UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

IGOR DE MORAES SAMPAIO

Sobre o problema de caminhos tropicais em grafos: formulação, heurística e resultados experimentais

São Paulo
2023

IGOR DE MORAES SAMPAIO

Sobre o problema de caminhos tropicais em grafos: formulação, heurística e resultados experimentais

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 19 de abril de 2023. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Profa. Dra. Karla Roberta Pereira Sampaio Lima

São Paulo

2023

Ficha catalográfica elaborada pela Biblioteca da Escola de Artes, Ciências e Humanidades,
com os dados inseridos pelo(a) autor(a)
Brenda Fontes Malheiros de Castro CRB 8-7012; Sandra Tokarevicz CRB 8-4936

Moraes Sampaio, Igor de
Sobre o problema de caminhos tropicais em grafos: formulação, heurística e resultados experimentais / Igor de Moraes Sampaio; orientadora, Karla Roberta Pereira Sampaio Lima. -- São Paulo, 2023.
90 p: il.

Dissertacao (Mestrado em Ciencias) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, 2023.
Versão corrigida

1. Coloração de grafos. 2. Caminho tropical. 3. Programação linear inteira. 4. Heurística. I. Lima, Karla Roberta Pereira Sampaio, orient. II. Título.

Dissertação de autoria de Igor de Moraes Sampaio, sob o título **"Sobre o problema de caminhos tropicais em grafos: formulação, heurística e resultados experimentais"**, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 19 de abril de 2023 pela comissão julgadora constituída pelos doutores:

Profa. Dra. Karla Roberta Pereira Sampaio Lima
Universidade de São Paulo
Presidente

Profa. Dra. Márcia Rodrigues Cappelle Santana
Universidade Federal de Goiás

Prof. Dr. José de Jesús Pérez-Alcázar
Universidade de São Paulo

Dedico esta dissertação de mestrado a Deus, meus pais, família, professores e amigos, que me deram amor, apoio e inspiração ao longo de toda a minha jornada acadêmica.

Agradecimentos

Gostaria de agradecer a todos que me apoiaram ao longo desta jornada. Sem a ajuda e incentivo de vocês, este trabalho não seria possível. Agradeço especialmente a minha orientadora, a professora doutora Karla Roberta Pereira Sampaio Lima, por toda a orientação e paciência que teve comigo durante a elaboração deste trabalho. Suas sugestões e críticas foram fundamentais para o aprimoramento do meu estudo.

Aos meus pais, Irailde Chagas de Moraes Sampaio e Edmilson Carlos Bispo Sampaio, meu amor e gratidão são infinitos. Vocês sempre me apoiaram em todas as minhas escolhas e me incentivaram a buscar meus objetivos. Agradeço também aos meus amigos e demais familiares, que sempre me encorajaram e estiveram presentes em todos os momentos.

Acima de tudo, agradeço a Deus por me dar saúde, força e sabedoria para concluir este trabalho. Sua presença em minha vida é fundamental e eu sou muito grato por todas as bênçãos que recebo diariamente.

Por fim, gostaria de agradecer aos órgãos de fomento que tornaram possível a realização deste estudo. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Além disso, este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (Proc. 423833/2018-9), pelo Programa de Apoio à Pós-Graduação (Proap)-CAPES e pela FAPESP (Proc.2021/07080-6).

Mais uma vez, meu sincero agradecimento a todos vocês que fizeram parte deste processo. O apoio e incentivo de vocês foram fundamentais para a conclusão deste trabalho.

"O importante é não parar de questionar. A curiosidade tem sua própria razão de existir."

(Albert Einstein)

Resumo

SAMPAIO, Igor de Moraes. **Sobre o problema de caminhos tropicais em grafos: formulação, heurística e resultados experimentais**. 2023. 90 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2023.

Neste trabalho, estudamos o problema do caminho tropical máximo em grafos, MTPP. Um caminho em um grafo colorido nos vértices é dito ser *tropical* se os vértices do caminho possuem as cores usadas pela coloração dos vértices grafo. Para o problema MTPP é dado um grafo e uma coloração de seus vértices, e o objetivo é encontrar um caminho cuja coloração use o maior número possível de cores desta coloração. A motivação para estudar o MTPP surge da constatação de que, até o início desta pesquisa, não havia na literatura abordagens baseadas em modelos de programação linear inteira, nem algoritmos heurísticos para o problema de interesse. Sabe-se que o MTPP é NP-difícil para grafos em geral, grafos direcionados acíclicos, grafos cacto e grafos de intervalo. Nesta pesquisa, foi desenvolvida uma modelagem para o problema MTPP como um problema de programação linear inteira para grafos simples e uma simplificação do modelo proposto para DAGs também é apresentada. Uma formulação similar é apresentada para uma segunda versão de otimização do problema em que o objetivo é encontrar um caminho tropical cuja soma dos pesos das arestas seja o menor possível. A contribuição principal desta pesquisa consiste na construção de uma heurística de tempo polinomial para o MTPP que juntamente com o modelo de PLI foi possível avaliar o desempenho de ambos por meio de experimentos computacionais em instâncias aleatórias e reais.

Palavras-chaves: Coloração de grafos. Caminho tropical. Programação linear inteira. Heurística.

Abstract

SAMPAIO, Igor de Moraes. **On the problem of tropical paths in graphs: formulation, heuristics and experimental results**. 2023. 90 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2023.

In this work, we studied the problem of maximum tropical path in graphs, MTPP. A path in a vertex-colored graph is said to be *tropical* if the vertices of the path have the colors used in the vertex coloring of the graph. For the MTPP problem, a graph and a coloring of its vertices are given, and the goal is to find a path whose coloring uses the largest possible number of colors from this coloring. The motivation for studying MTPP arises from the observation that, until the beginning of this research, there were no approaches based on integer linear programming models or heuristic algorithms for the problem of interest in the literature. It is known that MTPP is NP-hard for general graphs, directed acyclic graphs, cactus graphs, and interval graphs. In this research, a modeling for the MTPP problem as an integer linear programming problem for simple graphs was developed, and a simplification of the proposed model for DAGs is also presented. A similar formulation is presented for a second optimization version of the problem, in which the objective is to find a tropical path whose sum of edge weights is as small as possible. The main contribution of this research consists of constructing a polynomial-time heuristic for MTPP, which, together with the ILP model, made it possible to evaluate the performance of both through computational experiments on random and real instances.

Keywords: Graph coloring. Tropical path. Integer linear programming. Heuristics.

Lista de figuras

Lista de algoritmos

Lista de tabelas

Lista de abreviaturas e siglas

DAGs	Directed Acyclic Graphs
MTPP	Maximum Tropical Path Problem
PL	Programação Linear
PLI	Programação Linear Inteira
PPI	Protein-Protein Interaction
STPP	Shortest Tropical Path Problem
TPP	Tropical Path Problem

Lista de símbolos

$\mathbb{Z}$	Conjunto de números inteiros
$\mathbb{R}$	Conjunto de números reais
$\sum$	Somatório
$\emptyset$	Conjunto vazio
$\forall$	Para todo
$\in$	Pertence
$\neq$	Diferente de
$\subseteq$	Subconjunto de
$\dot{\cup}$	União disjunta de conjuntos
O	Notação Big O
Ω	Notação Omega
Θ	Notação Theta

Sumário

1 Introdução

O avanço computacional e tecnológico permite que problemas reais muito complexos possam ser investigados de maneira mais eficiente e em muitos casos resolvidos. A cada dia mais cientistas e pesquisadores são desafiados a aprender novos mecanismos para soluções de problemas computacionais.

Muitos problemas reais podem ser mapeados com intuito de investigar relações entre objetos de um determinado conjunto e alguns deles podem ser formulados computacionalmente através de um estrutura conhecida como grafos. Em linhas gerais, grafos são estruturas que consistem em um conjunto de pontos que são chamados de vértices que se relacionam através conexões conhecidas como arestas.

A teoria dos grafos é uma linha de pesquisa da ciência da computação teórica que possui vasta aplicação na modelagem de problemas reais, pois grafos podem representar e formular inúmeras estruturas de problemas de interesse prático. Um exemplo são sistemas complexos que podem ser essencialmente descritos por redes mapeadas em grafos (NEWMAN, 2003). Tais redes descrevem os elementos do sistema por meio dos vértices, que podem ser por exemplo, átomos, proteínas, computadores ou neurônios e descrevem também as relações entre os elementos por meio das arestas. Tais relações podem ser conexões entre átomos, forças eletrostáticas, proteínas ou uma conexão entre computadores. Note que qualquer subestrutura de interesse nesta rede define um problema de alta complexidade (VALVERDE; SOLÉ, 2009).

Os objetos e as relações de um problema estudado através de grafos apresentam diversas características de interesse. Uma técnica muito utilizada em pesquisas é atribuir cores às estruturas do grafo para representar e classificar uma determinada característica. Esta atribuição de cores pode ser em arestas, em vértices ou em ambos. Normalmente, a função mais utilizada é associar cores aos vértices do grafo.

Neste trabalho é investigado o problema de encontrar caminhos tropicais em grafos coloridos nos vértices. Um caminho em um grafo colorido nos vértices é dito ser tropical sempre que os vértices do caminho possuam as cores do grafo original. Potencialmente, muitas estruturas tropicais em grafos podem ser estudadas. Na literatura foram encontrados estudos sobre conjuntos dominantes tropicais, conjuntos independentes tropicais, componentes conectados tropicais, caminhos tropicais mais

curtos, entre outros (CHAPELLE *et al.*, 2014; COHEN *et al.*, 2017; D'AURIAC *et al.*, 2016; D'AURIAC *et al.*, 2018).

O problema de busca de caminhos tropicais em grafos coloridos nos vértices trata-se de uma variação do problema de ocorrência de *Motifs* em grafos. O problema de busca de motifs em grafos consiste em encontrar um subgrafo cujo conjunto de cores corresponda ao multiconjunto de cores dado como entrada. Note que para o problema de busca de caminhos tropicais, o ocorrência compreende a estrutura de um caminho e o multiconjunto de cores é o conjunto formado por todas as cores utilizadas pelo grafo de entrada.

Para simplificar, o problema de busca de caminhos tropicais em grafos coloridos nos vértices é denotado pela sigla TPP. Este trabalho estuda primordialmente uma versão de otimização do problema que é chamada de problema do caminho tropical máximo ou *Maximum Tropical Path Problem*, que será denotada pela sigla MTPP. Sabe-se que o MTPP é NP-difícil para grafos em geral, grafos direcionados acíclicos, grafos cacto e grafos de intervalo (COHEN *et al.*, 2019). O problema TPP, também possui uma versão de otimização chamada de problema do caminho tropical mínimo ou *Shortest Tropical Path Problem*, que é denotado pela sigla STPP. Ambas versões são formalmente definidas na seção a seguir.

1.1 Definição do problema

Seja $G = (V, E)$ um grafo simples, não direcionado, onde V representa o conjunto de vértices e E o conjunto de arestas de G. Dado um conjunto de cores $C = \{c_0, \ldots, c_{k-1}\}$, $G^C = (V, E)$ denota um grafo colorido nos vértices pelas cores de C. Em alguns casos G^C também pode conter pesos nas arestas, ou seja, cada aresta e é associado a um número real $w(e)$.

Formalmente, de acordo com as definições acima, um caminho P de G^C é dito ser *tropical* se, e somente se, cada cor de C aparece pelo menos uma vez em P.

Problema 1 *Problema do Caminho Tropical (TPP - Tropical Path Problem). Dado um grafo conexo $G = (V, E)$ e uma função $C : V \to C$, encontrar, caso exista, um caminho tropical P em G.*

Algumas observações devem ser feitas sobre o problema TPP. A coloração dos vértices do grafo é uma coloração arbitrária, ou seja, uma simples atribuição de cores aos

vértices do grafo sem quaisquer restrições. São considerados apenas caminhos simples, ou seja, nenhum vértice é visitado mais de uma vez.

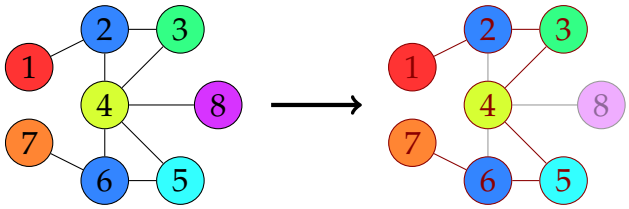
O TPP possui duas versões de otimização: o problema do caminho tropical máximo (MTPP) e o problema do caminho tropical mínimo (STPP), formalmente descritos a seguir.

> *Problema do caminho tropical máximo* (MTPP). Dado um grafo colorido nos vértices $G^C = (G, C)$, o problema consiste em encontrar um caminho P em G com o maior número de cores de C.

> *Problema do caminho tropical mínimo* (STPP). Dado um grafo ponderado colorido nos vértices $G^C_w = (G, C, w)$ e dois vértices s e t, o problema consiste em encontrar, se existir, um caminho P de s a t, que contenha todas as cores de C e a soma dos pesos das arestas de P seja o menor peso possível.

A Figura 1 exemplifica uma instância do problema MTPP, e permite observar algumas características do problema.

Figura 1 – Uma solução ótima é representada pelo caminho em destaque com 7 vértices e 6 cores. O grafo de entrada tem 7 cores.



Fonte – Igor Sampaio, 2023

Observe que no MTPP, para uma dada instância, se a solução do problema P for um caminho tropical, então a solução é certamente ótima, além disso, se cada vértice tiver uma cor diferente, estará lidando com o problema clássico do caminho hamiltoniano. Note que para o STPP um solução ótima nem sempre existe, já o MTPP uma solução ótima sempre existe. Em uma solução para o MTPP é permitido que o caminho possua vértices com cores repetidas. O foco principal deste trabalho foi investigar a versão de otimização MTPP e obter resultados na linha de programação linear inteira e algoritmos heurísticos.

1.2 *Aplicações e motivação*

O problema TPP está fundamentado na teoria de grafos, mas especificamente em grafos coloridos nos vértices. Os grafos são objetos de estudo há centenas de anos, por serem estruturas que modelam com muita qualidade diversos problemas no mundo real, desde o clássico problema das sete pontes de Königsberg apresentado por Euler (1736), que é considerado o primeiro resultado da teoria dos grafos, até o famoso problema do caixeiro viajante que já foi estudado por diversos pesquisadores.

Problemas em grafos coloridos nos vértices, assim como o TPP, têm inúmeras aplicações em diversas áreas da ciência, em especial na biologia e bioinformática. Como exemplo, um site pode ser modelado como um grafo colorido nos vértices onde a cor de um vértice representa o conteúdo da página correspondente (vermelho para matemática, amarelo para física, etc.) (BRUCKNER *et al.*, 2013). Podemos citar aplicações em uma série de problemas, como no agendamento de voos de aeronaves (MARX, 2004), prevenção de conflitos (um exemplo seria a separação de produtos com risco de explosão) e atribuição de frequências de rádio (BONDY; MURTY, 2008). Na biologia e bioinformática podemos citar também aplicações no mapeamento físico de DNA (FELLOWS; HALLETT; WAREHAM, 1993) em filogenia perfeita (MCMORRIS; WARNOW; WIMER, 1994), aplicações em pipeline de alinhamento de múltiplas sequências (COREL; PITSCHI; MORGENSTERN, 2010), análise filogenética (CHOR *et al.*, 2007), entre outros.

Como antes mencionado o TPP deriva do problema da ocorrência de Motifs em grafos. Tais problemas foram amplamente estudados nos últimos 20 anos, e notavelmente tem aplicações importantes em análise de redes metabólicas. Lacroix, Fernandes e Sagot (2006) investigaram o problema da ocorrência de Motifs no contexto de análise de redes metabólicas, em que o conjunto de reações envolvidas na síntese e degradação de determinadas moléculas são representados através de uma rede e os Motifs correspondem aos módulos onde essa rede pode ser decomposta, possibilitando a interpretação das funções que cada um desses módulos desempenha no metabolismo celular. Eles introduziram um algoritmo para esse fim que apresentou bons resultados na prática. Em redes regulatórias de genes (SHEN-ORR *et al.*, 2002) e redes de interação proteína-proteína chamadas de PPI (SCOTT *et al.*, 2006), (FREIRE; LIMA; ROJAS, 2018) investigaram este problema para a estrutura específica das árvores. Nesse caso, as redes representam

certas interações biológicas. Eles propuseram modelos de programação linear inteira e apresentaram resultados experimentais para instâncias reais de redes PPI.

Atualmente, pesquisadores de diferentes áreas estão aprendendo a lidar com problemas de natureza discreta, classicamente conhecidos como problemas NP-difíceis. Sabe-se que poucas são as técnicas que apresentam resultados práticos significativos, em termos de solução exata, para problemas dessa magnitude. De fato, a complexidade envolvida é um fator limitante crucial para o avanço computacional. Por outro lado, existem diversas perspectivas de estudo que tem-se mostrado como poderosas ferramentas para tratar tais problemas.

