Uma Implementação eficiente para problemas de caminhos e ciclos em grafos com arestas coloridas

Rogério da Silva Batista
Instituto Federal do Piauí
Teresina – PI – Brazil
rogeriobatista@ifpi.edu.br

Carlos Alberto de Jesus Martinhon
Universidade Federal Fluminense
Niterói – RJ – Brazil
mart@ic.uff.br

RESUMO

Nos últimos anos uma grande quantidade de problemas importantes tem sido modelada por grafos com cores nas arestas. Dado um grafo G^c com $c \geq 2$ cores nas arestas, $s, t \in V(G^c)$, dizemos que um $s-t$ caminho/trilha é propriamente colorido quando suas arestas sucessivas possuírem cores distintas. Neste trabalho, apresentamos um modelo de Programação Linear Inteira (PLI), implementamos e testamos algoritmos para obtenção do maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer. Quando G^c não contém ciclos propriamente coloridos o problema se torna polinomial. Neste caso, desenvolvemos um gerador de instancias sem ciclos propriamente coloridos e apresentamos um algoritmo polinomial utilizando *gadgets* eficientes, transformando o problema do maior $s-t$ caminho/trilha em G^c em um problema de emparelhamento perfeito de custo máximo em grafo associado G não colorido.

PALAVRAS CHAVE. Grafos com Cores nas Arestas, Caminhos e Trilhas Propriamente Coloridos, Programação Linear Inteira, Algoritmos e Otimização Combinatória.

ABSTRACT

In the last few years, a great number of important problems have been modeled by edge-colored graphs. Given an edge-colored graph G^c with $c \geq 2$ colors and $s, t \in V(G^c)$ vertices, we define properly edge-colored $s-t$ paths/trails when any two sucessive edges differ in color. In this work, an Integer Linear Programming (ILP) formulation is presented and an algorithm is implemented and tested for finding the longest properly edge-colored path/trail in an arbitrary graph G^c . When G^c contains no properly edge-colored cycles, this problem becomes polynomial. In this case, we developed a generator of instances with no properly edge-colored cycles and present a polynomial algorithm using efficient gadgets, reducing the longest $s-t$ properly edge-colored path/trail problem in G^c to a maximum weighted perfect matching problem in a non-colored graph G.

KEYWORDS. Edge-Colored Graphs, Properly Edge-Colored Paths/Trails, Integer Linear Programming, Algorithms and Combinatorial Optimization.

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 (Abouelaoualim et al., 2008; Bang-Jensen and Gutin, 1997; Gourvès et al., 2010; Gutin and Kim, 2009; Lyra, 2009). Podemos encontrar aplicações em qualidade de serviço em redes de comunicações (Wang and Desmedt, 2011), ciências sociais (Chou et al., 1994), conectividade e transporte (Gamvros, 2006) entre outras.

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 objetivo deste trabalho é propor uma metodologia para resolução de problemas relacionados a grafos com cores nas arestas, mais especificamente na determinação do maior $s-t$ caminho propriamente colorido.

Este trabalho está organizado da seguinte forma: na Seção 2, introduzimos conceitos importantes a respeito da teoria de caminhos e ciclos, mais especificamente sobre caminhos e ciclos propriamente coloridos, além de apresentar alguns trabalhos relacionados. Na Seção 3, introduzimos os conceitos sobre construções de P-Gadgets necessários para construção de grafos não-coloridos associados a um grafo G^c e descreveremos o algoritmo implementado para a obtenção do maior $s-t$ caminho propriamente colorido em um grafo G^c sem ciclos propriamente coloridos. Apresentamos também um modelo de PLI proposto para resolução de ciclos propriamente coloridos na solução inicial encontrada. Na Seção 4, descrevemos os experimentos computacionais realizados. Por fim, a Seção 5 contém as conclusões deste trabalho juntamente com as sugestões de trabalhos futuros.

2. Fundamentação Teórica e Aplicações

Dado um grafo G^c com c cores nas arestas, 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 1 b) c)). 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 1 d)). 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 1.c) e trilhas fechadas propriamente coloridas.

O problema da determinação de $s-t$ caminhos propriamente coloridos foi inicialmente resolvido por Edmonds para um grafo com 2 cores nas arestas (Edmonds, 1965).

