $\{10000, 20000, 40000\}$ e cuja quantidade m de arestas está representada através de 3 funções lineares: $m = 2n$, $m = 4n$, $m = 8n$. Os grafos densos foram gerados em instâncias cuja quantidade n de vértices variam no conjunto $\{500, 1000, 2000\}$. Para $n = 500$ e $n = 1000$ a quantidade m de arestas está representada através das funções quadráticas: $m = n^2/10$, $n^2/5$ e $2n^2/5$. Para $n = 2000$ a quantidade m de arestas está representada através das funções quadráticas: $n^2/20$, $n^2/10$ e $n^2/5$. Além disso, para cada par (n, m) de vértices e arestas de um grafo G, temos dois conjuntos de quantidades de cores: I_c^{min}, onde $|I_c^{min}| = 2$ e I_c^{max}, onde $|I_c^{max}| = \Delta(G^c)$. Ou seja, I_c^{min} representa o conjunto mínimo de cores para denominarmos um grafo G com arestas coloridas e I_c^{max} representa um conjunto superior de cores que, para viabilizar os nossos testes, convencionamos que este valor seria igual ao grau máximo $\Delta(G^c)$ de um grafo G^c , onde $\Delta(G^c) = max\{d_{G^c}(v)|v \in V(G^c)\}$. A justificativa para a escolha deste valor se dá em virtude de que o numero de cores mínimo necessário para termos um grafo G^c onde todas as arestas incidentes aos vértices do grafo G^c possuam cores distintas é $\Delta(G^c)$. A partir deste ponto denominaremos como $C = |I_c^{max}|$, o número máximo de cores do grafo G^c.

Nos testes para o maior $s-t$ caminho propriamente colorido, por se tratar de um problema NP-Difícil, utilizamos apenas grafos esparsos com 3 cores nas arestas e reduzimos o tamanho das instâncias em relação aos testes com o menor $s-t$ caminho propriamente colorido. Os grafos esparsos foram gerados em instâncias cuja quantidade n de vértices

variam no conjunto $\{250, 500, 1000\}$ e cuja quantidade m de arestas está representada através das funções lineares: $m = 2n$, $m = 3n$ e $m = 4n$.

Portanto a identificação de cada instancia é composta pela quantidade de vértices n, pela quantidade de arestas m e pela quantidade de cores c.

5.3 Comparação entre os algoritmos

Para realizar a comparação entre os algoritmos, cada um foi executado 10 vezes para cada instância com dez sementes distintas. Ou seja, para cada semente, um novo grafo aleatório é gerado com n vértices, m arestas e c cores. Ao final, é gerada a média dos tempos das execuções dos algoritmos para os 10 problemas testes gerados de cada instância.

Na geração dos grafos aleatórios, para que exista um caminho $s - t$, é necessário que os vértices s e t estejam na mesma componente conexa de G^c, caso contrário, nenhuma solução pode ser obtida. Esta restrição foi implementada juntamente com uma restrição para que os vértices s e t não sejam adjacentes, para termos um caminho $s - t$ com pelo menos duas cores para o caso do menor $s - t$ caminho propriamente colorido.

Nos testes com o maior $s - t$ caminho propriamente colorido, geramos uma classe particular de grafos sem ciclos propriamente coloridos a fim de testarmos a primeira parte do **Algoritmo 9**, onde o problema é resolvido polinomialmente através da construção XP-Gadget e emparelhamento perfeito. Para a geração desse grafo particular, a partir de um grafo G não colorido com n vértices e m arestas, onde para cada vértice $v \in G$ escolhemos uma cor i no conjunto de cores I_c e atribuímos esta cor i a todas as arestas não coloridas incidentes a v. No final temos um grafo G^c sem ciclos propriamente coloridos (veja Teorema 4 na Seção 2.1).

Para esta classe especial de grafos, testamos também uma solução baseada no modelo matemático descrito na Seção 4.2 e comparamos os resultados obtidos. Para o caso geral, quando o grafo praticamente possui ciclos propriamente coloridos na solução (o que ocorreu em $92,2\%$ dos casos), apenas a segunda parte do Algoritmo 9 (MAX-PECPATH-LP) é executada.

Os tempos considerados nos testes levaram em conta apenas o momento do início da chamada das funções respectivas aos algoritmos testados. Não foram considerados os tempos de leitura dos arquivos e do carregamento em memória. Em algumas instancias, os tempos de execução dos algoritmos não foram computados devido a ocorrência de falta

de memória do computador. Nas Tabelas abaixo estes casos estão definidos pela palavra "mem".

5.3.1 Menor $s - t$ caminho propriamente colorido

As Tabelas 5.1 e 5.2 abaixo, apresentam respectivamente os resultados computacionais obtidos para o conjunto de instâncias representados pelos grafos esparsos e grafos densos. Cada linha das Tabelas representa a média dos tempos dos algoritmos correspondentes a 10 execuções de cada instancia.

Nº	INSTÂNCIA			Função	Tempo em segundos	
	Vértices	Arestas	Cores		MIN-PECPATH	MIN-PECPATH-LP
1	10000	20000	2	2n	**0,018**	5,467
2	10000	20000	14	2n	**0,053**	3,163
3	10000	40000	2	4n	**0,057**	10,171
4	10000	40000	21	4n	**0,144**	6,331
5	10000	80000	2	8n	**0,211**	13,219
6	10000	80000	33	8n	**0,418**	11,106
7	20000	40000	2	2n	**0,051**	17,243
8	20000	40000	14	2n	**0,122**	7,590
9	20000	80000	2	4n	**0,174**	27,027
10	20000	80000	21	4n	**0,342**	14,810
11	20000	160000	2	8n	**0,587**	29,985
12	20000	160000	34	8n	**0,985**	25,864
13	40000	80000	2	2n	**0,137**	16,335
14	40000	80000	16	2n	**0,286**	16,286
15	40000	160000	2	4n	**0,437**	63,925
16	40000	160000	22	4n	**0,777**	32,767
17	40000	320000	2	8n	**1,361**	60,398
18	40000	320000	35	8n	mem	mem

Tabela 5.1: Comparação entre os tempos de execução dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para instancias de grafos esparsos

Podemos verificar na Tabela 5.1 que em todas as instancias com exceção da última, o tempo de execução do algoritmo MIN-PECPATH foi bem inferior ao do algoritmo

MIN-PECPATH-LP. Percebemos também que o aumento do tempo de execução do algoritmo MIN-PECPATH é diretamente proporcional ao aumento da densidade do grafo não-colorido (gerado a partir da construção XP-Gadget) equivalente ao grafo colorido original. Analisando a Tabela 5.1, o pior desempenho do Algoritmo MIN-PECPATH em relação ao Algoritmo MIN-PECPATH-LP ocorre na instancia 2, onde a diferença nos tempos de execução dos Algoritmos é de 3,11 segundos. O melhor desempenho ocorre na instância 15, onde a vantagem do algoritmo MIN-PECPATH foi de 63,5 segundos. Podemos concluir com os gráficos ilustrados nas Figuras 5.1 e 5.2 abaixo que o aumento do número de cores em instancias de grafos esparsos, diminui a eficácia do algoritmo MIN-PECPATH em relação ao algoritmo MIN-PECPATH-LP e isto pode ser explicado com mais detalhes nos gráficos ilustrados na Figura 5.3 onde em todas as instâncias o tempo de execução do algoritmo MIN-PECPATH para c cores foi superior ao tempo de execução do mesmo Algoritmo para 2 cores e na Figura 5.4 onde em todas as instâncias o tempo de execução do algoritmo MIN-PECPATH-LP para c cores foi inferior ao tempo de execução do mesmo Algoritmo para 2 cores.