A motivação para estudar o MTPP surge da constatação de que, até o início desta pesquisa, não havia na literatura abordagens baseadas em modelagem computacional para o problema de interesse.

1.3 Contribuições

Todos os resultados deste trabalho proporcionam uma abordagem inédita na literatura para o problema proposto, cujo impacto é comprovado pelas motivações apresentadas. Assim, as seguintes contribuições são destacadas no presente trabalho:

- uma formulação do problema MTPP como um modelo de programação inteira;
- simplificação do modelo de programação inteira para o problema MTPP para o caso específico de DAGs;
- algoritmo heurístico de tempo polinomial para o problema MTPP em grafos;
- resultados experimentais para instâncias reais e aleatórias.

1.4 Organização do texto

Além deste capítulo de introdução, este trabalho possui outros seis capítulos. Cada capítulo é detalhado a seguir:

- O Capítulo 1 é a introdução do trabalho e apresenta todo o contexto do estudo. Ele descreve o problema estudado, bem como a sua aplicabilidade, motivação e contribuições desta dissertação.

- O Capítulo 2 é o capítulo onde é detalhado os conceitos computacionais utilizados no trabalho. Ele explora principalmente os conceitos e terminologias utilizados nas áreas de teoria dos grafos e teoria poliédrica.
- O Capítulo 3 é uma revisão literária que traz uma análise de trabalhos semelhantes ou tecnicamente importantes para o desenvolvimento deste estudo. Ele menciona os principais resultados conhecidos para todas as variantes de problemas com estruturas tropicais, com enfoque nos resultados conhecidos para o problema TPP. Também descreve de forma mais aprofundada tais resultados.
- O Capítulo 4 apresenta os resultados de modelos de programação linear inteira para o problema. Este capítulo inclui uma formulação linear inteira do problema MTPP, bem como uma simplificação do modelo de programação linear inteira para o caso específico de grafos dirigidos acíclicos (DAGs) e uma formulação linear inteira do problema STPP.
- O Capítulo 5 apresenta a heurística desenvolvida para o problema MTPP. Ele começa com um algoritmo que extrai um grafo cacto a partir de um grafos simples para utilizar um algoritmo heurístico para a versão do MTPP aplicado a grafos cactos. Em seguida, é apresentado o algoritmo heurístico.
- O Capítulo 6 realiza uma análise empírica, que é o principal capítulo de validação dos resultados produzidos pela pesquisa. Através da análise empírica dos resultados de experimentos computacionais realizados sobre todas as contribuições deste trabalho, este capítulo traz importantes conclusões e características do modelo PLI e da heurística apresentados.
- O Capítulo 7 é a conclusão do trabalho e apresenta as considerações finais. Todas as conclusões encontradas são discutidas, e são levantadas questões em aberto que podem ser exploradas em trabalhos futuros.

2 Conceitos

Neste capítulo são definidos alguns conceitos básicos fundamentais relacionados a este trabalho. No decorrer do trabalho utilizamos notações e definições consistentes com as que são geralmente usadas em teoria dos grafos, em complexidade computacional e em combinatória poliédrica.

2.1 Teoria dos grafos

A teoria dos grafos estuda relações entre objetos de um determinado conjunto. As estruturas utilizadas para estudar essas relações são chamadas de *grafos*. Um grafo G é um par ordenado $(V(G), E(G))$, onde $V(G)$ é um conjunto finito não vazio de objetos denominados *vértices* e $E(G)$ é um conjunto de pares não-ordenados e distintos de vértices $V(G)$ chamados *arestas*. Por simplicidade tais conjuntos serão mencionados por V e E e o número de vértices e arestas de G por n e m, respectivamente.

Por definição e é uma aresta formada por vértices extremos u e v dito adjacentes. Por simplicidade, denotamos uma aresta $\{u, v\}$ por uv. O *grau* de um vértice v é o número de arestas incidentes em v.

Uma aresta que possui o mesmo vértice em suas extremidades, ou seja, possui extremidades idênticas é chamada de *loop* ou *laço*. Duas ou mais arestas com o mesmo par de extremidades são consideradas *arestas paralelas*. Quando em um grafo, para toda aresta não há distinção entre vértice de origem (fonte) ou vértice de destino (alvo) temos um *grafo não-direcionado*, ou *grafo não-dirigido*, ou ainda *grafo não-orientado*.

Observamos que na maioria dos textos, o objeto que definimos como grafo é chamado de grafos simples. Em outras palavras, um *grafo simples* é um grafo não direcionado que não possui laços nem arestas paralelas. Quando um grafo não é simples, o grafo é dito ser um *multigrafo*.

Um *grafo direcionado* G é um par ordenado $(V(G), A(G))$, no qual $A(G)$ é um conjunto de pares ordenados e distintos de vértices $V(G)$ chamados *arcos*. Se a é uma aresta direcionada $uv \in A$, então a sai de u e entra em v. O *grau de entrada* de um vértice v é a quantidade de arestas que entram em v. De modo similar, o *grau de saída* de um vértice v é a quantidade de arestas que saem de v.

Neste trabalho, é denotado por $\delta(v)$ o conjunto formado pelos vértices *vizinhos* de v, isto é, os vértices que estão nas extremidades das arestas de v. Da mesma forma $\delta^+(v)$ e $\delta^-(v)$ denotam o conjunto formado pelos vértices vizinhos de saída e entrada de v, respectivamente.

Em muitos problemas reais, algumas características adicionais são necessárias, como custos associados às arestas. Esses problemas são modelados por grafos ponderados. Para a modelagem de um *grafo ponderado* temos que para cada aresta e de G, associa-se um número real $w(e)$, que chamaremos de peso de e. Então G, junto com esses pesos em suas arestas, é chamado de *grafo ponderado* e denotado $G = (V, E, w)$.

Um grafo G' é dito ser um *subgrafo* de G se $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$ e escrevemos $G' \subseteq G$. Se $G' \subseteq G$ e G' contém todas as arestas $uv \in E(G)$ com $u, v \in V(G')$, então G' é um *subgrafo induzido* de G; dizemos que $V(G')$ induz G' em G e escrevemos $G' := G[V(G')]$. Assim, Se $W \subseteq V(G)$, então $G - W$ denota o subgrafo obtido a partir de G removendo-se W, i.e. $G - W = G[V(G) \backslash W]$. Por simplicidade, se v é um vértice de G, então $G - v$ denota o subgrafo G após a remoção de v e as arestas que incidem em v.

Um *caminho* em G é um sequência alternada $(P = u_1, e_1, u_2, e_2, \ldots, e_{n-1}, u_n)$ de vértices $u_1, u_2, \ldots, u_n$ e arestas $e_1, e_2, \ldots, e_{n-1}$, com $n \geq 1$ e $1 \leq i < n$, onde $e_i = (u_i, u_{i+1})$, u_1 e u_n são as extremidades de P e os vértices restantes são os vértices internos de P. Se u é uma extremidade de P, denotamos que P é um *u-caminho*. Um *ciclo* é um caminho em que u_1 e u_n coincidem e $n \geq 3$.

Um *grafo conexo* é um grafo em que para quaisquer pares de vértices u e v é possível estabelecer um caminho de u a v ou de v a u.

Um *grafo cacto*, ou *cacto* para abreviar, é um grafo conexo em que quaisquer dois ciclos simples têm no máximo um vértice em comum, equivalentemente, cada aresta pertence a no máximo um ciclo simples. Um cacto G no qual um vértice é definido como um raiz é chamado de *cacto enraizado*. A raiz de um cacto G é denotada por $raiz(G)$.

Grafos direcionados acíclicos ou DAG sigla do nome em inglês *"directed acyclic graph"*, são grafos direcionados sem ciclos.

Coloração de grafos

Coloração de grafos é uma função que atribui rótulos tradicionalmente chamados *cores* a elementos de um grafo. Neste trabalho os rótulos são atribuídos aos vértices do grafo e cada vértice recebe somente uma cor. Assim uma coloração dos vértices de um grafo $G = (V, E)$ é uma função $C : V \to \mathcal{C}$, onde $\mathcal{C}$ é um conjunto de cores. Convencionamos aqui se C é uma coloração, então $\mathcal{C}$ denota o conjunto de cores usadas por C.

Importante esclarecer que a coloração de grafos aqui descrita é uma simples atribuição de cores aos vértices onde não obedece nenhuma definição clássica de coloração de grafos entendida em teoria de grafos como a coloração própria em que dois vértices adjacentes não podem ter a mesma cor (WEST *et al.*, 2001).

Uma coloração dos vértices de um grafo é geralmente definida por uma característica que desejamos estudar e que está associada aos objetos que se referem aos vértices. Embora qualquer conjunto de elementos podem representar as cores de uma coloração, neste trabalho os números naturais são empregados para representar tais cores. Um grafo colorido é um par (G, C) que consiste em um grafo G e uma coloração C de G.

2.2 *Programação linear inteira*

Um problema de programação linear (PL) consiste em um problema de maximizar ou minimizar uma função objetivo sujeita a um número finito de restrições lineares. Estas restrições podem ser igualdades e/ou desigualdades. Problemas com essa estrutura surgem em diversas situações reais, por esse motivo a programação linear vem sendo utilizada em larga escala tanto teoricamente quanto na prática (THIE; KEOUGH, 2011).

Um tipo especial de modelo de programação linear são os problemas de programação linear inteira (PLI) ou apenas programação inteira (PI) onde algumas ou todas as variáveis do problema pertencem ao conjunto dos números inteiros. Enquanto PL's podem ser resolvidos em tempo polinomial, em PLI's, a adição de uma restrição ao modelo, em que suas variáveis possuem valores inteiros torna o problema em geral NP-Difícil (CORMEN *et al.*, 2009).

Caso todas as variáveis do problema sejam inteiras, o modelo é chamado de programação inteira pura. No caso em que algumas variáveis não possuam restrições de integralidade, o modelo é chamado de programação inteira mista e o caso em que todas as variáveis possuem valores binários 0,1 é chamado programação inteira binária (WOLSEY, 2020). Resolver programas inteiros é uma tarefa NP-difícil em geral.

Um problema de PLI pode ser definido como: dados dois vetores $c \in R^m$ e $b \in R^n$, e uma matriz $A \in R^{n \times m}$, encontrar um vetor $x \in N^m$ que maximize (ou minimize) cx, tal que $Ax \leq b$ (ou $Ax \geq b$, no caso de maximização). Um modelo de programação linear inteira de maximização, por exemplo, é da seguinte forma:

$$\begin{aligned} \text{maximizar} \quad & cx \\ \text{sujeito a} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

Observando esse exemplo, podemos notar que $Ax \leq b$ é um sistema de inequações (ou restrições) lineares. A função objetivo do modelo é a função linear cx. Qualquer vetor x que satisfaça as restrições é uma solução viável para o modelo e esta solução é considerada ótima se o valor da sua função objetivo for maior que qualquer valor de uma solução viável para o modelo.

2.3 Teoria dos poliedros

Dado um subconjunto $S \subseteq \mathbb{R}^n$, um vetor $x \in R^n$ é uma combinação linear dos vetores $x_1, x_2, \ldots, x_t \in \mathbb{R}^n$, se

$$x = \sum_{i=1}^{t} \alpha_i x_i$$

para algum $\alpha = (\alpha_1, \ldots, \alpha_t) \in \mathbb{R}^t$. Uma tal combinação linear é chamada *afim*, se $\alpha_1 + \cdots + \alpha_t = 1$; *cônica*, se $\alpha_1, \ldots, \alpha_t \geq 0$; convexa, se for afim e cônica.

Para um conjunto não-vazio $S \subseteq \mathbb{R}^n$ denotamos por lin(S) o fecho linear dos elementos de S; isto é, o conjunto de todos os vetores que são combinação linear de um número finito de vetores de S. Analogamente, definimos os fechos afim, cônico e convexo, denotados por afim(S), cone(S) e conv(S), considerando a combinação correspondente. Se S for o conjunto vazio, então lin(S) := cone(S) := 0 e conv(S) = afim(S) := $\emptyset$. Dizemos que $S \subseteq \mathbb{R}^n$ é um subespaço linear se S = lin(S). Analogamente, definimos subespaço afim, cone e conjunto convexo. Se A é uma matriz, então cada um dos conjuntos

lin(A), afim(A), cone(A) e conv(A) são definidos analogamente, considerando-se os vetores-coluna de A.

Um conjunto $S \subseteq \mathbb{R}^n$ é *linearmente independente* se para qualquer subconjunto finito $\{x_1, \ldots, x_t\}$ de S, sempre que $\sum_{i=1}^{t} \alpha_i x_i = 0$, onde $\alpha_i \in \mathbb{R}$, temos que $\alpha_1 = \cdots = \alpha_t = 0$. Dizemos que S é *afim-independente* se para qualquer subconjunto finito $\{x_1, \ldots, x_t\}$ de S,$\sum_{i=1}^{t} \alpha_i x_i = 0$, onde $\alpha_i \in \mathbb{R}$ e $\sum_{i=1}^{t} \alpha_i = 0$, implica $\alpha_1 = \cdots = \alpha_t = 0$. Um conjunto S é *linearmente dependente* se ele não for linearmente independente. Definimos afim-dependente de forma análoga.

Para $S \subseteq \mathbb{R}^n$, o *posto* de S, denotado por posto(S), é a cardinalidade de um maior subconjunto de S que é linearmente independente. Analogamente, definimos o *posto afim* de S, denotado por posto-afim(S), como sendo a cardinalidade de um maior conjunto afim independente contido em S. Observe que 0 é afim-independente, mas não é linearmente independente. Por outro lado, se S é um conjunto linearmente independente, então S é afim-independente. Prova-se que para todo $S \subseteq \mathbb{R}^n$, se $0 \in$ afim(S), então posto-afim(S) = posto(S) + 1; e se $0 \notin$ afim(S), então posto-afim(S) = posto(S).

O *posto de uma matriz* A, denotado por posto(A), é o posto do conjunto de vetores-coluna de A, que prova-se ser igual ao posto do conjunto de vetores-linha de A. Uma matriz $A \in \mathbb{R}^{m \times n}$ tem posto-linha completo se posto(A) = m, e tem posto-coluna completo se posto(A) = n.

Para $S \subseteq \mathbb{R}^n$, a *dimensão* dim(S) de S é definida como dim(S) := posto-afim(S) - 1.Dizemos que S tem dimensão plena se dim(S) = n.

Um *poliedro* $\mathcal{P} \subseteq \mathbb{R}^n$ é um conjunto de vetores que satisfazem um número finito de desigualdades lineares; isto é

$$\mathcal{P} = \{x \in \mathbb{R}^n | Ax \leq b\},$$

onde $A \in \mathbb{R}^{m \times n}$ é uma matriz e $b \in \mathbb{R}^m$ é um vetor. Um poliedro $\mathcal{P}$ definido por uma matriz A e por um vetor b é denotado por $\mathcal{P}(A, b)$.

Um poliedro $P(A, b)$ é um poliedro P definido por uma matriz A e por um vetor b. Um *politopo* é um poliedro limitado, isso significa que existe um real α tal que todo ponto $x \in P$ satisfaz $||x|| \leq \alpha$.

Um poliedro é chamado *semi-espaço* se $a \in \mathbb{R}^n$ e $\alpha \in \mathbb{R}$ então o poliedro é definido por: $\{x \in \mathbb{R}^n | a^T x \leq \alpha\}$.

Um conjunto $\{x \in \mathbb{R}^n | a^T x \leq \alpha\}$, onde a é não nulo, é chamado *hiperplano*. O hiperplano é um conjunto de pontos que divide o espaço em dois.

$P(A, b)$ é uma interseção de um número finito de semi-espaços, pois cada uma das linhas da matriz A juntamente com a correspondente entrada do vetor b define um semi-espaço.

Resultados e outros conceitos sobre poliedros, podem ser encontrados no texto de Ferreira e Wakabayashi (1996).

2.4 *Complexidade computacional*

Ao analisar um algoritmo, um dos principais pontos a se analisar é o tempo de computação necessário para encontrar uma solução; os principais recursos medidos para classificar a complexidade de um problema são o tempo e o armazenamento, porém outras medidas de complexidade também podem ser usadas.

Uma classe de complexidade importante é *NP*, que significa "tempo polinomial não determinístico". Note que no decorrer do trabalho trataremos de um problema inserido na classe de problemas *NP-difícil*. Um problema Q é considerado *NP-difícil* se todos os problemas D em *NP* forem redutíveis a Q em tempo polinomial. Ou seja, existem dois algoritmos polinomiais, de modo que o primeiro produz uma instância $J(I)$ de Q para qualquer instância I de D, e o segundo produz a resposta correta para I dada uma solução para $J(I)$. Podemos dizer informalmente que o problema é pelo menos tão difícil quanto os problemas mais difíceis em *NP* e que o problema não pode ser resolvido em tempo polinomial.