Manoussakis (Manoussakis, 1995) mostra que encontrar um caminho alternante ou propriamente colorido para um grafo com 2 cores nas arestas é similar a encontrar um ca-

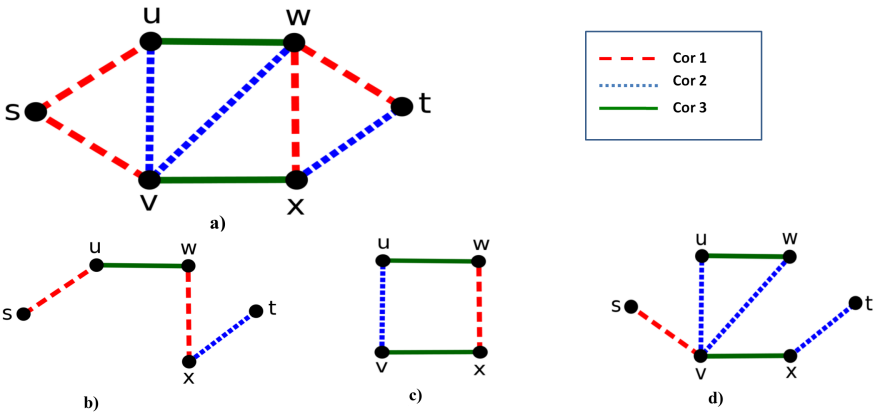

Figura 1. a)Um Grafo com 3-cores nas arestas; b): $s-t$ caminho propriamente colorido; c) ciclo propriamente colorido d)$s-t$ trilha propriamente colorida;

minho 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. Mais tarde Szeider (Szeider, 2003) 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 (Szeider, 2003) no seguinte teorema:

Teorema 1 *(Szeider, 2003) Seja s e t dois vértices em um grafo G^c com $c \geq 2$ cores nas arestas. 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 (Edmonds, 1965) 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, abordamos os principais tipos de construções P-Gadgets (responsáveis pela geração de grafos não-coloridos a partir de um grafo G^c) presentes na literatura.

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 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 (Yeo, 1997) generalizou o problema proposto por Grossman and Häggkvist (Grossman and Häggkvist, 1983) para grafos com 2 ou mais cores nas arestas, através do seguinte teorema:

Teorema 2 *(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.*

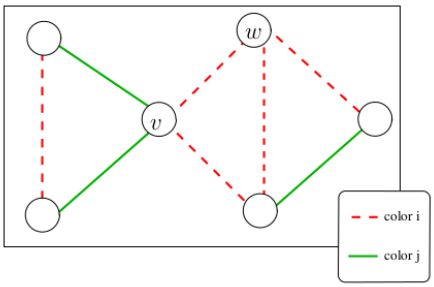

Figura 2. Um grafo com 2 vértices de corte separando cores: v e w (Lyra, 2009)

Este resultado proposto no Teorema 2 pode ser utilizado para se determinar se um grafo G^c possui ou não ciclos propriamente coloridos. Yeo (Yeo, 1997) 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 contenha vértices com pelo menos duas arestas incidentes com cores diferentes (neste caso, G^c contém ciclos propriamente coloridos).

2.1. Alguns Trabalhos Relacionados

Em Gourvès et al. (Gourvès et al., 2010), 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:

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

Em outras palavras, o objetivo é encontrar um caminho/trilha/passeio ρ entre s, t com custo de reconexão mínimo.

Em Martinhon & Faria (Martinhon and Faria, 2013), 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 (Martinhon and Faria, 2013), esta propriedade se refere especificamente a caminhos/trilhas/ciclos propriamente coloridos ou monocromáticos em grafos ou digrafos.

3. Algoritmos Propostos

Inicialmente, para a resolução de problemas de caminhos e ciclos em grafos com arestas coloridas, analisamos construções **P-Gadgets** existentes na literatura.

As construções P-Gadgets (Gutin and Kim, 2009) 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) satisfazendo as propriedades descritas em (Gutin and Kim, 2009), então G_x é chamado P-Gadget.

Os três P-Gadgets mais conhecidos na literatura são SP-Gadget (Szeider, 2003), BJGP-Gadget (Bang-Jensen and Gutin, 1997) e XP-Gadget (Gutin and Kim, 2009).

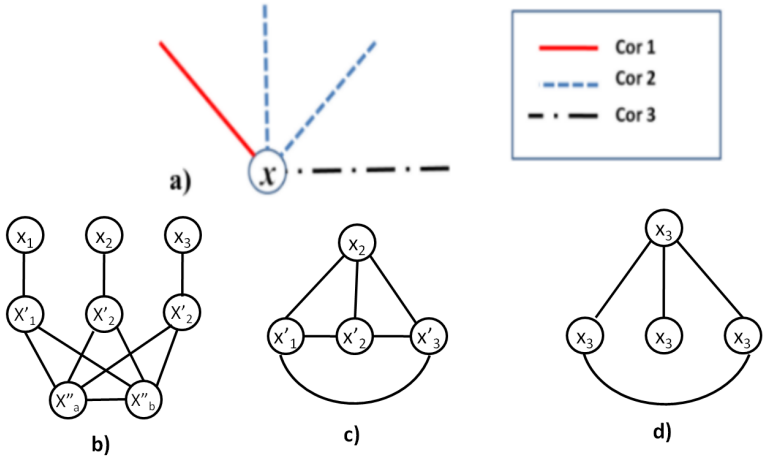

Figura 3. a)Exemplo da representação de um vértice x com 3-cores nas arestas; b) Representação G_x utilizando a construção SP-Gadget;c) Representação G_x utilizando a construção BJGP-Gadget; d) Representação G_x utilizando a construção XP-Gadget