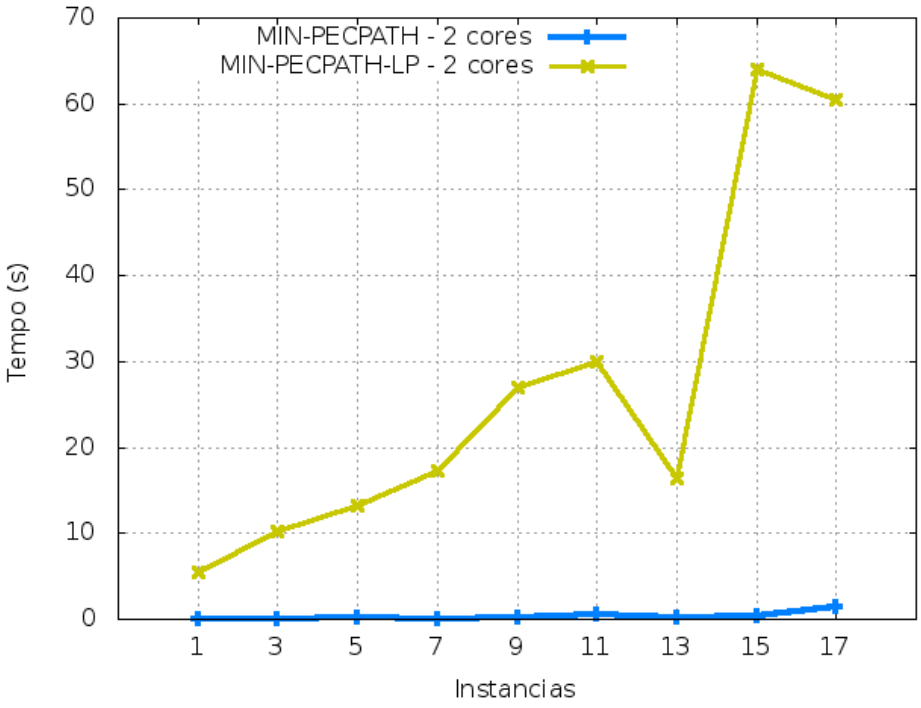

Figura 5.1: Desempenho dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para grafos esparsos para 2 cores

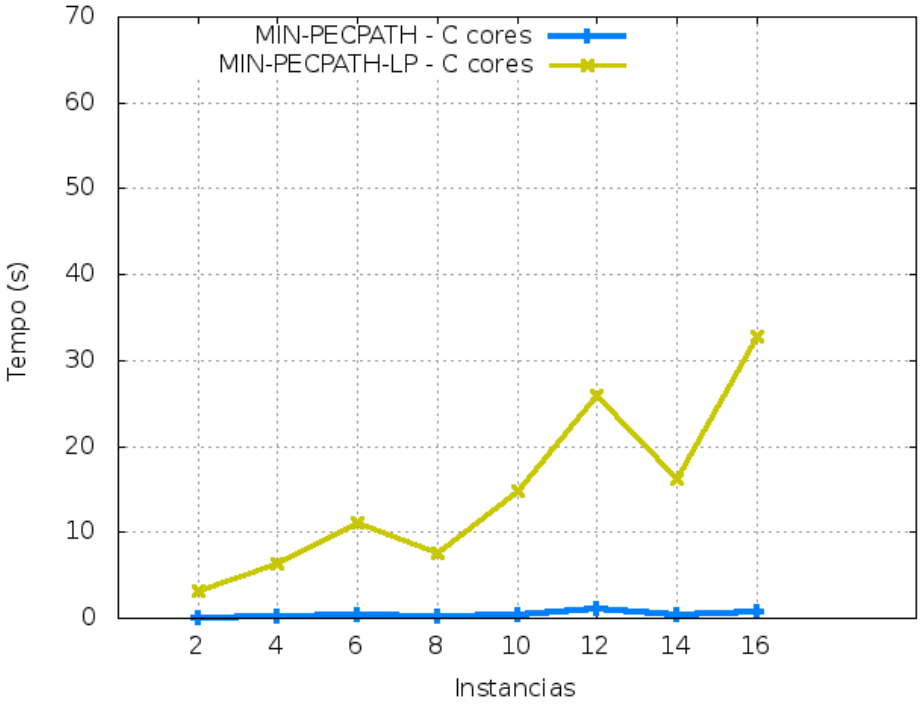

Figura 5.2: Desempenho dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para grafos esparsos para C cores

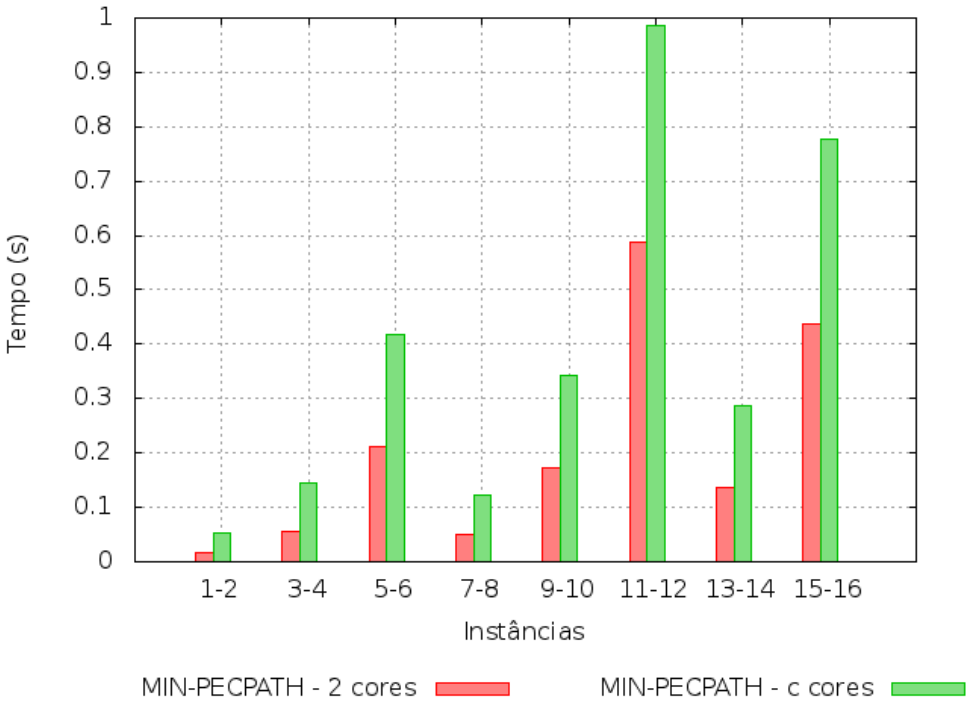

Figura 5.3: Desempenho do algoritmo MIN-PECPATH com relação à quantidade de cores para grafos esparsos