Recomendamos que para um maior aprofundamento na definição formal da teoria de complexidade computacional e da classe de problemas *NP-difícil* o leitor veja os livros de Arora e Barak (2009) e de Conforti, Cornuéjols e Zambelli (2014).

3 Revisão literária

Este capítulo apresenta uma revisão da literatura sobre problemas relacionados à busca de caminhos tropicais em grafos. São abordadas as principais contribuições do trabalho de Cohen *et al.* (2019) acerca da dicotomia da complexidade do MTPP. Esse trabalho apresenta resultados relevantes para o problema MTPP, especialmente para DAGs e grafos cacto. As provas apresentadas por Cohen *et al.* (2019) sobre a NP-dificuldade do problema MTPP para esses tipos de grafos são particularmente relevantes para este trabalho.

Motifs

O conceito de padrões de cores ou padrões coloridos usados na bioinformática está relacionado ao conceito de caminhos tropicais. Para compreender o estado da arte do estudo de Motifs, foram buscados inicialmente artigos que realizassem uma revisão sistemática, categorizando as principais abordagens, técnicas, vantagens, desvantagens e outras características relevantes. Ademais, foi adicionada uma breve revisão de alguns trabalhos que apresentaram conhecimentos importantes para o entendimento do estudo de Motifs em relação ao problema TPP.

- O artigo de Sandve e Drabløs (2006) apresenta uma revisão dos métodos de descoberta de Motifs no DNA, utilizando uma estrutura bem definida que integra todos os elementos relevantes.
- Em Makolo (2015), é realizada uma análise comparativa dos últimos desenvolvimentos em algoritmos de descoberta de Motifs e é proposto um algoritmo baseado em uma abordagem combinatória de padrões e estatística.
- Já o artigo de Hashim, Mabrouk e Al-Atabany (2019) apresenta uma classificação geral dos algoritmos de descoberta de Motifs, com novas subcategorias que facilitam a construção de um algoritmo de descoberta de Motifs bem-sucedido.

A seguir estão descritos, de forma sucinta, os trabalhos que foram julgados importantes para este estudo por relacionarem conceitos de Motifs e caminhos ou Motifs e modelos PLI.

- Scott *et al.* (2006) adaptou e estendeu técnicas eficientes para encontrar caminhos em grafos para o problema de identificar caminhos em redes de interação de proteínas. Eles apresentaram algoritmos de tempo linear para encontrar caminhos em redes sob várias restrições biologicamente motivadas. Essa metodologia foi aplicada para buscar vias proteicas na rede de interação proteína-proteína de levedura. O artigo demonstra que o algoritmo é capaz de reconstruir vias de sinalização conhecidas e identificar caminhos funcionalmente enriquecidos de maneira não supervisionada.
- Costa (2007) apresenta o conceito de caminho aleatório e o aplica para a investigação de propriedades estruturais de redes complexas e como meio para estimar o caminho mais longo.
- Joshi *et al.* (2010) apresentou um plugin Cytoscape para estudar a resposta celular a perturbações de fatores de transcrição, integrando dados de expressão perturbacional com redes de transcrição, proteína-proteína e fosforilação.
- Fellows *et al.* (2011) estudou o problema de encontrar ocorrências de Motifs em grafos coloridos por vértices. Esse problema não está interessado na estrutura real da ocorrência do padrão, apenas exige que preserve o requisito topológico básico de conectividade.
- Freire, Lima e Rojas (2018) investigaram o problema de busca de Motifs com número mínimo de componentes conexos para a estrutura específica das árvores. Nesse caso, as redes representam certas interações biológicas. Eles propuseram modelos de programação linear inteira e apresentaram resultados experimentais para instâncias reais de redes PPI.
- Thejaswi, Gionis e Lauri (2020) estudou uma família de problemas de detecção de padrões em grafos temporais coloridos por vértices. Em particular, dado um grafo temporal colorido por vértice e um multiconjunto de cores como uma consulta, o objetivo é buscar caminhos temporais no grafo que contenham as cores especificadas na consulta.

Caminhos em Grafos

Encontrar um caminho específico em um grafo é um dos problemas fundamentais na teoria dos grafos. Isso se deve ao fato de que o clássico problema das sete pontes de Königsberg, apresentado por Euler (1736), considerado o primeiro resultado da teoria dos grafos que consiste em encontrar um caminho em um grafo.

Um dos problemas mais famosos em teoria de grafos que busca encontrar um caminho é o problema do caixeiro viajante, que já foi amplamente estudado por diversos pesquisadores. Existe um caso especial desse problema, conhecido como o problema do ciclo hamiltoniano. Um caminho hamiltoniano é um caminho que passa por todos os vértices de um grafo, sem repetir nenhum deles. Próximo ao problema do caminho hamiltoniano temos o problema do caminho mais longo, que consiste em encontrar um caminho simples de comprimento máximo em um grafo.

Os três problemas citados acima, desde seus respectivos surgimentos, vêm sendo amplamente estudados por diversos pesquisadores. Na literatura, é possível encontrar inúmeros trabalhos com abordagens baseadas em modelagem computacional e geração de algoritmos, tanto exatos para classes de grafos específicas quanto heurísticas, visto que os três problemas são NP-difíceis.

Ao observar as definições das versões do problema TPP, é possível notar que no caso especial em que cada vértice tem uma cor distinta, o problema MTPP se reduz ao problema do caminho mais longo. Além disso, o STPP também se reduz ao problema do caminho mais longo, sempre que cada vértice tem uma cor distinta e todos os pesos das arestas são iguais.

O problema do caminho mais longo tem sido recentemente investigado e diversos trabalhos relacionados podem ser citados, como o trabalho de Rezende e Wakabayashi (2015). Sabe-se que, para qualquer constante $\epsilon > 0$, não é possível aproximar o caminho mais longo em um grafo geral por uma razão constante em tempo polinomial, a menos que **P** = **NP**. Também é sabido que não é possível aproximar o caminho mais longo em um grafo geral por uma razão $2^{O(log^{1-\epsilon}n)}$, a menos que **NP** esteja contido dentro de um tempo determinístico quase polinomial (KARGER; MOTWANI; RAMKUMAR, 1997). Porém, o problema do caminho mais longo pode ser resolvido em tempo polinomial para várias classes especiais de grafos, como grafos acíclicos direcionados (DAGs),

árvores, grafos de blocos, grafos biconvexos intervalares, entre outros (IOANNIDOU; MERTZIOS; NIKOLOPOULOS, 2009; UEHARA; UNO, 2004; UEHARA; VALIENTE, 2007; MARKOV *et al.*, 2012).

No que diz respeito à busca por caminhos coloridos em grafos, atualmente, é comum definir um caminho colorido como um caminho em que vértices adjacentes possuem cores distintas, o que é conhecido como coloração própria. No entanto, no TPP, dois vértices adjacentes podem ter a mesma cor na solução.

Para analisar o estado da arte, serão apresentados e discutidos alguns trabalhos selecionados relacionados a caminhos coloridos.

Normalmente, os estudos sobre grafos coloridos nos vértices estão intimamente relacionados à coloração própria. Fung (1989) investigou a existência de caminhos coloridos que incluem todas as cores em colorações próprias ótimas de vértices de grafos. Em relação a esse problema, o trabalho de Li (2001) estudou uma versão em que os caminhos começam em qualquer dado vértice, com o objetivo de determinar a existência desses caminhos. Embora esses problemas sejam semelhantes ao TPP, a característica de coloração própria dos vértices diferencia-os e essa diferença torna-se mais evidente ao analisar as provas simples apresentadas por Lin (2007) em relação aos resultados desses trabalhos, pois essas provas não se aplicam ao TPP.

Assim como muitos trabalhos definem grafos coloridos nos vértices como grafos com uma coloração própria, muitos trabalhos também definem um caminho colorido como um caminho que possui uma coloração própria de seus vértices. O trabalho de Akbari, Liaghat e Nikzad (2011) utiliza tal conceito de caminho colorido no estudo que verifica se existe uma coloração para um grafo com um caminho colorido que possua todas as cores do grafo e inicie a partir de cada vértice do grafo.

Um trabalho importante que estuda uma das variações de caminhos coloridos é o de Kowalik e Lauri (2016), que analisa o problema de encontrar caminhos coloridos, dentre outros problemas relacionados a subconjuntos coloridos. Esse trabalho define um caminho colorido de acordo com a definição introduzida pelo clássico trabalho de Alon, Yuster e Zwick (1995), que define que dado um grafo $G = (V, E)$ e uma função de coloração arbitrária $c : V \to [k]$, um caminho colorido é um caminho com k vértices que visita cada cor. É importante notar que a coloração do grafo tem k cores e o problema busca um caminho com k vértices, logo as cores não repetem, sendo assim um caminho de coloração própria, diferentemente do TPP.

Outros trabalhos, estudaram caminhos propriamente coloridos em grafos propriamente coloridos com diferentes abordagens em classes específicas de grafos. Por exemplo, o trabalho de Babu *et al.* (2017) investigou essa questão em grafos livres de triângulos, enquanto o trabalho de Dondi e Hosseinzadeh (2021) trabalhou com a mesma estrutura em grafos temporais.

É interessante mencionar o estudo realizado por Bessy e Bousquet (2017) sobre caminhos coloridos em grafos 3-cromáticos. Este estudo prova que todo grafo conexo 3-cromático, com exceção de C7 (um grafo de ciclo com 7 vértices), admite uma coloração de 3 vértices em que cada vértice é o início de um caminho 3-cromático com 3 vértices. Vale ressaltar que um grafo 3-cromático é aquele em que todos os vértices podem ser coloridos com três cores distintas, de forma que nenhum vértice adjacente tenha a mesma cor, e o mesmo se aplica aos caminhos 3-cromáticos. Assim, é notável que este trabalho está diretamente relacionado ao conceito de coloração própria.

Ao analisar o estado da arte da pesquisa sobre caminhos coloridos, observa-se que os estudos atuais se concentram em caminhos propriamente coloridos. No entanto, o problema TPP apresenta uma versão diferente, mas muito relacionada a versão clássica, do conceito de caminhos coloridos.

Problemas que utilizam outras estruturas tropicais

Os primeiros resultados encontrados para problemas de estruturas tropicais em grafos foram sobre conjuntos conexos tropicais de tamanho mínimo, obtidos por (CHAPELLE *et al.*, 2014). Esses estudos estão relacionados ao problema de ocorrência de Motifs em grafos, que tem várias aplicações em biologia e redes metabólicas, e tem sido amplamente estudado nos últimos anos. Outros problemas sobre conjuntos tropicais em grafos foram abordados nas seguintes referências: (COHEN *et al.*, 2017), (D'AURIAC *et al.*, 2016) e (D'AURIAC *et al.*, 2018).

3.1 Caminhos tropicais em grafos coloridos no vértices

O principal trabalho relacionado a este estudo é o artigo de Cohen *et al.* (2019), que introduz o problema do caminho tropical em grafos e suas versões de otimização.

Como principais contribuições, Cohen *et al.* (2019) trazem uma visão geral da dicotomia da complexidade do STPP e do MTPP. Especificamente, mostram que ambos os problemas são NP-difíceis para DAGs, grafos cacto e grafos de intervalo. Isso contrasta com o problema do caminho mais longo, que é polinomial para essas classes de grafos.

O trabalho também apresenta algoritmos para os problemas STPP e MTPP. Para o problema STPP, é provada uma propriedade sobre a estrutura das soluções ótimas que é útil para projetar um algoritmo parametrizado fixo. Dado um conjunto de cores C, seja P um caminho mais curto de u a v em G^c com seu conjunto de cores $C(P) = C$ e P' um subcaminho de P de w a t com seu conjunto de cores $C(P') \subseteq C(P)$. Então, P' deve ser um caminho mais curto de w a t em G^c com o conjunto de cores $C(P')$. Como resultado, isso produz um algoritmo de programação dinâmica com complexidade $O(2^c n^2)$, onde c é o número total de cores no grafo de entrada. Esse algoritmo de parâmetro fixo pode ser útil em aplicações práticas de grafos coloridos por vértices onde o número de cores é pequeno.

Para o MTPP, é mostrado que ele pode ser resolvido em tempo polinomial para várias classes de grafos, como árvores, grafos de blocos, grafos de intervalo próprio e, em particular, para grafos de cadeia bipartidos e grafos de limiar, que são os principais resultados relacionados ao MTPP. São apresentados dois algoritmos polinomiais, um para grafos de cadeia bipartidos com tempo de execução $O(c \cdot M(m, n))$ e outro para grafos de limiar com tempo de execução $max(O(c \cdot M(m, n)), O(n^4))$, onde $M(m, n)$ é o tempo de execução para encontrar um emparelhamento máximo (o maior conjunto possível de arestas não adjacentes) em um grafo geral com m arestas e n vértices. A ideia principal por trás desses algoritmos é mostrar que em grafos de cadeia bipartidos, bem como em grafos de limiar, o número de cores de qualquer caminho tropical máximo está fortemente relacionado ao número de cores de qualquer correspondência tropical. Em particular, é exatamente igual ao número de cores de qualquer combinação tropical ou é um a mais do número de cores de qualquer combinação tropical. Essa propriedade crucial permite identificar o conjunto de vértices candidatos para caminhos tropicais máximos e usar algoritmos eficientes de caminho mais longo (IOANNIDOU; MERTZIOS; NIKOLOPOULOS, 2009; UEHARA; VALIENTE, 2007) nesses vértices para calcular os caminhos tropicais máximos correspondentes.

Complexidade computacional do problema MTPP

A seguir, apresentamos o resultado obtido por Cohen *et al.* (2019), que estabelece que o problema MTPP é NP-difícil para DAGs e grafos cacto.

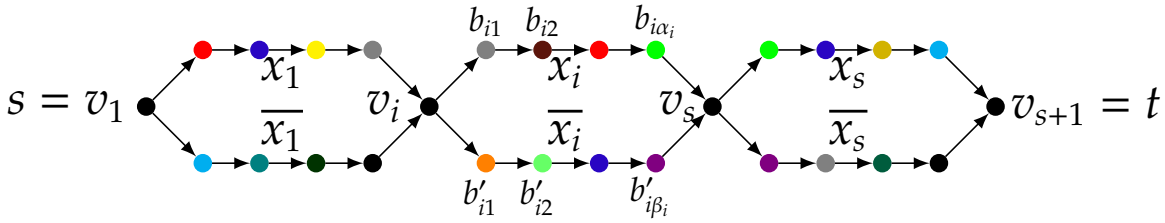
O MTPP é ainda mais difícil do que o problema do caminho mais longo. Como este último não pode ser aproximado por nenhum fator constante (KARGER; MOTWANI; RAMKUMAR, 1997), fica evidente que nenhum algoritmo de tempo polinomial pode encontrar uma aproximação de fator constante para o MTPP, a menos que **P** = **NP**. O Teorema 3.1.1 mostra que o MTPP é NP-difícil para casos especiais de grafos, como DAGs, grafos cacto e grafos de intervalo, por meio de reduções adequadas de MAX-SAT.

Teorema 3.1.1 *MTPP é NP difícil para DAGs e grafos cacto .*

Prova Considere uma expressão booleana B na *CNF* com variáveis $X = x_1, \ldots, x_s$ e cláusulas $B = b_1, \ldots, b_t$. Além disso, suponha que B contenha exatamente 3 literais por cláusula (na verdade, também podemos considerar cláusulas de tamanho arbitrário). Mostramos como construir um grafo colorido por vértices G^c associado a qualquer fórmula B, tal que existe uma atribuição de verdade às variáveis de B satisfazendo t' cláusulas se e somente se G^c contém um caminho com $t' + 1$ cores distintas. Suponha que $\forall 1 \leq i \leq s$, a variável x_i apareça nas cláusulas $b_{i1}, b_{i2}, \ldots, b_{i\alpha}$ e $\overline{x_i}$ aparece nas cláusulas $b'_{i1}, b'_{i2}, \ldots, b'_{i\beta_i}$ em que $b_{ij} \in B$ e $b'_{ik} \in B$. Agora um grafo colorido nos vértices G^c é construído da seguinte forma. Criamos $s + 1$ vértices: $s = v_1, v_2, \ldots, v_s, v_{s+1} = t$. Para cada par de vértices $(v_i, v_i + 1)$, criamos dois caminhos direcionados de v_i para v_{i+1}: $(v_i \to b_{i1} \to b_{i2} \to \cdots \to b_{i\alpha_i} \to v_{i+1})$ e $(v_i \to b'_{i1} \to b'_{i2} \to \cdots \to b'_{i\beta_i} \to v_{i+1})$. Esses dois caminhos correspondem a duas variáveis x_i e $\overline{x_i}$, respectivamente. Agora usamos $t + 1$ cores para G^c: uma cor c_0 e cada cor c_i para cada cláusula b_i, $1 \leq i \leq t$. Todos os vértices v_i são coloridos com c_0, $1 \leq i \leq s + 1$. No caso b_{ij} é b_l em B então o vértice b_{ij} é colorido com a cor c_l. Procedemos analogamente para $b'ik$. Observe que nosso grafo construído é um DAG.