A construção **XP-Gadget** foi implementada neste trabalho, por apresentar um menor número de vértices e arestas gerados no grafo não-colorido G em relação às outras construções ($2z - 2$ vértices e $3z - 5$ arestas). Mais detalhes sobre a implementação deste algoritmo podem ser encontrados em (Batista, 2014).

3.1. Obtenção do maior $s - t$ caminho 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 (Abouelaoualim et al., 2008). Descreveremos nesta seção a implementação de um algoritmo aplicado a um 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.

Algoritmo 1 Obter o maior $s-t$ caminho propriamente colorido de um grafo G^c qualquer

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

1: **Inicio**
2: **Enquanto** $\exists\, v$ monocromatico $\in G^c$, onde $v \notin \{s,t\}$ **faça**
3: $G^c = G^c \setminus \{v\}$
4: **Fim-enquanto**
5: $G \leftarrow$ XP-Gadget(G^c)
6: Defina um conjunto de arestas $E' = \bigcup_{x \in W} E(G_x)$
7: **Para todo** $pq \in E(G) \backslash E'$ **faça**
8: $custo(pq) \leftarrow -1$;
9: **Fim-para**
10: **Para todo** $pq \in E'$ **faça**
11: $custo(pq) \leftarrow 0$;
12: **Fim-para**
13: **Se** $\nexists$ emparelhamento perfeito de custo mínimo no grafo G **então**
14: Não existe um $s-t$ caminho propriamente colorido em G^c
15: **senão**
16: Seja G^c_{sol} o grafo solução com um $s-t$ caminho do grafo G^c
17: **Se** G^c_{sol} não contém ciclos propriamente coloridos **então**
18: Retorne G^c_{sol}
19: **senão**
20: Aplique um modelo de PLI em G^c
21: **Fim-se**
22: **Fim-se**
23: **Fim**

Na primeira parte do **Algoritmo 1**, procuramos simplificar o grafo original G^c retirando todos os vértices $v \in V(G^c) \setminus \{s,t\}$ monocromáticos. O Grafo G^c resultante deste processo torna a sua construção XP-Gadget menos custosa uma vez que estamos apenas interessados em caminhos propriamente coloridos. Na linha 5, chamamos a construção XP-Gadget (implementada em (Batista, 2014)) para gerar um grafo não-colorido G a partir de um Grafo de entrada G^c com c cores nas arestas. A seguir, na linha 6, 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 7 a 12 do **Algoritmo 1**, 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.

Em seguida, aplicamos um algoritmo de emparelhamento perfeito de custo mínimo proposto em (Kolmogorov, 2009) (linha 13) 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 (linha

18). Isto significa que não existe um 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.

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

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

O modelo de PLI é descrito a seguir:

$$\textit{Maximizar} \sum_{(i,j)\in E(G^c),\ k\in I_c} x^k_{i,j} \tag{2}$$

Sujeito a:

$$\sum_{j\in V(G^c)\ e\ k\in I_c} x^k_{sj} = 1 \tag{3}$$

$$\sum_{i\in V(G^c)\ e\ k\in I_c} x^k_{it} = 1 \tag{4}$$

$$\sum_{k\in I_c}\sum_{i\in V(G^c)} x^k_{ij} - \sum_{k'\in I_c}\sum_{l\in V(G^c)} x^{k'}_{jl} = 0; \qquad \forall j \in V(G^c)\setminus\{s,t\} \tag{5}$$

$$\sum_{i\in V(G^c)} x^k_{ij} + \sum_{l\in V(G^c)} x^k_{jl} <= 1, \qquad \forall j \in V(G^c)\setminus\{s,t\}, k \in I_c \tag{6}$$

$$\sum_{k\in I_c}\sum_{i\in V(G^c)} x^k_{ij} <= 1,\ \forall j \in V(G^c)\setminus\{s\} \tag{7}$$

$$\sum_{k \in I_c} \sum_{l \in v(G^c)} x_{jl}^k <= 1, \ \forall j \in V(G^c) \setminus \{t\} \tag{8}$$

$$\sum_{k \in I_c} \sum_{(i,j) \in E(S)} x_{ij}^k <= |S| \ - \ 1, \ \forall S \subset V(G^c) \setminus \{s,t\}, \ 2 \leqslant |S| \leqslant n-2. \tag{9}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall i,j \in V(G^c), \ k \in I_c. \tag{10}$$

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.