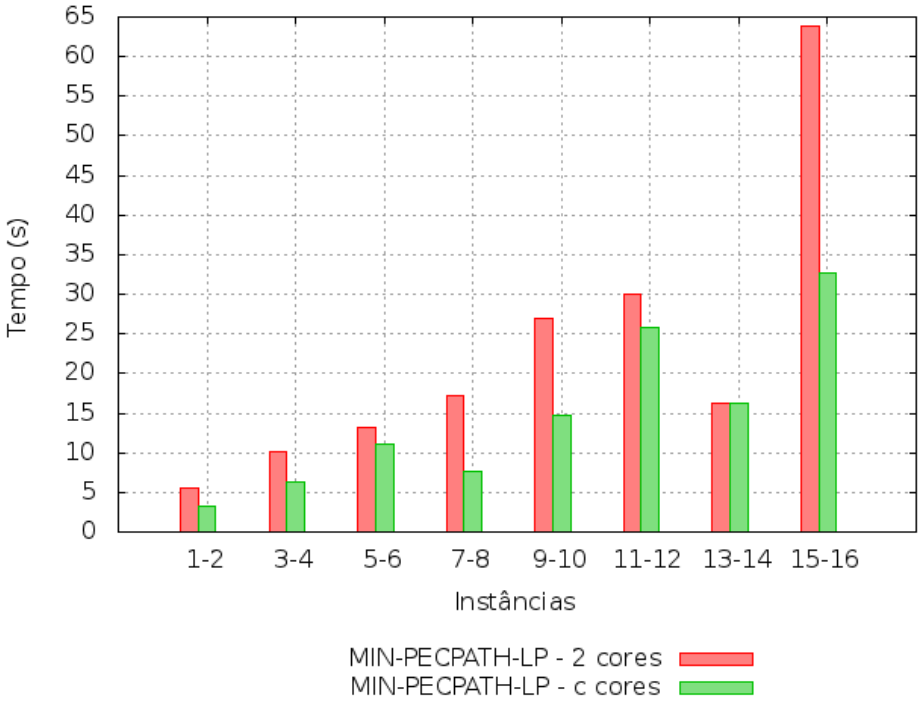

Figura 5.4: Desempenho do algoritmo MIN-PECPATH-LP com relação à quantidade de cores para grafos esparsos

Durante os testes para grafos densos (ver Tabela 5.2 e Figuras 5.5 e 5.6), cerca de 18% das instancias tiveram problemas de falta de memória em pelo menos um dos algoritmos. Este número já era esperado em virtude do aumento considerável de vértices e arestas dos grafos gerados alocados nas estruturas de dados que foram definidas aliado a limitações de memória de máquina e arquiteturas de API's utilizadas neste trabalho. Analisando a Tabela 5.2, percebemos que o aumento na densidade dos grafos teve um aumento considerável nos tempos de execução do algoritmo MIN-PECPATH (19 segundos em média). Para exemplificar, o tempo de execução do algoritmo MIN-PECPATH na instância 5 já é maior que o tempo do mesmo algoritmo para o pior caso nos grafos esparsos. Para o algoritmo MIN-PECPATH-LP, a variação dos seus tempos de execução foi pequena (3 segundos em média).

Nº	INSTÂNCIA			Função	Tempo em segundos	
	Vértices	Arestas	Cores		MIN-PECPATH	MIN-PECPATH-LP
1	500	25000	2	$n^2/10$	**0,105**	1,586
2	500	25000	127	$n^2/10$	**0,193**	2,258
3	500	50000	2	$n^2/5$	**0,523**	3,317
4	500	50000	233	$n^2/5$	**0,763**	5,064
5	500	100000	2	$2n^2/5$	**4,516**	7,404
6	500	100000	426	$2n^2/5$	**5,103**	14,381
7	1000	100000	2	$n^2/10$	**2,040**	7,650
8	1000	100000	243	$n^2/10$	**2,573**	12,041
9	1000	200000	2	$n^2/5$	**12,593**	17,327
10	1000	200000	453	$n^2/5$	mem	**33,159**
11	1000	400000	2	$2n^2/5$	57,512	**35,672**
12	1000	400000		$2n^2/5$	mem	mem
13	2000	200000	2	$n^2/20$	**6,002**	16,120
14	2000	200000	249	$n^2/20$	mem	**27,740**
15	2000	400000	2	$n^2/10$	**30,205**	40,320
16	2000	400000	462	$n^2/10$	mem	mem
17	2000	800000	2	$n^2/5$	129,488	**60,608**
18	2000	800000	886	$n^2/5$	mem	mem

Tabela 5.2: Comparação entre os tempos de execução dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para instancias de grafos densos

Das 15 instancias válidas, o algoritmo MIN-PECPATH foi superior em 11 delas (os melhores resultados de cada instancia estão em negrito na Tabela). Das quatro instâncias em que o algoritmo MIN-PECPATH perdeu para o algoritmo MIN-PECPATH-LP, em duas delas (instâncias 10 e 14) não foi possível o término da execução do algoritmo MIN-PECPATH por falta de memória. Isto ocorreu devido à uma limitação da alocação do número de vértices gerados do grafo não-colorido na estrutura definida na API do SGB, que é limitada a 500.000 vértices. As instâncias 12 e 16 também não foram representadas pelo mesmo motivo.