Dada uma atribuição de verdade para B, obtemos um caminho de s para t em G^c como segue. Para cada variável x_i que é verdadeira, selecionamos o subcaminho $(v_i \to b_{i1} \to b_{i2} \to \cdots \to b_{i\alpha_i} \to v_{i+1})$ no caminho final. Caso contrário, para cada variável x_i que é falso, selecionamos $(v_i \to b'_{i1} \to b'_{i2} \to \cdots \to b'_{i\beta_i} \to v_{i+1})$.

Figura 2 – Redução do problema MAX-SAT para MTPP para DAG, grafo cacto.



Fonte – Cohen *et al.* (2019)

Por outro lado, de um caminho de s para t em G^c, obtemos uma atribuição de verdade para B como segue. No caso do nosso caminho seguir ($v_i \to b_{i1} \to b_{i2} \to \cdots \to b_{i\alpha_i} \to v_{i+1}$), então atribuímos x_i como verdadeiro; caso contrário, x_i é atribuído como falso. Observe que se uma cláusula b_l estiver satisfeita, então a cor correspondente c_l aparecerá em nosso caminho final e vice-versa. Assim existe uma atribuição de verdade para as variáveis de B satisfazendo t' cláusulas se e somente se G^c contém um caminho com $t' + 1$ cores distintas. Em outras palavras, $opt(G) = opt(B) + 1$ em que $Opt(G)$ é o número de cores de um caminho tropical máximo e $Opt(B)$ é o número máximo de cláusulas satisfeitas. Como um consequência, o MTPP é NP-difícil para DAGs. Note que se não considerarmos as direções das arestas de G^c, então obtemos um grafo cacto. Assim, o lema também vale para grafo cacto.

4 Modelos de programação linear inteira

Neste capítulo, é apresentado um modelo de programação linear inteira para o MTPP para grafos em geral. Também é mostrado que a formulação pode facilmente ser simplificada e aplicada para grafos acíclicos direcionados (DAGs). E por último, é obtido um modelo de programação linear inteira para STPP.

Ao longo deste capítulo considere k e n inteiros positivos, onde n representa o número de vértices do grafo e k representa o número de cores utilizadas pelo grafo. Além disso, suponha que $n \geq 3$ e $k \geq 3$, caso contrário, a instância correspondente pode ser facilmente resolvida.

4.1 Uma formulação linear inteira do problema MTPP

Para formular o MTPP como um programa linear inteiro, considere que uma instância desse problema consiste em um grafo colorido nos vértices $G^C = (G, C)$. Para a formulação apresentada, cada aresta $uv \in E$, será substituída por dois arcos direcionados uv e vu, $A = \{(u, v) \cup (v, u), \forall uv \in E\}$.

Para cada vértice $u \in V$ e cor $c \in C$, considere a variável binária X_{uc} como segue:

$$X_{uc} = \begin{cases} 1, & \text{se o vértice } u \text{ de cor } c \text{ está na solução,} \\ 0, & \text{caso contrário.} \end{cases}$$

Para cada arco $uv \in A$, considere a variável binária Y_{uv} com o seguinte significado:

$$Y_{uv} = \begin{cases} 1, & \text{se o arco } uv \text{ está na solução,} \\ 0, & \text{caso contrário.} \end{cases}$$

Para expressar a função objetivo, considere a variável Z_c, para cada cor $c \in C$, definida como segue:

$$Z_c = \begin{cases} 1, & \text{se a cor } c \text{ esta na solução,} \\ 0, & \text{caso contrário.} \end{cases}$$

Por fim, é usada uma função $l_v : V \to \mathbb{N}$ que enumera, rotula os vértices ao longo do caminho em ordem crescente. Tal função é apresentada em Miller, Tucker e Zemlin (1960) com intuito de eliminar ciclos na solução.

A seguir é apresentado uma formulação 0/1 para o MTPP como segue:

$$\max \sum_{c \in C} Z_c$$

s. t.

$$Z_c \leq \sum_{u \in V} X_{uc}, \qquad \forall c \in C \tag{1}$$

$$\sum_{c \in C} X_{vc} - \sum_{u \in \delta^-(v)} Y_{uv} \geq 0, \qquad \forall v \in V \tag{2}$$

$$\sum_{c \in C} X_{vc} - \sum_{w \in \delta^+(v)} Y_{vw} \geq 0, \qquad \forall v \in V \tag{3}$$

$$(l_v - l_u) \geq Y_{uv} - (1 - Y_{uv})N, \qquad \forall uv \in A \tag{4}$$

$$\sum_{u \in V} \sum_{c \in C} X_{uc} - \sum_{u \in V} \sum_{v \in V} Y_{uv} = 1, \tag{5}$$

$$X_{uc}, Y_{uv} \in \{0, 1\}, \qquad \forall u \in V, \forall uv \in A, \forall c \in C \tag{6}$$

$$Z_c \in \{0, 1\}, l_v \geq 0; \qquad \forall v \in V, \forall c \in C \tag{7}$$

- A função objetivo maximiza o número de cores que devem compor a solução.
- A restrição (1) impede que a variável Z_c assuma valor 1 sem que pelo menos um vértice de cor c esteja na solução.
- Dado um vértice v, as restrições (2) e (3) garantem que se um vértice v está na solução o número de arestas de entrada e saída de v serão no máximo 1, respectivamente, trata-se da restrição de conservação do fluxo.
- A restrição (4) faz uso da função l_v que permite numerar os vértices ao longo do caminho em ordem crescente. Se um dado arco (u, v) está na solução, o lado direito da inequação é 1 e a diferença entre os rótulos associados aos vértices do arco uv deve ser pelo menos 1. A lacuna dessa diferença, no lado esquerdo, mostra exatamente a distância entre dois vértices em uma solução. Essa desigualdade é essencial para se evitar ciclos na solução.
- A igualdade (5) garante que a diferença entre o número de vértices e o número de arestas que compõe a solução é exatamente 1. Vale ressaltar que a restrição (5) isoladamente não elimina ciclos, embora seja verdadeira para caminhos e árvores. Por exemplo, podemos ter uma solução desconexa formada por um ciclos e um caminho e ainda satisfazer a restrição (5).

- As restrições (6) e (7) garantem a condição de integralidade das variáveis.

Uma vez que o objetivo do problema MTPP é encontrar um caminho P cujo conjunto de cores usadas por P seja o maior possível, o modelo apresentado resolve o problema MTPP e será provado a seguir.

Teorema 4.1.1 *A solução restrita x das soluções inteiras (x, y, z) para a formulação de (1) à (7) define um caminho no grafo (G, C) que usa $|C'|$ cores, onde $C' \subseteq C$.*

Prova Para qualquer solução inteira viável (x, y, z) da formulação, seja $V_P = \{u : X_{uc} = 1\}$, $A_P = \{uv : Y_{uv} = 1\}$ e $C_P = \{c : Z_c = 1\}$. Note que se um vértice $u \in V_P$ então $c \in C_P$. Seja C' o conjunto formado pelas cores dos vértices em que $X_{uc} = 1$. Se um arco $uv \in A_P$, pela restrição (2), $\sum_{c \in C} X_{vc} = 1$ e pela restrição (3), $\sum_{c \in C} X_{uc} = 1$ logo $u, v \in V_P$. Pelas mesmas condições impostas nas restrições (2) e (3), para todo $u \in V_P$, existe no máximo um arco saindo de u e no máximo um arco entrando em u. Note que o subgrafo $G = (V_P, A_P, C_P)$ está bem definido, porém, não necessariamente conexo. Se um arco $uv \in A_P$, pela restrição (4) $Y_{uv} = 1$ e a diferença entre os rótulos dos vértices u e v tem que ser pelo menos um, o que impede circuitos. Esta restrição juntamente com a restrição (5) garante a conectividade de G e portanto x indica um caminho que usa C' cores, o que completa a prova.

É sabido que o MTPP também é NP-difícil para DAGs. Na formulação proposta, para instâncias de DAGs, a restrição 4 pode ser removida uma vez que não temos ciclos. Destacamos que para instâncias de DAGs, não é realizada a substituição das aresta por arcos direcionados, pois DAGs já possuem a característica de terem arcos direcionados.

4.1.2 Análise da complexidade do modelo

Uma indicação da boa qualidade dessa modelagem para o MTPP é que cada conjunto de restrições tem uma quantidade linear de desigualdades adicionadas ao modelo. Note que as restrições 1 adicionam $|C|$ desigualdades, as restrições 2 e 3 adicionam cada uma $|V|$ desigualdades e as restrições 4 adicionam $2|A|$ desigualdades. Para completar, a 5 adiciona uma desigualdade ao modelo. Pela somatória geral, o modelo possui $|C| + 2|V| + 2|A| + 1$ desigualdades, ou seja, possui uma quantidade linear de desigualdades em relação ao tamanho da instância.

4.2 *Uma formulação linear inteira do problema STPP*

Outro problema diferente do MTPP, mas também relacionado ao problema TPP, é o STPP. Neste problema, dado um grafo ponderado colorido nos vértice $G^C = (G, C, w)$ e dois vértices s e t, o objetivo consiste em encontrar um caminho tropical de s a t com peso total mínimo. Note que embora o MTPP e o STPP sejam problemas que buscam por caminhos tropicais, os objetivos propostos são distintos e somente uma solução ótima para o STPP pode ser vista como uma solução ótima para o MTPP, o recíproca não é verdade.

Esta seção apresenta uma formulação do STPP como um programa linear inteiro. Considere que uma instância desse problema consiste em um grafo ponderado colorido nos vértices $G^C = (G, C, w)$ e dois vértices do grafo s e t.

Na formulação do STPP toda aresta $uv \in E$ foi substituída por dois arcos direcionados uv e vu, de forma que $A = \{\{(u, v) \cup (v, u), \forall uv \in E\}\}$. O peso atribuído a cada arco $uv \in A$ é equivalente ao peso $w(uv)$ de sua aresta correspondente. A formulação do STPP mantém as variáveis X_{uc} e Y_{uv} seguindo as mesmas definições da formulação para o MTPP.

Para expressar a função objetivo, considere constantes w_{uv}, para cada arco $uv \in A$, que define o peso atribuído ao arco uv.

Assim como no problema MTPP, na formulação para o problema STPP é usada a função $l_v : V \to \mathbb{N}$ apresentada por Miller, Tucker e Zemlin (1960) com intuito de eliminar circuitos da solução.

A seguir é apresentada uma formulação 0/1 para o STPP como segue:

$$\min \sum_{u \in V} \sum_{v \in V} w_{uv} Y_{uv}$$

s. t.

$$\sum_{u \in V} X_{uc} \geq 1, \qquad \forall c \in C \tag{8}$$

$$\sum_{c \in C} X_{vc} - \sum_{u \in \delta^-(v)} Y_{uv} = 0, \qquad \forall v \in V - \{s, t\} \tag{9}$$

$$\sum_{c \in C} X_{vc} - \sum_{w \in \delta^+(v)} Y_{vw} = 0, \qquad \forall v \in V - \{s, t\} \tag{10}$$

$$\sum_{v \in \delta^+(s)} Y_{sv} = 1, \sum_{v \in \delta^-(s)} Y_{vs} = 0, \tag{11}$$

$$\sum_{v \in \delta^-(t)} Y_{vt} = 1, \sum_{v \in \delta^+(t)} Y_{tv} = 0, \tag{12}$$

$$(l_v - l_u) \geq Y_{uv} - (1 - Y_{uv})N, \qquad \forall uv \in A \tag{13}$$

$$X_{uc}, Y_{uv} \in \{0, 1\}, \qquad \forall u \in V, \forall uv \in A, \forall c \in C \tag{14}$$

$$l_v \geq 0; \qquad \forall v \in V, \forall c \in C \tag{15}$$

Note que apesar das semelhanças entre as restrições das formulações MTPP e STPP, as restrições definem um conjunto de soluções validas distintas, principalmente pela diferença entre objetivos dos problemas.

Uma vez que o objetivo do problema STPP é encontrar um caminho P de s a t cujo conjunto de cores usadas por P possua todas as cores do grafo de entrada com o menor peso possível, a primeira característica é determinar que a função objetivo minimize o peso em relação as arestas que compõem a solução. Note que a restrição (8) garante que pelo menos um vértice que possua cada cor c esteja em P. Como P sempre possui todas as cores do grafo de entrada, então a solução é ótima, e P é um caminho tropical.

As igualdades (11) e (12) definem respectivamente que o número de arcos que saem de s é 1 e o número de arcos que entram em s é 0 e o número de arcos que entram em t é 1 e número de arcos que saem de t é 0. Como todos os outros vértices do caminho são vértices internos, as restrições (9) e (10) definem o número de arcos de entrada e saída dos vértices internos como exatamente 1.

A restrição (13) se repete em relação à formulação para o MTPP e permite numerar os vértices ao longo do caminho em ordem crescente. Reforçando, essa desigualdade

é extremamente poderosa para evitar ciclos na solução. As restrições de (9) a (13) combinadas garantem que a solução seja exatamente um único caminho conexo de s a t e não outra qualquer estrutura. Para manter a integralidade das variáveis temos as restrições (14) e (15).

O STPP também é NP-difícil para DAGs logo o modelo pode facilmente ser simplificado para executar esse tipo de instância assim como feito para o MTPP.

A equação linear $|C|+2|V|+2|A|$ representa o número de desigualdades adicionadas ao modelo em relação ao tamanho da instância para o problema STPP, uma restrição a menos que no problema MTPP.

É de conhecimento que a modelagem de um problema NP-difícil como um problema de programação linear inteira não é uma solução pratica para o problema, devido principalmente ao tempo de execução para instâncias de grande porte, porém tal formulação é muito importante para o estudo de problemas de natureza discreta.

Sabe-se que poucas são as técnicas que apresentam resultados práticos significativos em termos de solução exata para problemas dessa magnitude. De fato, a complexidade envolvida é um fator limitante crucial para o avanço computacional. Por outro lado, uma das poucas área de pesquisa que apresenta técnicas que tem-se mostrado eficaz na resolução de instâncias reais de grande porte é a combinatória poliédrica (FERREIRA; WAKABAYASHI, 1996).

A combinatória poliédrica é um área da computação baseada na teoria de poliedros e a de programação linear. Em termos bem amplos, pode-se dizer que em combinatória poliédrica estuda-se propriedades de poliedros associados a problemas combinatórios (FERREIRA; WAKABAYASHI, 1996).

Alguns exemplos de técnicas que podem obter soluções para problemas combinatórios são:

- Algoritmos gulosos;
- Algoritmo simplex;
- Branch e bound;
- Branch e cut;
- Programação dinâmica;
- Relaxação lagrangeana;
- Métodos de planos-de-corte.

Tais técnicas são implementadas aos problemas por meio de sua modelagem como um problema de programação linear inteira. O uso dessas técnicas apresentam bons resultados em encontrar soluções ótimas para instâncias reais de grande porte de problemas NP-difíceis. Uma outra vantagem de se usar tais métodos, é o fato de que podemos derivar soluções aproximadas para os problemas, com garantia de qualidade (FERREIRA; WAKABAYASHI, 1996).

5 Heurísticas para o problema MTPP

Sabe-se que o MTPP é NP-difícil para grafos simples em geral, em especial para DAGs, grafos cacto e grafos de intervalo. A intratabilidade inerente do MTPP encoraja o desenvolvimento de heurísticas que são projetadas para encontrar limitantes, no caso inferiores, para o problema.

Heurísticas são métodos aproximativos utilizados dentro do contexto dos métodos de otimização. As heurísticas, quando bem desenvolvidas, geram boas soluções em um tempo computacional viável para aplicações práticas (TALBI, 2009). O uso de heurísticas para tratar problemas em otimização vêm sendo, há muitos anos, uma alternativa aos métodos exatos. Relembrando, métodos exatos são aqueles que garantem encontrar o ótimo global, mas podem demandar uma grande quantidade de recursos computacionais e tempo para isto. A aplicação de heurísticas para o tratamento de problemas da área de otimização vem crescendo devido à sua capacidade de gerar soluções com qualidade aceitável para problemas que possuem instâncias consideravelmente grandes e difíceis de serem tratadas.