A função objetivo representada na Equação 2, 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 3 e 4 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 5 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 6 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 7 e 8 garantem que um nó só poderá ser visitado no máximo uma vez. E finalmente a última restrição, correspondente à Equação 9, 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 (Dantzig et al., 1954) 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 10 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.

4. Experimentos Computacionais

Nos testes da determinação do maior $s-t$ caminho propriamente colorido para um grafo G^c qualquer, comparamos o desempenho do **Algoritmo 1**, considerando os ca-

sos 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 1 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 1 onde o maior $s-t$ caminho propriamente colorido é resolvido através da implementação do modelo matemático definido na Seção 3.1, 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.

4.1. Ambiente de Execução

Os algoritmos testados neste trabalho foram implementados na linguagem C++ e compiladas com o g++ versão 4.6.3. Além disso, o modelo matemático implementado no algoritmo MAX-PECPATH-LP foi construído 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.

4.2. Instâncias Utilizadas

As instâncias utilizadas neste trabalho foram criadas randomicamente para grafos esparsos. 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.

Por se tratar de um problema NP-Difícil e para viabilizar nossos testes, utilizamos apenas grafos esparsos com 3 cores nas arestas. 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.

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. Para o caso particular de grafos sem ciclos propriamente coloridos, a instância do grafo foi gerada 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 de acordo com o teorema proposto por Yeo em (Yeo, 1997).

4.3. Maior $s - t$ caminho propriamente colorido

A Tabela 1, apresenta os resultados computacionais obtidos para o conjunto de instâncias representados por grafos esparsos sem ciclos propriamente coloridos. Podemos verificar na Tabela 1 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 4). 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.

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 1. Comparação entre os tempos de execução dos algoritmos para instâncias de grafos esparsos sem ciclos propriamente coloridos

A Tabela 2 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.

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).

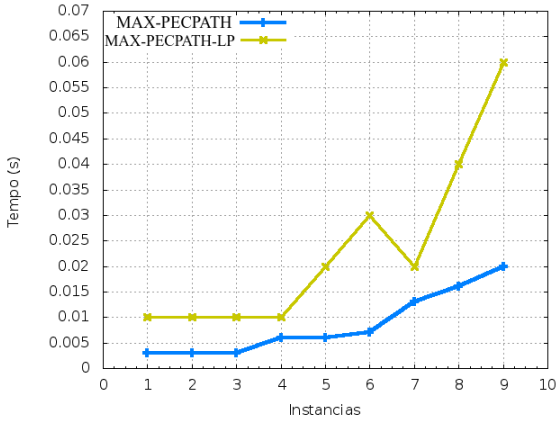

Figura 4. Desempenho dos algoritmos MAX-PECPATH e MAX-PECPATH-LP para grafos esparsos sem ciclos propriamente coloridos

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 2. Tempos de execução do Algoritmo MAX-PECPATH-LP para grafos esparsos G^c quaisquer

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 baixo tempo computacional em situações onde não há ciclos propriamente coloridos na solução encontrada pelo algoritmo MAX-PECPATH.

5. Conclusões e trabalhos futuros

Apresentamos e implementamos um modelo de PLI para o problema do maior $s-t$ caminho propriamente colorido em um grafo G^c qualquer. A construção e implementação de P-Gadgets gerando grafos G não-coloridos, aliados a algoritmos de emparelhamento de custo máximo possibilitou a determinação polinomial de $s-t$ caminhos mais longos em grafos sem ciclos propriamente coloridos.

No geral os resultados indicaram um bom desempenho do algoritmo MAX-PECPATH. Considerando o caso particular para grafos sem ciclos propriamente coloridos, o algoritmo obteve uma redução aproximada de 65% 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.

Como trabalhos futuros, pretende-se criar um modelo matemático para determina-

ção da menor e maior $s-t$ trilha propriamente colorida a fim de compararmos os resultados com o algoritmo polinomial baseado em emparelhamento perfeito de custo mínimo. Além disso, pretende-se investigar 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

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

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

Batista, R. d. S. (2014). Implementações eficientes para problemas de caminhos e ciclos em grafos com arestas coloridas. Master's thesis, Universidade Federal Fluminense.

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

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

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

Gamvros, I. (2006). Satellite network, design, optimization, and management.

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

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

Gutin, G. and Kim, E. J. (2009). Properly coloured cycles and paths: results and open problems. In *Graph Theory, Computational Intelligence and Thought*, pages 200–208. Springer.

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

Lyra, A. R. d. (2009). *On Paths and Trails in Edge-Colored Graphs and Digraphs*. PhD thesis, Universidade Federal Fluminense.

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

Martinhon, C. A. and Faria, L. (2013). 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*, pages 231–240. Springer.

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

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

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