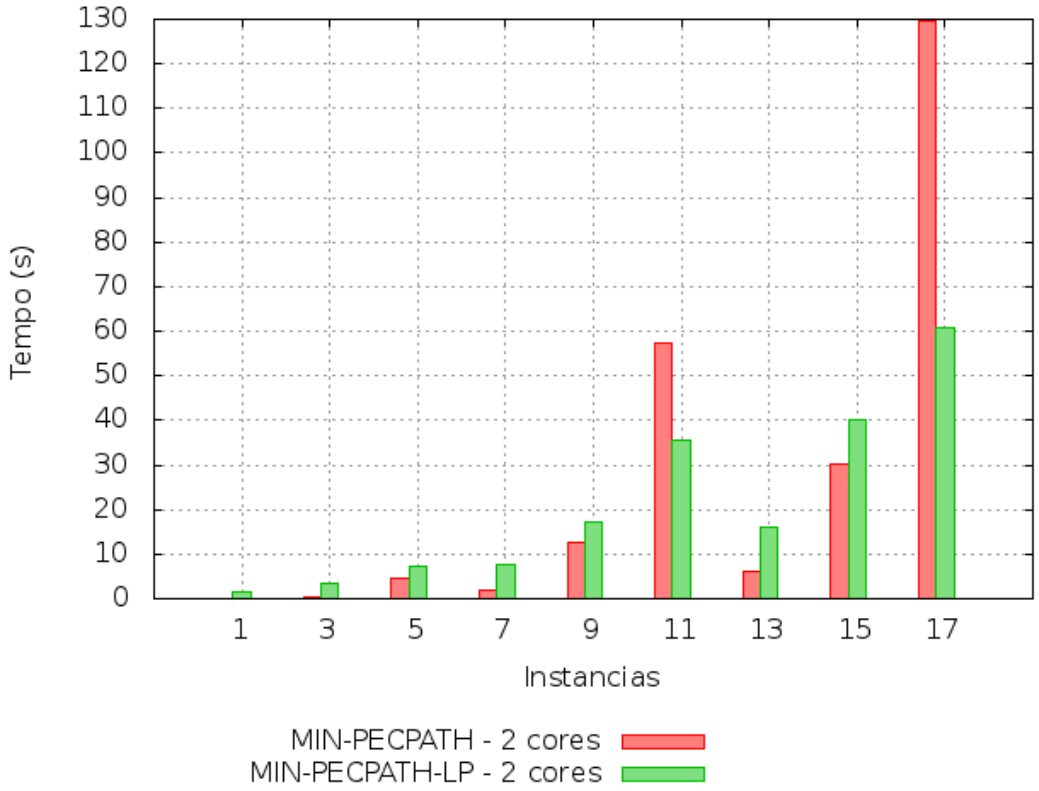

Figura 5.5: Desempenho dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para grafos densos com 2 cores nas arestas

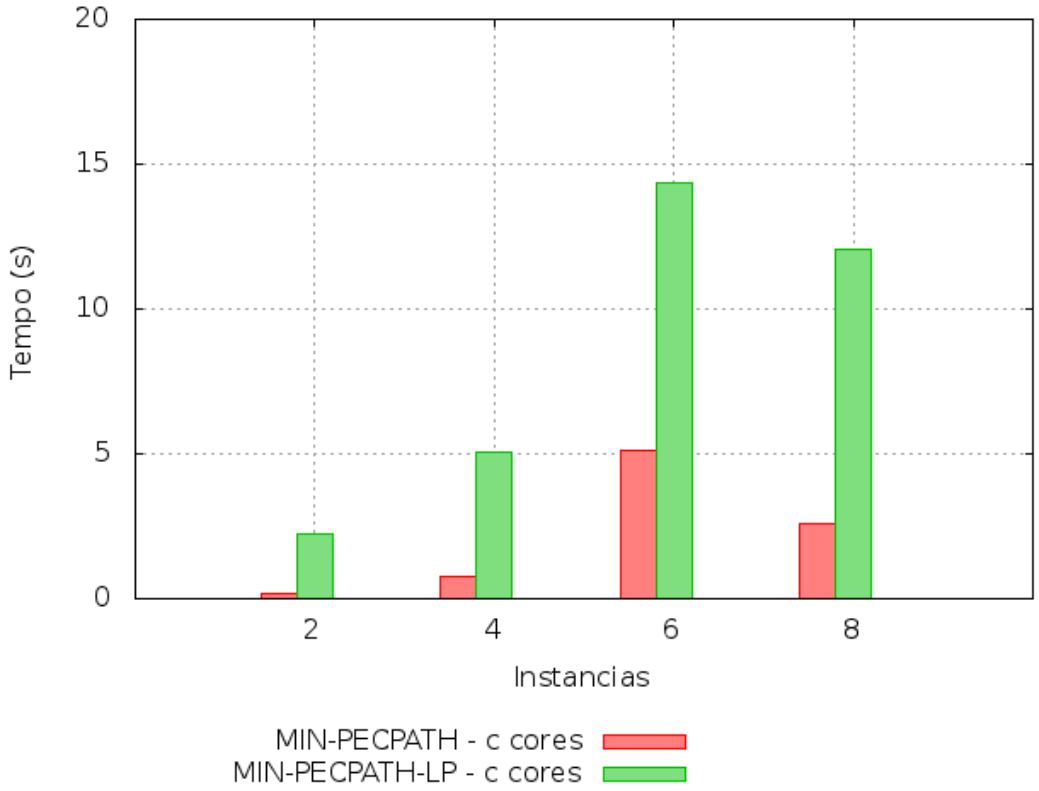

Figura 5.6: Desempenho dos algoritmos MIN-PECPATH e MIN-PECPATH-LP para grafos densos com c cores nas arestas

Um outro aspecto interessante foi a variação (Gap) dos tempos de execução dos algoritmos MIN-PECPATH e MIN-PECPATH-LP com o aumento de cores para um mesmo número de vértices e arestas em relação às instancias dos grafos esparsos. O cálculo da variação (Gap) é dado por $Gap = (T^c_{n,m} - T^2_{n,m})/T^2_{n,m}$ onde $T^c_{n,m}$ é o tempo gasto para execução do Algoritmo para n vértices, m arestas e c cores e $T^2_{n,m}$ é o tempo gasto para a execução do Algoritmo para n vértices, m arestas e 2 cores. Quanto mais próximo de zero for este valor, significa que o tempo de execução do Algoritmo para c cores com o mesmo número de vértices e arestas teve uma variação mínima em relação ao menor tempo considerado para 2 cores. Nas 8 primeiras instâncias referentes a 4 grupos de vértices e arestas, o variação no tempo de execução do algoritmo MIN-PECPATH-LP foi superior à variação no tempo de execução do algoritmo MIN-PECPATH em 3 desses grupos (veja Figura 5.7). A média da variação do algoritmo MIN-PECPATH para estas instâncias foi $0,42$ contra $0,61$ do Algoritmo MIN-PECPATH-LP. Ou seja, para estas 8 primeiras instâncias, a variação do tempo de execução do Algoritmo MIN-PECPATH em relação ao Algoritmo MIN-PECPATH-LP foi 45% menor.

A razão para o bom desempenho do algoritmo MIN-PECPATH nas 8 primeiras ins-

tancias em relação ao algoritmo MIN-PECPATH-LP é que para 2 cores, a densidade do grafo não-colorido resultante (apos a aplicação da construção XP-Gadget) praticamente permanece idêntica à densidade do grafo colorido original enquanto que para o limite superior de cores c, o número de vértices do grafo não-colorido resultante aumenta muito em relação ao número de arestas, fazendo com que a densidade deste grafo seja mínima e nessas condições o algoritmo MIN-PECPATH tem o desempenho parecido ao aplicado a um grafo esparso.