Neste capítulo é apresentada uma heurística de tempo polinomial para encontrar caminhos tropicais máximos em grafos coloridos nos vértices. Não foi encontrado na literatura algoritmos exatos, de aproximação ou heurísticas para o problema MTPP. O principal algoritmo da heurística apresentada é chamado *Tropical Cacto*. O algoritmo *Tropical Cacto* foi projetado a partir da ideia do algoritmo proposto por Markov *et al.* (2012) que trata-se de um algoritmo de tempo linear para calcular caminhos mais longos em grafos cacto. Com isso o algoritmo *Tropical Cacto* tem como entrada um grafo cacto colorido nos vértices e calcula o número de cores de um caminho em um grafo cacto e pode ser facilmente modificado para produzir um caminho colorido em um grafo cacto.

Com o propósito de utilizar o algoritmo *Tropical Cacto* para encontrar caminhos coloridos em grafos, foi desenvolvido um algoritmo auxiliar que constrói a partir de um grafo simples, um grafo cacto. O algoritmo será nomeado de *Grafo Cacto*. O algoritmo *Grafo Cacto* foi obtido a partir de algumas modificações do algoritmo de construção de árvores de decomposição em circuitos apresentado por Rezende (2014) co-m base no algoritmo de Gabow e Nie (2008) com o intuito de encontrar um caminho de comprimento $exp(\Omega(\sqrt{log\ L}))$, onde L é o comprimento de um caminho mais longo.

Dessa forma é possível executar o algoritmo *Tropical Cacto* para grafos que possui como sub-rotina o algoritmo *Grafo Cacto* de tal maneira que um caminho colorido em um grafo cacto é um caminho colorido em um grafo simples.

Para cada etapa da heurística, primeiramente, são apresentados seus componentes essenciais e posteriormente, os algoritmos contendo todos os procedimentos. Para facilitar o entendimento, o texto detalha algumas passagens mais relevantes do funcionamento dos algoritmos.

5.1 Construção de Cactos

Dado um grafo simples G, a primeira etapa do algoritmo *Grafo Cacto* consiste em extrair de G um grafo cacto G'. Para realização desse processo, é necessário encontrar *ciclos longos* no grafo.

Dado um inteiro $k \geq 3$, um ciclo em grafo não direcionado é chamado de *longo* se ele tiver pelo menos k arestas. Este problema para grafos não direcionados é conhecido por ser tratável por parâmetros fixos.

Teorema 5.1.1 *Em um grafo conexo não direcionado com um* ciclo longo, *ou toda árvore de busca em profundidade tem um ciclo fundamental longo ou algum ciclo longo tem no máximo* $2k - 4$ *arestas.*

O Teorema 5.1.1 é provado por Gabow e Nie (2008). O resultado acima permite que um ciclo longo seja encontrado usando o algoritmo clássico de busca em profundidade associado ao algoritmo de Alon, Yuster e Zwick (1995) que trata-se de um método randomizado para encontrar caminhos e ciclos simples de um comprimento especificado k em grafos. O tempo esperado é $2^{O(k)}n$ e $2^{O(k)}n\ log\ n$ no pior caso.

Neste trabalho, a associação de algoritmos supracitada foi nomeada por *EncontraCiclo* tendo como entrada um vértice v e um grafo G, e como saída um ciclo em G que passe por v, caso existir. Sem perda de generalidade, k foi definido com 3.

Teorema 5.1.2 *Dado um grafo G com n vértices e m arestas, um vértice v e um inteiro k, é possível encontrar em G um ciclo de tamanho $\geq k$ (também chamado de $k^+ - ciclo$) que passa por v (se existir) em tempo $O(m) + 2^{O(k)}n\ log\ n$.*

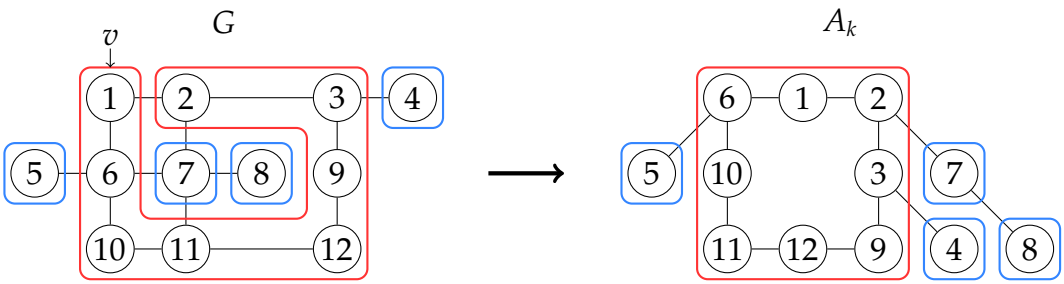
O Teorema 5.1.2, provado por Gabow e Nie (2008), garante que dado um vértice $v \in G$ e um inteiro k é possível encontrar um ciclo longo em G de comprimento pelo menos k que passa por v em tempo polinomial. Para $k \leq \left\lceil \frac{1}{4} \ log \frac{L}{2} \right\rceil$.

Os resultados dos Teoremas 5.1.1 e 5.1.2 justificam a metodologia empregada na construção da heurística, onde o 5.1.2 demonstra a existência do caminho procurado e o 5.1.1 mostra de que forma esse caminho será encontrado.

Desenvolvimento do algoritmo

Dado um grafo simples G, um vértice qualquer $v \in G$ e um inteiro positivo $k \geq 3$, o grafo construído é definido por $A_k = A_k(G, v)$. Dois tipos de componentes formam A_k. Um componente em A_k ou é um *componente-singular* ou é um *componente-circuito*. Um *componente-singular* corresponde a um único vértice $u \in G$, e define um vértice em A_k, enquanto um *componente-circuito* corresponde a um circuito $C \in G$, e define um circuito em A_k.

Figura 3 – Exemplo de transformação de G em A_k pra $k \geq 3$



Fonte – Igor Sampaio, 2023

A Figura 3 apresenta um exemplo de transformação de um grafo simples G em um grafo cacto A_k, onde k é o tamanho mínimo do circuito. Note que o conjunto de vértices circulados pela linha vermelha representa um *componente-circuito* no grafo A_k e possui comprimento 9. Os vértices circulados pela linha azul representam *componentes-singulares* no grafo A_k. Note que o vértice 7 poderia ser vizinho de 2, 6 ou 11 no grafo A_k além disso, numa segunda execução do algoritmo, um circuito de tamanho diferente poderia ser encontrado.

O primeiro vértice a fazer parte de A_k é o vértice v dado, e após a execução do algoritmo, v pode ser um componente singular ou fazer parte de um componente-circuito em A_k. Também é definido neste trabalho, de modo geral, o tamanho mínimo k de um

ciclo igual a 3, pois a varição de tal variável não apresentou mudança significativa nos resultados finais de caminhos encontrados pela heurística.

Algoritmo 1 Grafo Cacto

Entrada: G um grafo colorido; e v um vértice aleatório do grafo.
Saída: A_k.

```
1   C = EncontraCiclo(G, v);
2   se C ≠ ∅
3       A_k ← C − v;
4       para cada componente H ∈ G − C faça
5           Escolha um vértice u ∈ H vizinho de w ∈ C;
6           V(A_k) ← u;
7           E(A_k) ← (u, w);
8           se V(H)\u ≠ ∅
9               GrafoCacto(H, u)
10  senão
11      Escolha um vértice u ∈ G vizinho de v
12      V(A_k) ← u;
13      E(A_k) ← (v, u);
14      se G − v\u ≠ ∅
15          GrafoCacto(G − v, u)
```

Fonte – Igor Sampaio, 2023

Note que cada passo recursivo extrai de G um ciclo ou um vértice que está conectado a outro ciclo ou a um vértice, logo A_k é de fato um grafo cacto, além disso, o número de passos recursivos do algoritmo é linear, portanto o algoritmo *Grafo Cacto* que extrai de um grafo simples G um grafo cacto G' é polinomial.

5.2 Heurística para grafos cacto

Seja (G, C) um grafo cacto em que V é o conjunto de vértices do grafo cacto, E o conjunto de arestas do grafo cacto e $C : V \to C$ a coloração dos vértices do grafo, na qual C é o conjunto de cores $\{c_1, \ldots, c_k\}$ utilizadas pelo grafo, ou seja, k é um inteiro positivo que representa o número de cores deste conjunto.

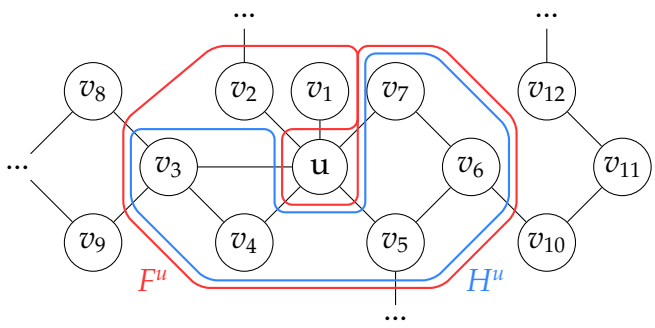
Em um grafo cacto dois ciclos simples têm no máximo um vértice em comum, isso é equivalente a um grafo conexo em que cada aresta pertence a no máximo um ciclo. Este é um conceito essencial para a definição das estruturas utilizadas no algoritmo. Será adotado que um *grafo cacto enraizado* G é um grafo cacto em que um vértice foi escolhido como uma raiz de G ($raiz(G)$).

Definição 5.2.1 *Dado um vértice u em um grafo cacto G. Os filhos de u são todos os vértices adjacentes a u ou que estão em um ciclo que contenha u.*

Definição 5.2.2 *Dado um grafo cacto G e um vértice $u \in V(G)$, denota-se por F^u o conjunto de todos os vértices que são filhos de u e por $H^u \subseteq F^u$ o subconjunto de F^u formado somente pelos filhos de u que pertencem a algum ciclo que contém u.*

Definição 5.2.3 *Um vértice u que não possui filhos ou é um vértice folha ou é um vértice ponte, ou seja, um vértice que pertence a um ciclo.*

Figura 4 – Um grafo cacto enraizado em u



Fonte – Igor Sampaio, 2023

Para ilustração, a Figura 4 demonstra um grafo cacto enraizado em u, e destaca um exemplo de formação dos conjuntos F^u e H^u. Na Figura F^u está demarcado pela linha vermelha, e neste exemplo é formado pelos vértices $F^u = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, já H^u está demarcado pela linha azul, e neste exemplo é formado pelos vértices $H^u = \{v_3, v_4, v_5, v_6, v_7\}$.

As propriedades de um grafo cacto garantem que ao excluir um vértice qualquer $u \in V(G)$ do grafo cacto é possível definir um subgrafo cacto conexo diferente para cada filho de u.

Ao associar um valor inteiro a cada vértice de um grafo cacto tem-se um *grafo cacto rotulado*. Neste algoritmo, dois rótulos são associados a cada vértice $u \in V$, o rótulo parcial e o rótulo total, ambos definidos abaixo.

Definição 5.2.4 *O valor do rótulo parcial de u ($r_p(u)$) representa o maior número de cores de um caminho em um subgrafo cacto enraizado em u.*

Definição 5.2.5 *O valor do rótulo total de u ($r_t(u)$) representa o maior número de cores de um caminho que contém u e está contido em subgrafos cacto enraizados em u.*

Nesse sentido, o algoritmo *Grafo Cacto* consiste em retornar o maior valor entre os rótulos $r_t(v)$ para todo $v \in V(G)$.

Para calcular o rótulo dos vértices, todo vértice $u \in V$ recebe uma lista de cores $L(u)$ com uma cor inicial que é a cor de u e é atualizada a cada execução do algoritmo. Tal lista L armazena o conjunto de cores de um caminho em um subgrafo cacto enraizado em u que contém o maior número de cores. O rótulo de cada vértice (exceto vértice folha e vértice ponte que tem rótulos = 1) é calculado a partir das lista de cores de seus filhos.

5.2.6 O algoritmo

A seguir é descrita uma heurística de tempo polinomial para o problema MTPP, nomeada por *Tropical Cacto*. Este algoritmo, por sua vez, utiliza um algoritmo auxiliar, chamado *Calcula Rótulos*, que rotula os vértices do grafo cacto enraizado. Sua descrição é apresentada em seguida.

Para simplificar, é possível dizer que o algoritmo *Tropical Cacto* utiliza o algoritmo auxiliar *Calcula Rótulos* para rotular todos os vértices do grafo cacto G, dado como entrada um vértice u que será a raiz de G. Para cada raiz $u' \neq u$ os vértices de G recebem uma rotulagem, dessa forma cada vértice do grafo cacto tem $|V|$ rótulos que podem ter valores diferentes dependendo do vértice que foi definido como raiz do grafo cacto.

Em síntese, o algoritmo *Tropical Cacto* retorna o maior valor encontrado dos rótulos $r_t(u)$ para todo $u \in V$. Pela definição de $r_t(u)$, isso é equivalente ao maior número de cores de um caminho que passa por u entre os subgrafos cacto enraizados em u, logo equivale ao número de cores do melhor caminho encontrado pelo algoritmo. Caso o algoritmo encontre um $r_t(u)$ igual ao número de cores do grafo cacto de entrada, este $r_t(u)$ representará um caminho tropical e portanto uma solução ótima para o problema.

O algoritmo *Calcula Rótulos* é o cerne da heurística aqui apresentada, pois nele é realizado todo o cálculo do rótulo de cada vértice do grafo cacto.

Algoritmo 2 Tropical Cacto

Entrada: Um par (G, C), onde G é um grafo cacto e C é a coloração de G.
Saída: $|C^*|$, o maior número encontrado de cores de um caminho no grafo de entrada.

```
1  para cada root ∈ V
2      CalculaRótulos(G, C, root)
3  devolva max(r_t(root))
```

Fonte – Igor Sampaio, 2023

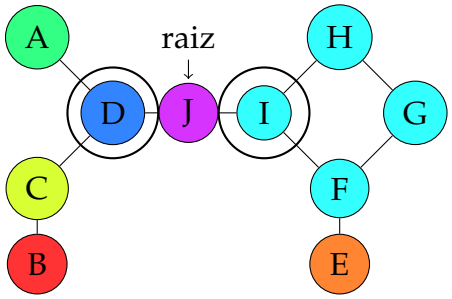
O algoritmo *Calcula Rótulos* é recursivo, o que faz com que a rotulagem dos vértices inicie pelos vértices folhas depois nos vértices pontes e por último a raiz. O algoritmo *Calcula Rótulos* é executado para todos os vértices do conjunto F^u, filhos da raiz u. O rótulo do vértice u só é calculado após todos os vértices de F^u já possuir rótulo.

Dado um grafo G, seja $|F^u| = d$ a quantidade de filhos de um vértice u, e t a quantidade de vértices de um ciclo qualquer em G. Pelas propriedades dos grafos cactos pode-se destacar que $d \geq 0$ e $t \geq 3$.

Vale observar que o algoritmo pode ser facilmente adaptado para retornar os vértices do melhor caminho encontrado, em tempo linear. Para isso, basta armazenar os vértices do caminho que formam a lista de cores $L(u)$. Assim é possível que o algoritmo retorne esse conjunto de vértices que possui o maior $r_t(u)$ entre as rotulagens do grafo.

Para um melhor entendimento do algoritmo *Calcula Rótulos*, exibimos nas Figuras 6 e 7 o processo de rotulagem da instância para o problema apresentado na Figura 5. Neste exemplo, o grafo cacto está enraizado no vértice J. A Figura 8 é a rotulagem realizada pelo algoritmo *Calcula Rótulos*.

Figura 5 – Grafo cacto enraizado no vértice J



Fonte – Igor Sampaio, 2023

Algoritmo 3 Calcula Rótulos

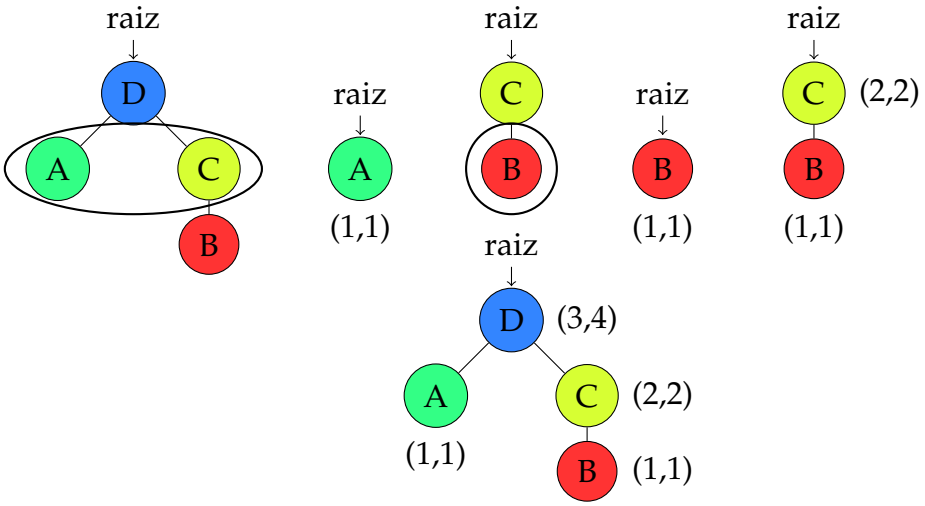
Entrada: (G, C, u) grafo cacto colorido e u um vértice raiz do grafo.

Saída: grafo cacto rotulado G^r.

se u é uma folha ou ponte
  $r_p(u) \leftarrow 1$; $r_t(u) \leftarrow 1$; $L(u) \leftarrow c(u)$
senão
  /* $F^u = v_1, v_2, \ldots, v_d$ `filhos de` u */
  para cada $v_i \in F^u$ **faça**
    $CalculaRótulos(G, C, v_i)$
  para cada $v_i \in F^u$ **faça**
    se $v_i \in H^u$
  /* `Seja` $\{w_1, w_2, \ldots, v_i, \ldots, w_t\} \subseteq H^u$, $P_1 = \{w_1, \ldots, v_i\}$ `e` $P_2 = \{v_i \ldots w_t\}$*/
      $C(P_1) \leftarrow \{L(v_i) \dot\cup c(u) \dot\cup \{c(w_j) \mid j : 1 \ldots v_i\}\}$;
      $C(P_2) \leftarrow \{L(v_i) \dot\cup c(u) \dot\cup \{c(w_j) \mid j : v_i \ldots t\}\}$;
      se $|C(P_1)| \geq |C(P_2)|$
        $CF_{v_i} \leftarrow C(P_1)$
      senão
        $CF_{v_i} \leftarrow C(P_2)$;
    senão
      $CF_{v_i} \leftarrow L(v_i) \dot\cup c(u)$
  $L(u) \leftarrow CF_{v_i}$, onde $|CF_{v_i}| \geq |CF_{v_j}| \; \forall v_i \neq v_j \; e \; v_i, v_j \in F^u$
  $r_p(u) \leftarrow |L(u)|$
  $r_t(u) \leftarrow \max(|CF_{v_i} \dot\cup CF_{v_j}|) \mid \forall v_i \neq v_j \; e \; v_i, v_j \in F^u$
devolva G^r

Fonte – Igor Sampaio, 2023

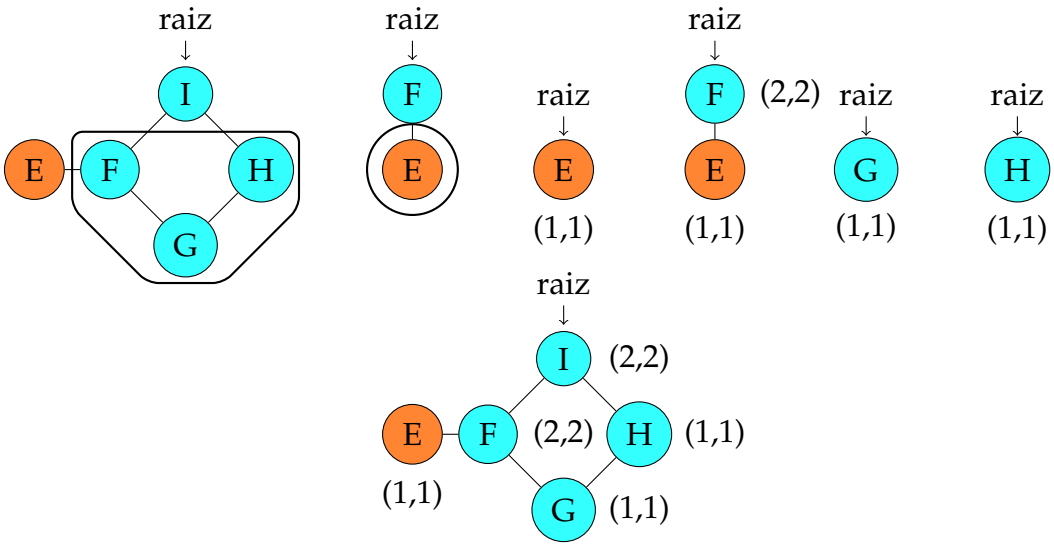
Figura 6 – Rotulagem dos vértices



Fonte – Igor Sampaio, 2023

A Figura 6, mostra os primeiros passos do algoritmo, iniciando pelo cálculo do rótulo do primeiro subgrafo cacto enraizado no vértice J. O vértice D é um filho do vértice J e ele possui os vértices A e C como filhos. É sabido que para o cálculo do rótulo de qualquer vértice v, todos os filhos de v já devem ter sido rotulados e assim sucessivamente. Dessa forma as folhas A e B são os primeiras a receberem seus rótulos, em sequência o vértice C e por último o vértice D.

Figura 7 – Continuação da rotulagem dos vértices



Fonte – Igor Sampaio, 2023

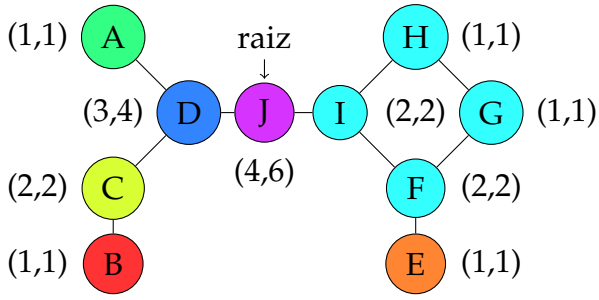
O cálculo dos rótulos de um ciclo é demonstrado na Figura 7. O vértice I pertence a um ciclo, dessa forma todos os vértices que também compõem este ciclo são filhos de I que é o caso dos vértices F, G, H. Na Figura pode-se notar que antes de calcular o rótulo

do vértice I, que é a raiz do subgrafo cacto, todos os vértices que também compõem o ciclo tem seus rótulos calculados a priori.

No exemplo, o vértice F forma um subgrafo cacto composto por ele e pelo vértice E. Como o vértice E é uma folha, o rótulo de E é calculado primeiro e logo em seguida o rótulo de F.

Através do exemplo, é possível confirmar que um vértice ponte é tratado com a mesma prioridade de um vértice folha pelo algoritmo. Isso pode ser observado na forma como o rótulo dos vértices G e H é calculado. É importante destacar que o rótulo se refere à quantidade de cores e não à quantidade de vértices no caminho, como pode ser visto no rótulo definido para o vértice I na Figura.

Figura 8 – Grafo totalmente rotulado



Fonte – Igor Sampaio, 2023

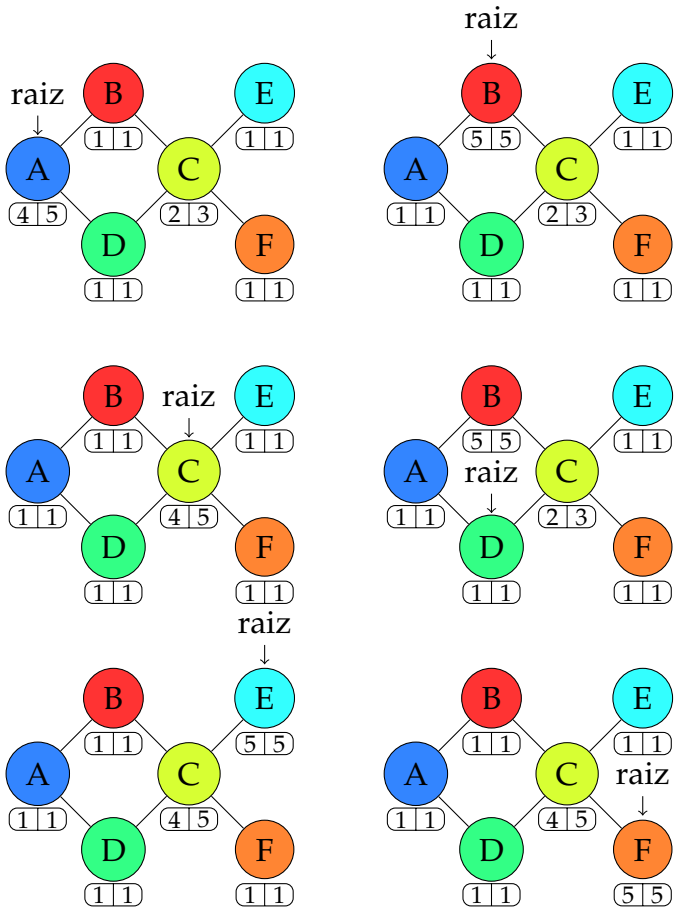
A Figura 8 apresenta o grafo totalmente rotulado enraizado no vértice J. A ordem da rotulagem dos vértices do grafo nesse caso foi: $A, B, C, D, E, F, G, H, I, J$.

Note que o vértice folha ou ponte de cada subgrafo cacto é sempre o primeiro vértice a ter seu rótulo calculado enquanto que o vértice raiz do subgrafo cacto é sempre o ultimo.

Algo importante nesse exemplo é que essa rotulagem encontrou um caminho tropical, pois o número do maior rótulo r_t é igual ao número de cores presentes no grafo de entrada. Nesse caso, o cálculo dos rótulos com o grafo cacto enraizado em outros vértices se faz desnecessário, pois uma solução ótima já pode ser retornada.

A Figura 9 apresenta um exemplo de um grafo no qual o algoritmo atribui rótulos diferentes aos vértices, dependendo da raiz escolhida para o grafo. Neste caso, é executado o algoritmo para cada vértice, considerando-o como a raiz do grafo. Isso acontece porque, neste exemplo, nenhum r_t tem o valor igual ao número total de cores presentes no grafo.

Figura 9 – Grafo rotulado com cada vértice sendo raiz



Fonte – Igor Sampaio, 2023

No exemplo, o grafo de entrada possui 6 cores e o maior r_t retornado entre todas as rotulagens é 5, com isso já é possível concluir que o algoritmo não encontrou um caminho tropical, embora, neste exemplo, o grafo de fato não possua um caminho tropical, sendo portanto o valor ótimo para esta instância.

A seguir serão mostrados alguns resultados teóricos do algoritmo apresentado.

Corretude

Proposição 5.2.7 *O algoritmo Calcula Rótulos atribui para cada vértice $u \in V(G)$, na variável $L(u)$, o conjunto de cores de um caminho que possui u como extremidade.*

Prova A demonstração será feita por indução no número de iterações necessárias para calcular o conjunto de cores de cada vértice.

Base da indução: No início, se houve somente uma iteração, o vértice inicial u é o único vértice do problema, logo o vértice é uma folha ou uma ponte e o conjunto de cores $L(u) = c(u)$ e o caminho associado é trivial.

Hipótese de indução: Suponha que o algoritmo *Calcula Rótulos*, para cada vértice $u \in V(G)$ visitado em k etapas, atribuiu a lista de cores de u, na variável $L(u)$. Além disso, são conhecidos os respectivos caminhos que representam cada $L(u)$ calculado. Note que, para cada caminho que está associado a algum $L(u)$ todos os vértices desse caminho foram visitados em etapas anteriores a k. Em outras palavras, suponha que após a execução de k iterações do algoritmo, todos os nós $u \in V(G)$ que foram visitados pelo algoritmo, é conhecido o $L(u)$ e o respectivo caminho associado a $L(u)$.

Passo da indução: Considere a execução de $k + 1$ iterações. Na $k + 1-$iteração, o algoritmo irá atribuir $L(v)$ para algum vértice v cujos filhos são conhecidos a lista L e o respectivo caminho associado, assim o algoritmo atribui a $L(v)$ a cor de v e as cores da lista de algum filho de v que melhor contribui para o caminho de maior número de cores, seja u tal vértice. A existência do caminho que contém v como extremidade e está associado à lista $L(v)$ pode ser encontrado nas seguintes condições:
Caso (a) Se u, filho de v, não participa de um ciclo que contenha v.

> Nesse caso, de acordo com a Definição 5.2.1, u é um vértice adjacente a v, formando um caminho simples, então o caminho que representa $L(v)$ é o caminho que representa $L(u)$ adicionado da aresta vu.

Caso (b) Se u, filho de v, participa de um ciclo que contenha v.

> Nesse caso, de acordo com a Definição 5.2.1, u é um vértice que participa de um ciclo que contém v e não é necessariamente adjacente de v, ou seja, existem exatamente dois caminhos simples possíveis entre v e u formados por vértices que estão no ciclo que contém v e u. Assim o caminho que representa $L(v)$ é o caminho que representa $L(u)$ juntamente com um dos dois caminhos para se alcançar v, aquele que melhor contribuir para o aumento da cardinalidade de $L(v)$.

Portanto, após $n \in \mathcal{N}$ iterações, o algoritmo Calcula Rótulos atribui a cada vértice $v \in V(G)$ uma lista de cores que está associado a um caminho que possui v como extremidade.

A partir da Proposição 5.2.7 apresentada anteriormente, é possível obter o Corolário 5.2.8.

Corolário 5.2.8 *Dado um grafo cacto* (G, C) *e um vértice raiz* u*, o algoritmo Calcula Rótulos, rotula todos os vértices de* G *com base no conjunto de cores encontrados para cada vértice e atribuídos a* L*.*

Nos mesmos passos apresentados na Proposição 5.2.7, os rótulos de cada vértice u são atribuídos. Se u é um vértice folha ou ponte ambos rótulos $r_p(u)$ e $r_t(u)$ recebem o valor 1 (linha 2 do algoritmo). No demais o algoritmo calcula os rótulos dos vértices a partir do conjunto de cores de seus filhos atribuídos a L. O algoritmo define para um vértice u, $r_p(u)$ como sendo a cardinalidade do conjunto $L(u)$ (linha 17 do algoritmo), e como $r_t(u)$ a cardinalidade da união disjunta de $c(u) \dot{\cup} L(u') \dot{\cup} L(u'')$ que apresenta o maior número de cores distintas, onde u' e u'' são quaisquer dois filhos de u (linha 18 do algoritmo).

Convergência e Complexidade

Os resultados a seguir mostram a complexidade e a convergência do algoritmo *Tropical Cacto*. A ideia nesse contexto é mostrar que a execução do algoritmo não permite loop infinito e leva tempo polinomial.

Lema 5.2.9 *O algoritmo* Calcula Rótulos *é executado em tempo linear* $\Theta(|V(G)|)$.

Prova O algoritmo é inicialmente executado para a raiz u. Na linha 4 o algoritmo é chamado recursivamente para cada filho u' de u que, ou será uma ponte ou folha, ou haverá um subgrafo cacto enraizado em u', do qual, claramente, u não faz mais parte, ou seja, não há possibilidade de que u venha aparecer novamente numa próxima chamada recursiva, logo, o número total de chamadas recursivas do algoritmo é a somatória do número de vértices do grafo. Adicionalmente, note que as operações das linhas 1 e 2 e das linhas 6-18 levam tempo $O(1)$ o que mostra que o algoritmo possui complexidade linear no tamanho da entrada e portanto, converge. Apesar de haver um laço nas linhas 6-15 esse laço só será executado uma vez para cada vértice do grafo.

Proposição 5.2.10 *O algoritmo* Tropical Cacto *é executado em tempo polinomial* $\Theta\left(|V(G)|^2\right)$

Prova Sabe-se que o algoritmo *Tropical Cacto* faz $|V(G)|$ chamadas ao algoritmo *Calcula Rótulos*, sendo um para cada possível raiz, e pelo Lema 5.2.9 o algoritmo *Calcula Rótulos* funciona em tempo $\Theta(|V(G)|)$, portanto o resultado segue.

6 Análise empírica

Este capítulo apresenta uma análise empírica dos resultados produzidos por esta dissertação. Uma análise foi realizada para investigar como a variação do número de vértices, arestas e cores impactam no desempenho do modelo de Programação Linear Inteira para o problema MTPP para grafos simples em geral e sua simplificação para o caso específico de grafos dirigidos acíclicos, ambos NP-completos. O fator principal de investigação é o tempo de execução destes modelos, mas outros fatores importantes também são discutidos.

Também neste capítulo, uma análise empírica dos resultados obtidos pela heurística proposta é apresentada em comparação com a solução ótima obtida através do modelo PLI. A análise do algoritmo heurístico se baseia principalmente na comparação entre os valores ótimos e os tempos de execução.

Tais análises ocorrem sobre os resultados de experimentos computacionais executados sobre instâncias MTPP geradas aleatoriamente e instâncias tratadas e derivadas de conjuntos de dados reais.

6.1 Metodologia dos experimentos

Os experimentos foram realizados tanto em instâncias geradas aleatoriamente quanto em instâncias derivadas de conjuntos de dados reais. As instâncias reais representam, por exemplo, interações entre proteínas, mapeamento de doenças, escalonamento de tarefas e interações sociais. O uso variável de instâncias reais e aleatórias é um procedimento vantajoso na análise do comportamento dos resultados. O uso de instâncias geradas aleatoriamente permite ter um número maior de instâncias que é possível ter o controle da variação das características do grafo, como o número de vértices, arestas e cores. Isso possibilita analisar o impacto de cada característica nos experimentos computacionais. Por outro lado, o uso de instâncias reais possibilita visualizar como os modelos e algoritmos possivelmente podem se comportar em problemas reais.