A conclusão que tiramos é que para instancias pequenas de grafos densos (grafos com até 1000 vértices com densidade de até 20%) o algoritmo MIN-PECPATH apresenta um melhor desempenho. Para densidades de até 40% o desempenho fica restrito ao limite inferior de cores e para densidades acima de 40%, o desempenho do algoritmo MIN-PECPATH piora bastante a ponto de inviabilizá-lo. Na média geral para grafos densos, o algoritmo MIN-PECPATH obteve uma média de 19,35 s conta 18,98 s do algoritmo MIN-PECPATH-LP.

Na média geral para os dois tipos de grafos, devido ao excelente desempenho para os grafos esparsos, o algoritmo MIN-PECPATH foi mais eficiente com o média de $9,85s$ contra $20,12s$ do algoritmo MIN-PECPATH-LP.

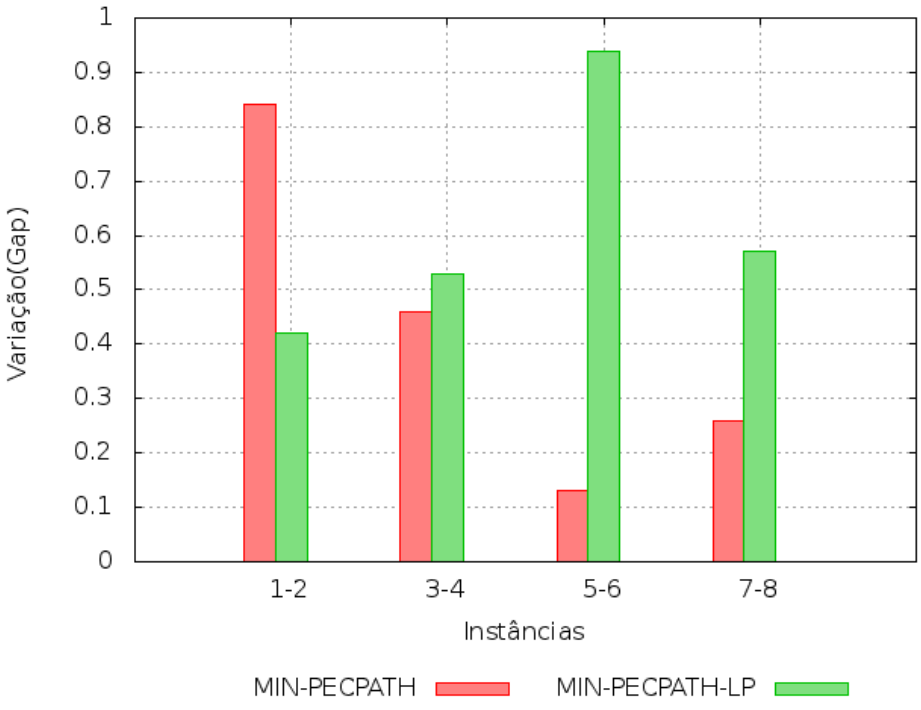

Figura 5.7: Desempenho dos algoritmos MIN-PECPATH e MIN-PECPATH-LP quanto à variação do número de cores

5.3.2 Maior $s - t$ caminho propriamente colorido

A Tabela 5.3, apresenta os resultados computacionais obtidos para o conjunto de instâncias representados por grafos esparsos sem ciclos propriamente coloridos. Cada linha da Tabela (instância) representa a média dos tempos do algoritmos correspondentes a 10 execuções.

Podemos verificar na Tabela 5.3 que em todas as instancias, o tempo de execução do algoritmo MAX-PECPATH foi superior ao do algoritmo MAX-PECPATH-LP (Veja gráfico ilustrado na Figura 5.8. Em média, o algoritmo MAX-PECPATH foi 3 vezes mais rápido. Considerando como parâmetro o menor tempo de execução dos algoritmos, calculamos a variação desses tempos (GAP) para cada algoritmo em relação ao maior tempo encontrado em todas as instâncias e em relação ao maior tempo encontrado dentro de cada grupo de vértices. A fórmula para o cálculo é $GAP = (T_M - T_m)/T_m$, onde T_M é o maior Tempo considerado e T_m é o menor tempo considerado. Em relação ao maior tempo encontrado considerando todas as instâncias, o Algoritmo MAX-PECPATH teve um GAP maior em relação ao algoritmo MAX-PECPATH-LP (5,7 contra 5), no entanto, considerando variações dentro de cada grupo de vértices, o algoritmo MAX-PECPATH foi mais eficiente, com destaque para o grupo com 500 vértices (instancias 4, 5 e 6), onde para um aumento de 100% no número de arestas (da instância 4 para a instância 6) houve um aumento de $16,7\%$ no tempo de execução do algoritmo MAX-PECPATH contra um aumento de 200% no tempo de execução do algoritmo MAX-PECPATH-LP (veja o gráfico ilustrado na Figura 5.9).

Nº	INSTÂNCIA			Função	Tempo em segundos	
	Vértices	Arestas	Nº de cores		MAX-PECPATH	MAX-PECPATH-LP
1	250	500	3	2n	**0,003**	0,01
2	250	750	3	3n	**0,003**	0,01
3	250	1000	3	4n	**0,003**	0,01
4	500	1000	3	2n	**0,006**	0,01
5	500	1500	3	3n	**0,006**	0,02
6	500	2000	3	4n	**0,007**	0,03
7	1000	2000	3	2n	**0,013**	0,02
8	1000	3000	3	3n	**0,016**	0,04
9	1000	4000	3	4n	**0,02**	0,06

Tabela 5.3: Comparação entre os tempos de execução dos algoritmos para instâncias de grafos esparsos sem ciclos propriamente coloridos