Instâncias de grande porte, neste contexto, são instâncias com um grande número de vértices e arestas e que não são obtidas soluções inteiras pelo pacote de otimização em um limite de tempo pré-definido e que dificilmente seriam resolvidas por qualquer

outro método devido ao número exponencial de restrições criadas pelo modelo. O limite de tempo de execução para esses experimentos foi definido como 3600 segundos.

Ao utilizar experimentos computacionais para avaliar modelos PLI e algoritmos, é possível avaliar várias características cruciais para a análise da qualidade dos mesmos. No caso de modelos PLI, é possível avaliar a qualidade do modelo por meio dos resultados do tempo de execução, tamanho e crescimento do modelo. Para algoritmos heurísticos, é possível avaliar sua qualidade por meio do cálculo da diferença entre seus resultados e a solução ótima para uma determinada instância, além de seu tempo de execução.

Existem três parâmetros das instâncias deste problema que podem ser variados: o número de vértices, o número de arestas e o número de cores. Para uma análise mais eficaz do impacto da variação desses parâmetros em uma instância do problema, foram utilizadas instâncias geradas aleatoriamente com valores predefinidos para os parâmetros. Tanto a obtenção e interpretação das instâncias reais quanto a geração das instâncias aleatórias são descritas mais adiante neste capítulo. Para consolidar os resultados, todas as instâncias foram executadas 10 vezes. Isso garante a integridade dos resultados, pois impede que resultados incorretos, devido a possíveis problemas no ambiente computacional, sejam considerados válidos. Os resultados finais de cada instância são definidos pela média dos resultados válidos. Para uma análise mais precisa do modelo, os experimentos consideraram apenas o tempo de resolução de cada modelo, ignorando o tempo gasto para carregar o modelo no pacote de otimização.

Esta seção apresenta as configurações do ambiente computacional onde os modelos PLI e os algoritmos foram implementados e os experimentos computacionais foram executados. Além disso, descreve como as instâncias utilizadas nos experimentos computacionais foram geradas, no caso de instâncias aleatórias, e como as instâncias reais foram obtidas e interpretadas para representar uma instância do problema.

Detalhes do ambiente de implementação e execução

Tanto os modelos PLI quanto os algoritmos da heurística foram implementados na linguagem de programação *C#*. Os experimentos foram realizados em um ambiente computacional com as seguintes configurações: processador Intel(R) Core (TM) i7-8550U

CPU 1,99 GHz, 8GB de RAM e sistema operacional Windows 10 Home versão 20H2. Para a implementação dos modelos PLI, foi utilizado o software de otimização comercial Gurobi Optmizer versão 9.5.0. Os parâmetros do Gurobi foram configurados com os seus respectivos valores padrão.

Instâncias reais

As instâncias reais foram obtidas através de um repositório de conjuntos de dados (GEPHI, 2009). Alguns dos conjuntos de dados formam redes com características diferentes das instâncias do problema estudado, portanto, para o uso desses conjuntos de dados, algumas adaptações foram realizadas. A seguir estão descritas como ocorreu a obtenção, interpretação e adaptação das instâncias reais. Mais adiante nesta seção, será explicada a atribuição de colorações em instâncias que não apresentam características claras que possam ser consideradas como colorações.

- O conjunto de dados intitulado "*Les Miserables*"é uma rede ponderada de co-aparição de personagens no romance Les Miserables, musical francês composto por Claude-Michel Schönberg em 1980, com libreto de Alain Boublil e letras de Herbert Kretzmer. É um dos musicais mais famosos e encenados em todo o mundo. Essa rede foi apresentada no trabalho de Knuth (1993). A rede possui 77 nós (representando os personagens) e 254 arestas (representando o encontro entre os personagens correspondentes).
- Foi dado o nome "*Primary*"a uma rede que faz parte de um estudo de redes de contato em uma escola primária, relatado por Stehlé *et al.* (2011). O conjunto de dados compreende duas redes de proximidade face a face entre alunos e professores. São fornecidas duas redes, uma para cada dia de estudo. A rede referente ao primeiro dia possui 236 vértices e 5899 arestas, enquanto a rede referente ao segundo dia possui 238 vértices e 5539 arestas. Os vértices representam os indivíduos e as arestas representam as interações face a face entre os indivíduos. Os vértices possuem o atributo "classe"que foi definido como a coloração dos vértices do grafo, gerando assim uma coloração com 11 cores para ambos os dias. Um exemplo de caminho tropical dessa coloração pode ser um caminho que interliga todas as turmas por apenas um individuo por turma.

- Foi escolhida uma rede direcionada e ponderada que representa a rede neural de C. Elegans, a qual será denominada como "*C.Eleg*". Sabe-se que entre os sistemas nervosos, essa é a rede neural mais simples. Vários comportamentos essenciais, como o movimento, dependem desse tipo de rede, e essas características simplistas fazem dessa rede neural um objeto amplamente investigado (WHITE *et al.*, 1986). Originalmente, a rede possui 306 vértices e 2359 arestas. Após a remoção de vértices isolados, arestas paralelas e loops, a rede ficou com 297 vértices e 2148 arestas. Os pesos contidos nas arestas trazem informações sobre algum grau de associação entre dois objetos, mas não foram considerados para o problema investigado. Embora nenhuma coloração tenha sido extraída do conjunto de dados apresentado, é sabido que uma análise detalhada das informações contidas no banco de dados pode levar a uma coloração de interesse e qualquer subestrutura desta instância pode fornecer informações relevantes sobre sua composição.
- A rede chamada CPAN Explorer ou apenas "*CPAN*"é um projeto de visualização que analisa as relações entre desenvolvedores de pacotes na linguagem Perl. Duas redes são derivadas desse projeto chamadas de "*CPAN Authors*"e "*CPAN Distributions*". A primeira é uma rede de desenvolvedores vinculados pelo uso do mesmo módulo Perl, e a segunda é uma rede de dependências dos módulos Perl. Os dados originais são encontrados online, sem vínculo a trabalhos acadêmicos (LINKFLUENCE, 2009). A rede *CPAN Authors* originalmente possui 840 vértices (referentes aos desenvolvedores) e 2138 arestas (representando o vinculo entre o uso do mesmo módulo entre os desenvolvedores), mas após adaptações, passou a ter 839 vértices e 2112 arestas. Quatro características diferentes foram escolhidas para definir colorações. A rede *CPAN Distributions* originalmente possuía 2724 vértices e 5426 arestas, mas após adaptações, passou a ter 2719 vértices e 5016 arestas.
- Outra rede real utilizada é um mapeamento de interações entre atores da Science in Society na Web de 12 países europeus, chamada "*EuroSIS*"(referência online encontrada em (EUROSIS, 2009), sem vinculo a trabalho acadêmico). Ela possui 1285 vértices (representando os autores) e 7524 arestas (representando as interações entre os atores). Após remover arestas paralelas e loops, a rede ficou com 6462 arestas. Os países e instituições de origem dos autores foram definidos como colorações, gerando colorações com 13 e 22 cores respectivamente.

- A rede chamada *Diseasome* é uma rede de doenças e genes de doenças ligados por associações genéticas conhecidas, indicando a origem genética comum de muitas doenças. O conjunto de dados original foi desenvolvido por um estudo apresentado por Goh *et al.* (2007). Esta rede tem 1419 vértices (representando uma doença, distúrbio ou gene), 3926 arestas (que são as associações distúrbio-genético ou a associação entre doenças), após remover arestas paralelas e loops, a rede ficou com 2738 arestas. Um rótulo foi definido para cada classe de doença, gerando uma coloração com 23 cores.
- A rede chamada *Yeast*, é uma rede de interação proteína-proteína. As leveduras são organismos unicelulares com alto potencial de proliferação e crescimento. Algumas espécies de leveduras são patógenos oportunistas que podem causar infecção em pessoas com sistema imunológico comprometido. Tal rede foi obtida através do trabalho de Bu *et al.* (2003). Esta rede possui 2361 proteínas representadas pelos vértices e 7182 interações representadas pelas arestas. Após a remoção de vértices isolados, arestas paralelas e loops, a rede ficou com 2284 vértices e 6646 arestas. Também pelo fato de que qualquer subestrutura desta instância pode fornecer informações relevantes sobre sua composição, três colorações aleatórias foram definidas seguindo o padrão de 10%, 50% e 90% do número de vértices da rede.

Para o estudo do impacto da variação do número de cores na instância, foram estabelecidas três colorações aleatórias, com o padrão de 10%, 50% e 90% do número de vértices da rede, para todas as redes que não possuem uma coloração.

A partir das redes reais descritas acima, foram geradas 21 instâncias, cujos detalhes são apresentados na Tabela 1 que identifica as instâncias reais de grafos. Cada instância real é rotulada com um índice para identificação, seguindo o modelo R_i, sendo i um inteiro atribuído à instância. A Tabela 1 é composta pelas colunas Id, Nome, número de vértices ($|V|$), número de arestas ($|E|$) e número de cores ($|C|$).

Tabela 1 – Identificação das instâncias reais

Id	Nome	$\|V\|$	$\|E\|$	$\|C\|$
R_1	Les Miserables$_1$	77	254	8
R_2	Les Miserables$_2$	77	254	39
R_3	Les Miserables$_3$	77	254	70
R_4	Primary$_1$	236	5899	11
R_5	Primary$_2$	238	5539	11
R_6	C.Eleg$_1$	297	2148	30
R_7	C.Eleg$_2$	297	2148	149
R_8	C.Eleg$_3$	297	2148	268
R_9	CPAN Authors$_1$	839	2112	32
R_{10}	CPAN Authors$_2$	839	2112	60
R_{11}	CPAN Authors$_3$	839	2112	68
R_{12}	CPAN Authors$_4$	839	2112	310
R_{13}	EuroSIS$_1$	1285	6462	13
R_{14}	EuroSIS$_2$	1285	6462	22
R_{15}	Diseasome	1419	2738	25
R_{16}	Yeast$_1$	2284	6646	228
R_{17}	Yeast$_2$	2284	6646	1142
R_{18}	Yeast$_3$	2284	6646	2056
R_{19}	CPAN Distributions$_1$	2719	5016	272
R_{20}	CPAN Distributions$_2$	2719	5016	1360
R_{21}	CPAN Distributions$_3$	2719	5016	2448

Fonte – Igor Sampaio, 2023

Instâncias aleatórias

Devido às limitações nas instâncias reais em relação à variação no número de vértices, cores e arestas, foi proposto a geração de grafos aleatórios. Para investigar o comportamento do modelo em um grafo com número fixo de vértices, variando o número de arestas e cores, o número de vértices foi fixado em n e foram geradas quatro variações no número de arestas:

- primeira variação de instâncias contêm $n + (n/10)$ arestas, gerando um grafo muito esparso;
- a segunda variação tem $2n$ arestas, gerando um grafo esparso comum;
- a terceira tem $(n * (n - 1))/4$ arestas, com um quarto da quantidade de arestas de um grafo completo, gerando um grafo denso;
- e a última variação tem $(n * (n - 1))/3$ arestas, com um terço da quantidade de arestas de um grafo completo, gerando um grafo mais denso.

Essas variações foram escolhidas pois oferecem características interessantes em relação às instâncias reais, as quais podem ser úteis para entender melhor o problema. Para cada combinação de grafo com n vértices, são geradas três variações de quantidade de cores usadas na coloração do grafo representando 10%, 50% e 90% do número de vértices.

Devido às limitações no desempenho do modelo PLI, notou-se que ele começa a atingir o limite de tempo em instâncias com cerca de 800 vértices. Dessa forma, neste trabalho serão consideradas instâncias com 200, 400, 600 e 800 vértices para fins comparativos, apesar de algumas instâncias específicas maiores possam ser resolvidas dentro do limite de tempo estabelecido.

Seguindo a descrição das variações de número de vértices, arestas e cores, 48 instâncias aleatórias serão apresentadas nesta análise. Essas instâncias são descritas na Tabela 2. A Tabela 2 identifica as instâncias aleatórias de grafos que serão identificadas no decorrer do trabalho por um índice, o índice segue o modelo A_{ikj} sendo i um inteiro atribuído para cada variação no número de vértices, k um inteiro atribuído para cada variação no número de cores e j um inteiro atribuído para cada variação no número de arestas. A tabela para identificação das instâncias esta agrupada pelo número de vértices $|V|$ e cada conjunto e composto pelas colunas: Id, número de cores $|C|$, e número de arestas $|E|$.

Tabela 2 – Identificação das instâncias aleatórias

200			400			600			800		
Id	$\|C\|$	$\|E\|$	Id	$\|C\|$	$\|E\|$	Id	$\|C\|$	$\|E\|$	Id	$\|C\|$	$\|E\|$
A_{111}	20	220	A_{211}	40	440	A_{311}	60	660	A_{411}	80	880
A_{112}	20	400	A_{212}	40	800	A_{312}	60	1200	A_{412}	80	1600
A_{113}	20	9950	A_{213}	40	39900	A_{313}	60	89850	A_{413}	80	159800
A_{114}	20	13266	A_{214}	40	53200	A_{314}	60	119800	A_{414}	80	213066
A_{121}	100	220	A_{221}	200	440	A_{321}	300	660	A_{421}	400	880
A_{122}	100	400	A_{222}	200	800	A_{322}	300	1200	A_{422}	400	1600
A_{123}	100	9950	A_{223}	200	39900	A_{323}	300	89850	A_{423}	400	159800
A_{124}	100	13266	A_{224}	200	53200	A_{324}	300	119800	A_{424}	400	213066
A_{131}	180	220	A_{231}	360	440	A_{331}	540	660	A_{431}	720	880
A_{132}	180	400	A_{232}	360	800	A_{332}	540	1200	A_{432}	720	1600
A_{133}	180	9950	A_{233}	360	39900	A_{333}	540	89850	A_{433}	720	159800
A_{134}	180	13266	A_{234}	360	53200	A_{334}	540	119800	A_{434}	720	213066

Fonte – Igor Sampaio, 2023

Instâncias de Grafos Dirigidos Acíclicos - DAGs

Para avaliar a performance do modelo PLI simplificado para DAGs, instâncias do tipo DAG foram extraídas de todas as instâncias aleatórias e reais. Uma associação foi feita para cada instância original e a direção de cada aresta foi escolhida aleatoriamente. Para evitar ciclos, um ordenamento dos vértices foi estabelecido, de modo que, se houver uma aresta (a, b), então $a < b$. Assim, as instâncias DAG seguem a mesma identificação das instâncias de um modo geral.

Instâncias de Grafos Cacto

Pelo fato de as instâncias de grafos cacto terem uma estrutura diferente das instâncias gerais, foram geradas instâncias específicas para esse fim. A variação do número de vértices e cores segue o mesmo padrão explicado para grafos simples, mas a variação no número de arestas segue o número de circuitos da instância. Devido ao fato de grafos cacto não permitirem uma variação muito grande na densidade do grafo, optou-se por apresentar duas variações em relação a quantidade de arestas e, consequentemente, em relação à quantidade de circuitos.

Para esta análise serão apresentadas 24 instâncias aleatórias. As instâncias são identificadas na Tabela 3 com um índice designado a cada instância, no formato A^c_{ikj}, onde i é um inteiro atribuído para cada variação no número de vértices, k é um inteiro atribuído para cada variação no número de cores e j é um inteiro atribuído para cada variação no número de ciclos. A tabela é composta pelas colunas Id, número de vértices $|V|$, número de arestas $|E|$, número de cores $|C|$ e número de ciclos $|Ciclos|$.

Tabela 3 – Identificação das instâncias aleatórias de grafos cacto

Id	$\|V\|$	$\|C\|$	$\|E\|$	$\|Ciclos\|$
A^c111	200	20	209	10
A^c112	200	20	219	20
A^c121	200	100	209	10
A^c122	200	100	219	20
A^c131	200	180	209	10
A^c132	200	180	219	20
A^c211	400	40	419	20
A^c212	400	40	439	40
A^c221	400	200	419	20
A^c222	400	200	439	40
A^c231	400	270	419	20
A^c232	400	270	439	40
A^c311	600	60	629	30
A^c312	600	60	659	60
A^c321	600	300	629	30
A^c322	600	300	659	60
A^c331	600	540	629	30
A^c332	600	540	659	60
A^c411	800	80	839	40
A^c412	800	80	879	80
A^c421	800	400	839	40
A^c422	800	400	879	80
A^c431	800	720	839	40
A^c432	800	720	879	80

Fonte – Igor Sampaio, 2023

6.2 Experimentos computacionais

Esta seção apresenta os resultados obtidos pelo modelo PLI e pelos algoritmos implementados para o MTPP. Uma análise será feita dos resultados para cada implementação.