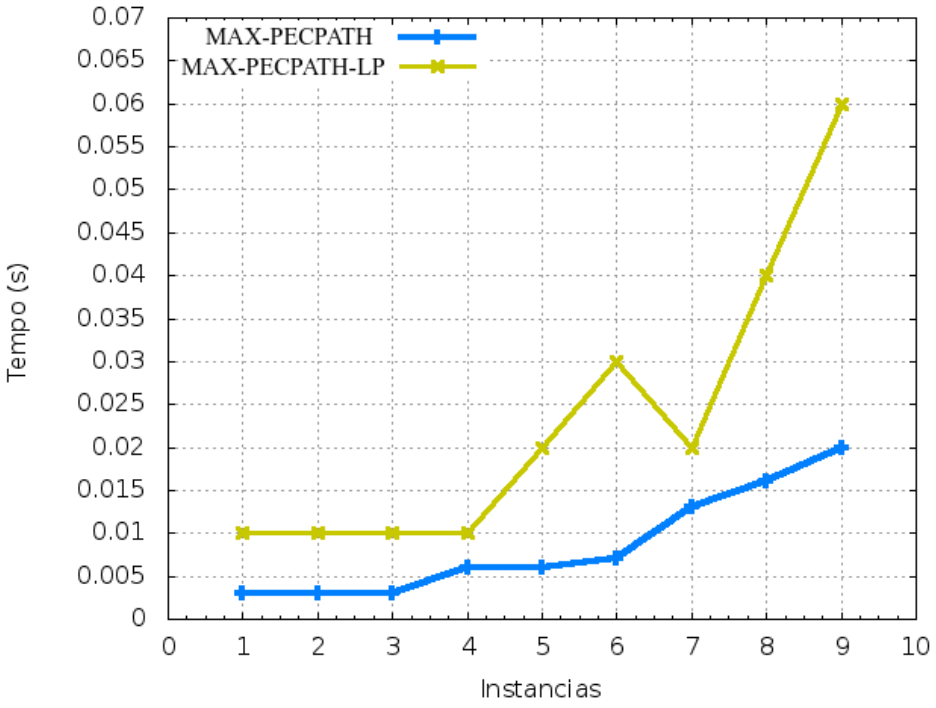

Figura 5.8: Desempenho dos algoritmos MAX-PECPATH e MAX-PECPATH-LP para grafos esparsos sem ciclos propriamente coloridos

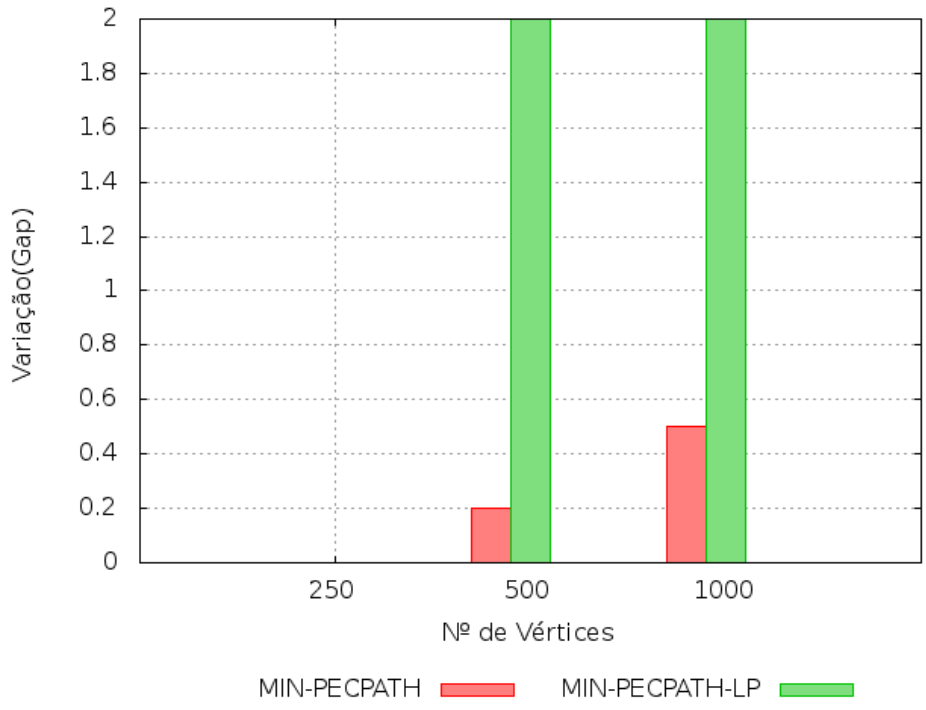

Figura 5.9: Variações nos tempos de execução por número de vértices dos Algoritmos MAX-PECPATH e MAX-PECPATH-LP para grafos esparsos sem ciclos propriamente coloridos

A Tabela 5.4 apresenta os resultados computacionais obtidos para o conjunto de instâncias representados por grafos esparsos G^c quaisquer. Para grafos quaisquer, a probabilidade de se encontrar ciclos propriamente coloridos na solução utilizando o algoritmo MAX-PECPATH é muito grande. Em nossos testes, isto ocorreu em $92,2\%$ dos casos e em virtude disto, só foi possível computar a média dos tempos de execução do algoritmo MAX-PECPATH-LP. Para os $7,8\%$ dos casos ocorridos nas instâncias marcadas com um asterisco '*' (veja a coluna Nº da Tabela 5.4), foram computados os tempos de execução dos algoritmos MAX-PECPATH e MAX-PECPATH-LP. Na Tabela 5.5, temos a relação detalhada das instâncias onde foi possível comparar os tempos de execução dos dois algoritmos.

Nº	INSTÂNCIA			Função	Tempo em segundos	
	Vértices	Arestas	Nº de cores		MAX-PECPATH	MAX-PECPATH-LP
*1	250	500	3	2n	-	13,91
2	250	750	3	3n	-	27,76
*3	250	1000	3	4n	-	15,16
*4	500	1000	3	2n	-	49,92
5	500	1500	3	3n	-	65,86
*6	500	2000	3	4n	-	341,90
7	1000	2000	3	2n	-	258,55
8	1000	3000	3	3n	-	155,51
*9	1000	4000	3	4n	-	1634,27

Tabela 5.4: Tempos de execução do Algoritmo MAX-PECPATH-LP para grafos esparsos G^c quaisquer

Instância	No. da execução	Tempo de segundos
		MAX-PECPATH
1	#5	0,003
1	#9	0,003
1	#10	0,003
3	#7	0,004
4	#3	0,008
6	#7	0,011
9	#10	0,023

Tabela 5.5: Tempos de execução do Algoritmo MAX-PECPATH para grafos esparsos G^c quaisquer em execuções onde não houve ciclos propriamente coloridos na solução encontrada

Por se tratar de um problema NP-Difícil quando a solução encontrada apresenta ciclos propriamente coloridos, o Algoritmo MAX-PECPATH-LP leva um tempo considerável para determinar a solução ótima para o maior $s - t$ caminho propriamente colorido em um grafo G^c qualquer, se comparado com o mesmo algoritmo para o caso particular com grafos G^c sem ciclos propriamente coloridos e este tempo tende a ser maior quanto maior for o tamanho das instâncias (Número de vértices, Número de arestas e número de cores). O gráfico ilustrado na Figura 5.10 mostra uma comparação das variações dos tempos de execução do Algoritmo MAX-PECPATH-LP para os dois tipos de grafos.

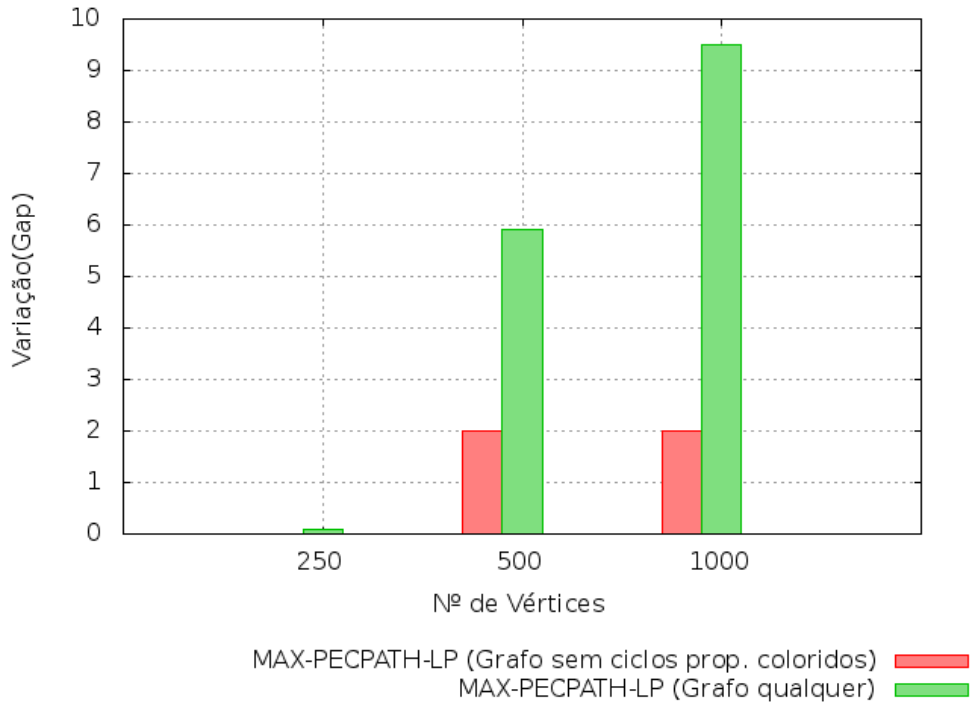

Figura 5.10: Variações nos tempos de execução do Algoritmo MAX-PECPATH-LP para os dois tipos de grafos

Os tempos das execuções obtidos pelo algoritmo MAX-PECPATH na Tabela 5.5 não foram considerados no cálculo da média dos tempos da instâncias. Uma vez que o Algoritmo original para determinação do maior $s-t$ caminho propriamente colorido para um grafo G^c qualquer (Veja Algoritmo 9 na Seção 3.7) vai executar apenas um dos dois algoritmos (MAX-PECPATH ou MAX-PECPATH-LP), podemos concluir pelos resultados apresentados que este Algoritmo além de resolver o problema do maior $s-t$ caminho propriamente colorido, ainda pode resolvê-lo em um ótimo tempo computacional em situações onde não há ciclos propriamente coloridos na solução encontrada pelo algoritmo MAX-PECPATH.

Neste capítulo, foram apresentados os resultados experimentais obtidos para a resolução do problema do menor $s-t$ caminho propriamente colorido em grafos arbitrários para dois conjuntos de cores, através dos algoritmos MIN-PECPATH e MIN-PECPATH-LP e do maior $s-t$ caminho propriamente colorido para um grafo G^c qualquer com 3 cores nas arestas através dos algoritmos MAX-PECPATH e MAX-PECPATH-LP com o objetivo de avaliar a eficiência desses algoritmos quanto ao tempo computacional despendido. As conclusões finais a partir destas analises descritas são apresentadas no Capítulo 6, assim como sugestões de trabalhos futuros.

Capítulo 6

Conclusões e Trabalhos Futuros

Vários problemas diferentes relacionados a $s-t$ caminhos e trilhas propriamente coloridos foram considerados. A implementação de construções P-Gadgets gerando grafos não-coloridos equivalentes aos grafos coloridos originais, aliados a algoritmos de emparelhamento de custo mínimo foram cruciais para a resolução dos problemas abordados e formaram a base das implementações desenvolvidas.

Neste trabalho, foram implementados algoritmos para o problema da obtenção de $s-t$ caminhos e trilhas propriamente coloridos propostos na literatura. O problema da obtenção da menor $s-t$ trilha propriamente colorida foi reduzido ao problema do menor $s-t$ caminho propriamente colorido, através de uma transformação de um grafo G^c em um grafo H^c equivalente. Sabemos da literatura que o problema de encontrar o maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c arbitrário é um problema NP-Difícil [1]. Como a obtenção de um $s-t$ caminho/trilha sem ciclos/trilhas fechadas propriamente coloridos é um problema polinomial [1], a implementação da verificação de um Ciclo/Trilha Fechada Propriamente Colorido(a) em um Grafo G^c com c Arestas Coloridas, foi importante e necessária para obtermos o maior $s-t$ caminho/trilha propriamente colorido em um grafo G^c com esta restrição. Para resolução do maior$s-t$ caminho/trilha propriamente colorido em um grafo G^c qualquer, implementamos um algoritmo que verifica inicialmente a existência ou não de ciclos propriamente coloridos antes de resolvê-lo diretamente através de um modelo de programação linear inteira (PLI) (descrito no Capítulo 4). Para resolução completa deste problema, caso existam subciclos propriamente coloridos, implementamos uma sub-rotina que identifica sub-ciclos e os adiciona como nova restrição ao modelo de PLI descrito. Por ultimo, implementamos um algoritmo para obtenção do menor custo de recoloração de arestas de forma a encontrar um $s-t$ caminho propriamente colorido em um grafo G^c. Nesta implementação, utilizamos uma adaptação

da construção XP-Gadget para a construção dos subgrafos G_x que diminuiu a quantidade de vértices gerados comparado com a implementação original definida em [29].

Para os experimentos computacionais, foram realizados testes para o problema da obtenção do menor e maior $s-t$ caminho propriamente colorido em um grafo G^c. Para os testes da obtenção do menor $s-t$ caminho propriamente colorido, foram utilizadas instancias randômicas de grafos esparsos e grafos densos, com número de vértices variando entre 10000, 20000 e 40000 para grafos esparsos e 500, 1000 e 2000 para grafos densos. Para os testes da obtenção do maior $s-t$ caminho propriamente colorido em um grafo G^c, por se tratar de um problema NP-Difícil, foram utilizadas apenas instâncias randômicas de grafos esparsos, com número de vértices variando entre 250, 500 e 1000. Para cada quantidade de vértices, foram definidas 3 funções para obtenção da quantidade de suas arestas respectivas. No geral os resultados indicaram um bom desempenho dos algoritmos MIN-PECPATH e MAX-PECPATH. Para a obtenção do menor $s-t$ caminho propriamente colorido (considerando a média para os dois tipos de grafos testados), o algoritmo MIN-PECPATH obteve uma redução média de $51,04\%$ em relação ao tempo de execução do algoritmo MIN-PECPATH-LP. Para a obtenção do maior $s-t$ caminho propriamente colorido, considerando o caso particular para grafos sem ciclos propriamente coloridos, o algoritmo MAX-PECPATH obteve uma redução média de $65,21\%$ em relação ao tempo de execução do algoritmo MAX-PECPATH-LP. Para o caso geral, o algoritmo MAX-PECPATH-LP obteve uma média de tempo de execução de $284,76$ segundos.

Para os algoritmos de menor $s-t$ trilha propriamente colorida e maiores $s-t$ caminhos/trilhas propriamente coloridos sem ciclos/trilhas fechadas propriamente coloridas, uma vez que a construção XP-Gadget e algoritmo de emparelhamento perfeito de custo mínimo fazem parte da base destes algoritmos, acreditamos no bom desempenho destes assim como foi demonstrado para os algoritmos MIN-PECPATH e MAX-PECPATH.

Como trabalhos futuros, pretende-se criar um modelo matemático para determinação da menor e maior $s-t$ trilha propriamente colorida a fim de compararmos os resultados com os algoritmos polinomiais propostos neste trabalho. Além disso, pretende-se abordar dois problemas NP-Difíceis encontrados na literatura: custo de recoloração mínimo para determinação de 2 $s-t$ caminhos monocromáticos disjuntos por arestas em um grafo G^c com $c \geq 2$ e custo de recoloração mínimo para destruição de ciclos/trilhas fechadas propriamente coloridos em grafos G^c.

Referências

[1] Abouelaoualim, a.; Das, K.; Faria, L.; Manoussakis, Y.; Martinhon, C.; Saad, R. Paths and trails in edge-colored graphs. *Theoretical Computer Science 409*, 3 (Dec. 2008), 497–510.

[2] Arenales, M.; Armentano, V.; Morabito, R. *Pesquisa operacional: para cursos de engenharia*. 2007.

[3] Balakrishnan, V. *Schaum's outline of theory and problems of graph theory*. McGraw-Hill, 1997.

[4] Bang-Jensen, J.; Gutin, G. Alternating cycles and paths in edge-coloured multigraphs: a survey. *Discrete Mathematics 165* (1997), 39–60.

[5] Berge, C. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America 43*, 9 (1957), 842.

[6] Bondy, J. A.; Murty, U. S. R. *Graph theory with applications*, vol. 290. Macmillan London, 1976.

[7] Chou, W.; MANOUSSAKIS, Y.; MEGALAKAKI, O.; Spyratos, M.; TUZA, Z. Paths through fixed verticles in edge-colored graphs. *Mathématiques, informatique et sciences humaines*, 127 (1994), 49–58.

[8] Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America* (1954), 393–410.

[9] Diestel, R. *Graph Theory*. Springer, 2005.

[10] Edmonds, J. Paths, trees, and flowers. *Canadian Journal of mathematics 17*, 3 (1965), 449–467.

[11] Edmonds, J.; Johnson, E.; Lockhart, S. Blossom i: a computer code for the matching problem. *IBM TJ Watson Research Center, Yorktown Heights, New York* (1969).

[12] Even, S.; Kariv, O. An $o(n^{2.5})$ algorithm for maximum matching in general graphs. In *Foundations of Computer Science, 1975., 16th Annual Symposium on* (1975), IEEE, pp. 100–112.

[13] Feofiloff, P.; Kohayakawa, Y.; Wakabayashi, Y. Uma introdução sucinta à teoria dos grafos. *IME-USP. Disponivel em: http://www. ime. usp. br/pf/teoriadosgrafos/texto/TeoriaDosGrafos. pdf Acesso em 22*, 11 (2009), 2009.

[14] Galbiati, G. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics 156*, 18 (2008), 3494–3497.

[15] Gamvros, I. Satellite network, design, optimization, and management.

[16] Goldbarg, M. C.; Luna, H. P. L. *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier, 2005.

[17] Gourvès, L.; Lyra, A.; Martinhon, C.; Monnot, J. The minimum reload s–t path, trail and walk problems. *Discrete Applied Mathematics 158*, 13 (2010), 1404–1417.

[18] Gourves, L.; Lyra, A. R. d.; Martinhon, C. A.; Monnot, J. On paths, trails and closed trails in edge-colored graphs. *Discrete Mathematics & Theoretical Computer Science 14*, 2 (2012), 57–74.

[19] Grossman, J. W.; Häggkvist, R. Alternating cycles in edge-partitioned graphs. *Journal of Combinatorial Theory, Series B 34*, 1 (1983), 77–81.

[20] Gutin, G.; Kim, E. J. Properly coloured cycles and paths: results and open problems. In *Graph Theory, Computational Intelligence and Thought*. Springer, 2009, pp. 200–208.

[21] Harju, T. Lecture notes on graph theory. *University of Turku–Finland* (2007).

[22] Hopcroft, J. E.; Karp, R. M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing 2*, 4 (1973), 225–231.

[23] Johnson, D. S.; McGeoch, C. C. *Network Flows and Matching: First DIMACS Implementation Challenge*. American Mathematical Society, Boston, MA, USA, 1993.

[24] Knuth, D. E. *The Stanford GraphBase: a platform for combinatorial computing*. ACM, New York, NY, USA, 1993.

[25] Kolmogorov, V. Blossom v: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation 1*, 1 (2009), 43–67.

[26] Kuhn, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly 2*, 1-2 (1955), 83–97.

[27] Lyra, A. R. d. *On Paths and Trails in Edge-Colored Graphs and Digraphs*. Tese de Doutorado, Universidade Federal Fluminense, 2009.

[28] Manoussakis, Y. Alternating paths in edge-colored complete graphs. *Discrete Applied Mathematics 56*, 2 (1995), 297–309.

[29] Martinhon, C. A.; Faria, L. The edge-recoloring cost of paths and cycles in edge-colored graphs and digraphs. In *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*. Springer, 2013, pp. 231–240.

[30] Micali, S.; Vazirani, V. V. An $o(\sqrt{|v|}.|e|)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on* (1980), IEEE, pp. 17–27.

[31] Pevzner, P. *Computational molecular biology: an algorithmic approach*, vol. 1. MIT press Cambridge, 2000.

[32] Sedgewick, R. *Algorithms in C, Part 5: Graph Algorithms*. Addison-Wesley, 2001.

[33] Szachniuk, M.; Cola, M. C. D.; Felici, G.; Blazewicz, J.; Werra, D. d. Optimal pathway reconstruction on 3d nmr maps. Submitted.

[34] Szeider, S. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics 126*, 2 (2003), 261–273.

[35] Tutte, W. T. The factorization of linear graphs. *Journal of the London Mathematical Society 1*, 2 (1947), 107–111.

[36] Wang, Y. Efficient ldpc code based secret sharing schemes and private data storage in cloud without encryption. Tech. rep., Technical report, UNC Charlotte, 2012.

[37] Wang, Y.; Desmedt, Y. Edge-colored graphs with applications to homogeneous faults. *Information Processing Letters 111*, 13 (2011), 634–641.

[38] Wirth, H.-C.; Steffan, J. Reload cost problems: minimum diameter spanning tree. *Discrete Applied Mathematics 113*, 1 (2001), 73–85.

[39] Yeo, A. A note on alternating cycles in edge-coloured graphs. *Journal of combinatorial theory. Series B 69*, 2 (1997), 222–225.