Os experimentos computacionais registram o tempo de execução retornado pelo atributo *Runtime* do Gurobi Optimizer. Esse atributo retorna o tempo de execução da otimização mais recente em segundos. É importante destacar que todos os tempos relatados pelo Gurobi Optimizer são *wall-clock times*, que representa o tempo total gasto para executar uma tarefa, incluindo possíveis interrupções. Uma interrupção pode ocorrer porque o processador divide o uso com outras threads, precisa aguardar um dispositivo externo ou o algoritmo está em espera sem executar nada.

Outro parâmetro registrado pelos experimentos computacionais é o trabalho gasto, obtido pelo atributo *Work* do Gurobi Optimizer. Esse atributo retorna o trabalho gasto na otimização mais recente. Ao contrário do *Runtime*, o atributo *Work* é determinístico, ou seja, o resultado será sempre o mesmo desde que o modelo seja resolvido no mesmo hardware com as mesmas configurações de parâmetros e atributos. As unidades dessa métrica são arbitrárias e uma unidade de trabalho equivale aproximadamente a um segundo em um único thread, mas esse valor pode variar de acordo com o hardware utilizado e o modelo resolvido. Esse valor está relacionado ao *CPU Time*, que é o tempo que o processador gasta para executar uma tarefa, sem contar interrupções.

Conjunto de resultados 1 - Instâncias aleatórias de grafos simples em geral

As Tabelas de 4 a 7 apresentam os resultados dos experimentos computacionais do modelo PLI e do algoritmo para grafos simples em geral em instâncias aleatórias e estão separadas por instâncias com o mesmo número de vértices. Em todas as tabelas, as instâncias são identificadas pela coluna Id (identificador da instância), seguida dos dados da execução do modelo PLI representados pelas colunas $|RL|$ referente ao número de restrições lineares do modelo PLI, T_{PLI} referente ao tempo de execução em segundos do modelo PLI, W_{PLI} referente ao trabalho gasto pelo modelo PLI e Sol_{PLI} referente ao valor da solução ótima encontrada pelo modelo PLI. Para finalizar é apresentado as colunas (T_{Alg}) referente ao tempo de execução em segundos do algoritmo heurísticos, (Sol_{Alg}) referente ao valor da solução encontrada pelo algoritmo heurístico, (GAP_T) referente a diferença entre o tempo de execução do modelo PLI e do algoritmo em porcentagem e (GAP_Sol) referente a diferença entre a solução ótima e a solução encontrada pelo algoritmo heurístico em porcentagem.

Nas instâncias com 200 vértices, todas as soluções ótimas foram encontradas em menos de 10 segundos. Em geral, instâncias com maior número de arestas e cores têm maior tempo de execução e trabalho gasto. O algoritmo encontrou soluções com diferença de no máximo cerca de 30% em relação à solução ótima, sendo que instâncias com menos cores tenderam a encontrar a solução ótima. Além disso, o tempo de execução do algoritmo foi significativamente mais rápido que o modelo PLI nas instâncias testadas.

Tabela 4 – Resultados dos experimentos computacionais (Instâncias com 200 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
A_{111}	861	1,32	0,79	20	0,06	20	95,25%	0,00%
A_{112}	1221	1,72	0,98	20	0,02	20	98,58%	0,00%
A_{113}	20321	1,71	1,21	20	0,02	20	99,06%	0,00%
A_{114}	26953	2,63	1,93	20	0,02	20	99,31%	0,00%
A_{121}	941	6,11	4,49	61	0,20	48	96,74%	21,31%
A_{122}	1301	1,45	0,64	100	0,37	83	74,60%	17,00%
A_{123}	20401	3,25	3,09	100	0,29	95	91,12%	5,00%
A_{124}	27033	9,05	7,80	100	0,46	97	94,96%	3,00%
A_{131}	1021	3,99	1,91	67	0,16	49	96,09%	26,87%
A_{132}	1381	2,25	1,09	171	0,57	123	74,82%	28,07%
A_{133}	20481	4,03	3,91	180	0,24	164	93,97%	8,89%
A_{134}	27113	6,25	6,29	180	0,33	172	94,79%	4,44%

Fonte – Igor Sampaio, 2023

Tabela 5 – Resultados dos experimentos computacionais (Instâncias com 400 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{DAG}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
A_{211}	1721	19,35	12,49	40	1,07	39	94,48%	2,50%
A_{212}	2441	10,24	8,00	40	0,12	40	98,82%	0,00%
A_{213}	80641	12,56	11,94	40	0,14	40	98,87%	0,00%
A_{214}	107241	14,67	13,53	40	0,37	40	97,47%	0,00%
A_{221}	1881	37,65	26,83	111	1,32	89	96,50%	19,82%
A_{222}	2601	48,19	32,41	198	6,89	166	85,70%	16,16%
A_{223}	80801	201,56	302,60	200	1,86	184	99,08%	8,00%
A_{224}	107401	47,13	58,78	200	2,50	198	94,69%	1,00%
A_{231}	2041	6,06	4,45	140	1,13	104	81,40%	25,71%
A_{232}	2761	329,34	253,57	343	5,87	238	98,22%	30,61%
A_{233}	80961	511,23	784,51	360	2,17	317	99,57%	11,94%
A_{234}	107561	319,54	581,44	360	2,81	335	99,12%	6,94%

Fonte – Igor Sampaio, 2023

Ao analisar as instâncias com 400 vértices, não se observou uma relação constante entre o número de arestas e cores com o tempo de execução e trabalho gasto. No entanto, o algoritmo manteve os mesmos pontos observados nas instâncias com 200 vértices.

Ao analisar as instâncias com 600 vértices, observa-se novamente a relação entre o aumento do número de arestas e cores e o aumento do tempo de execução e trabalho gasto. Esse conjunto de instâncias apresentou a primeira ocorrência em que o tempo limite de execução foi atingido. O algoritmo, por sua vez, continua a apresentar os mesmos pontos observados nas instâncias anteriores, mas com um aumento no pico para cerca de 40% entre a solução ótima e a solução encontrada pelo algoritmo.

Tabela 6 – Resultados dos experimentos computacionais (Instâncias com 600 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{DAG}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
A_{311}	2581	41,19	29,24	60	3,43	58	91,67%	3,33%
A_{312}	3661	43,88	34,15	60	0,29	60	99,34%	0,00%
A_{313}	180961	39,08	37,18	60	0,36	60	99,08%	0,00%
A_{314}	240861	41,91	46,59	60	0,41	60	99,02%	0,00%
A_{321}	2821	44,42	27,24	168	4,97	135	88,80%	19,64%
A_{322}	3901	134,80	120,96	292	31,09	239	76,94%	18,15%
A_{323}	181201	361,62	695,08	300	7,74	292	97,86%	2,67%
A_{324}	241101	582,18	947,82	300	10,43	296	98,21%	1,33%
A_{331}	3061	42,78	30,17	209	2,80	128	93,46%	38,76%
A_{332}	4141	1827,28	1605,42	501	21,66	348	98,81%	30,54%
A_{333}	181441	2955,88	3026,47	540	9,07	490	99,69%	9,26%
A_{334}	241341	3601,75	3751,88	539	11,27	501	-	-

Fonte – Igor Sampaio, 2023

Tabela 7 – Resultados dos experimentos computacionais (Instâncias com 800 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{DAG}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
A_{411}	3441	395,11	167,41	80	6,95	74	98,24%	7,50%
A_{412}	4881	46,12	28,05	80	1,78	80	96,14%	0,00%
A_{413}	321281	110,94	70,36	80	0,77	80	99,31%	0,00%
A_{414}	427813	144,46	92,30	80	0,98	80	99,32%	0,00%
A_{421}	3761	372,76	175,57	234	6,45	151	98,27%	35,47%
A_{422}	5201	760,51	560,51	393	26,05	299	96,58%	23,92%
A_{423}	321601	2273,05	2292,54	400	24,10	393	98,94%	1,75%
A_{424}	428133	3601,89	6094,00	400	30,02	388	-	-
A_{431}	4081	66,43	50,13	269	5,51	166	91,70%	38,29%
A_{432}	5521	2504,11	3348,03	672	28,37	417	98,87%	37,95%
A_{433}	321921	1914,03	3911,14	720	26,53	638	98,61%	11,39%
A_{434}	428453	2168,50	3923,89	720	39,87	679	98,16%	5,69%

Fonte – Igor Sampaio, 2023

As instâncias com 800 vértices concluem o conjunto de instâncias aleatórias. Dentre os pontos destacados nas instâncias anteriores, foi possível observar que as instâncias com maior número de arestas, dentro do conjunto de instâncias com mesma quantidade de cores, em geral, apresentam um tempo de execução e trabalho gasto maior. Este conjunto de instâncias apresentou a segunda ocorrência que atingiu o tempo limite de execução, por esse motivo foi decidido não realizar experimentos com instâncias com mais vértices. O algoritmo, concluindo a consistência dos resultados, manteve os mesmos pontos observados nas instâncias anteriores, porém também atingindo um pico de cerca de 40% entre a solução ótima e a solução encontrada pelo algoritmo.

Análise do modelo PLI para grafos simples em geral em instâncias aleatórias

Em uma análise inicial, é possível perceber o comportamento do modelo proposto em relação ao crescimento do número de restrições lineares adicionadas ao modelo. Isso pode ser observado por meio de experimentos computacionais. A Figura 10 apresenta a relação entre o aumento do número de restrições lineares no modelo e o aumento na quantidade de vértices do modelo.

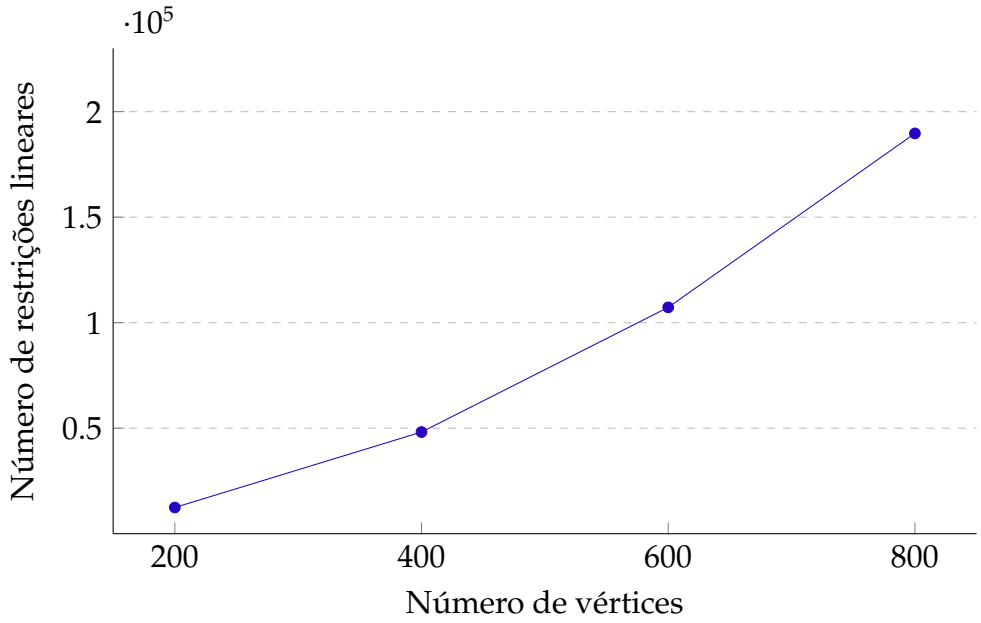


Figura 10 – Relação entre o crescimento do número de restrições lineares no modelo e o número de vértices da instância.

Com base na figura e nos dados apresentados, é possível notar que o número de restrições segue a formula $|C| + 2|V| + |A| + 1$, onde C é o número de cores, V é o número de vértices e a A é o número de arcos da instância. Esse resultado foi constante para todas as instâncias executadas e sugere que o número de restrições lineares no modelo aumenta de forma polinomial em relação ao número de vértices, o que pode ser um indicador empírico de boa formulação do modelo.

Devido ao limite de tempo de execução, foi escolhido executar instâncias de até 800 vértices. Embora algumas das instâncias executadas tenham atingido o limite de tempo de execução de 1 hora (3600 segundos), um pouco mais de 77% das instâncias executadas obtiveram soluções ótimas inteiras em tempo inferior a 7 minutos (420 segundos), o que representa um tempo de execução pelo menos 88% menor que o limite estipulado. Isso demonstra que é possível encontrar soluções inteiras ótimas para

instâncias ainda maiores que as executadas nos experimentos computacionais deste trabalho, dependendo das características dessas instâncias. Portanto, para o conjunto de testes apresentados, os resultados obtidos em termos de tempo computacional gasto foram bastante satisfatórios, considerando que se trata de um problema NP-difícil.

Devido ao limite de tempo de execução, foi escolhido executar instâncias de até 800 vértices. Embora algumas das instâncias executadas tenham atingido o limite de tempo de execução de 1 hora (3600 segundos), mais de 77% das instâncias executadas obtiveram soluções ótimas inteiras em menos de 7 minutos (420 segundos), representando um tempo de execução pelo menos 88% menor que o limite estipulado. Para o conjunto de testes apresentado, os resultados obtidos em termos de tempo computacional foram bastante satisfatórios, considerando que se trata de um problema NP-difícil.

A relação entre o crescimento do tempo de execução em segundos e do número de vértices da instância, é demonstrada na figura 11 a seguir.

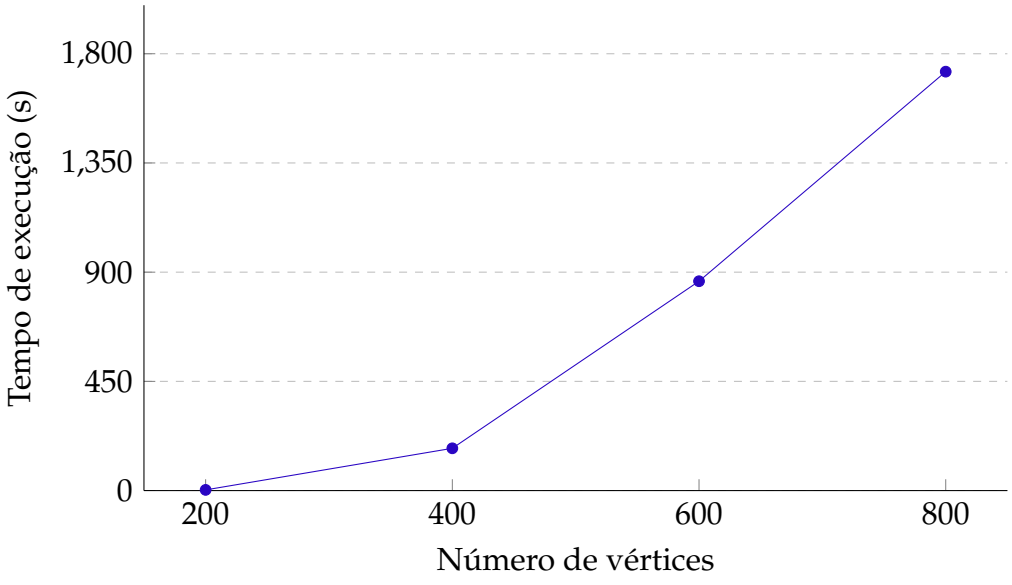


Figura 11 – Relação entre o crescimento do número de vértices da instância e seu tempo de execução.

Ao analisar as tabelas e o gráfico da Figura 11, é possível concluir que a quantidade de instâncias com tempo de execução significativamente acima da média aumenta à medida que o número de vértices da instância aumenta. Isso sugere que o tempo de execução cresce proporcionalmente ao número de vértices. Esse crescimento está aparentemente relacionado ao aumento do número de restrições lineares no modelo, como visto no gráfico da Figura 10.

É possível concluir que o número de vértices exerce um impacto muito mais significativo no tempo de execução do que o número de cores e arestas. No entanto, pode-se observar que, nos dados apresentados, tanto o aumento no número de cores quanto o aumento do número de arestas apresentam uma tendência a uma relação diretamente proporcional ao tempo de execução e ao trabalho gasto, onde, na maioria dos casos, quanto mais cores e arestas na instância, maior é o tempo de execução. No entanto, é importante destacar que algumas exceções se destacam dessa tendência.

Análise do algoritmo heurístico para o MTPP para grafos simples em geral em instâncias aleatórias

Para o problema MTPP em grafos simples, a heurística inicia a execução pelo algoritmo *Grafo Cacto* e, em seguida, pelo algoritmo *Tropical Cacto*. Neste sentido, dois fatores principais relacionados ao desempenho do algoritmo em relação aos valores ótimos serão avaliados: o tempo de execução e a precisão dos resultados obtidos.

Um ponto importante na avaliação do desempenho do algoritmo é avaliar a relação entre a solução ótima e a solução apresentada pelo algoritmo. Para isso, é possível observar o gráfico da Figura 12.

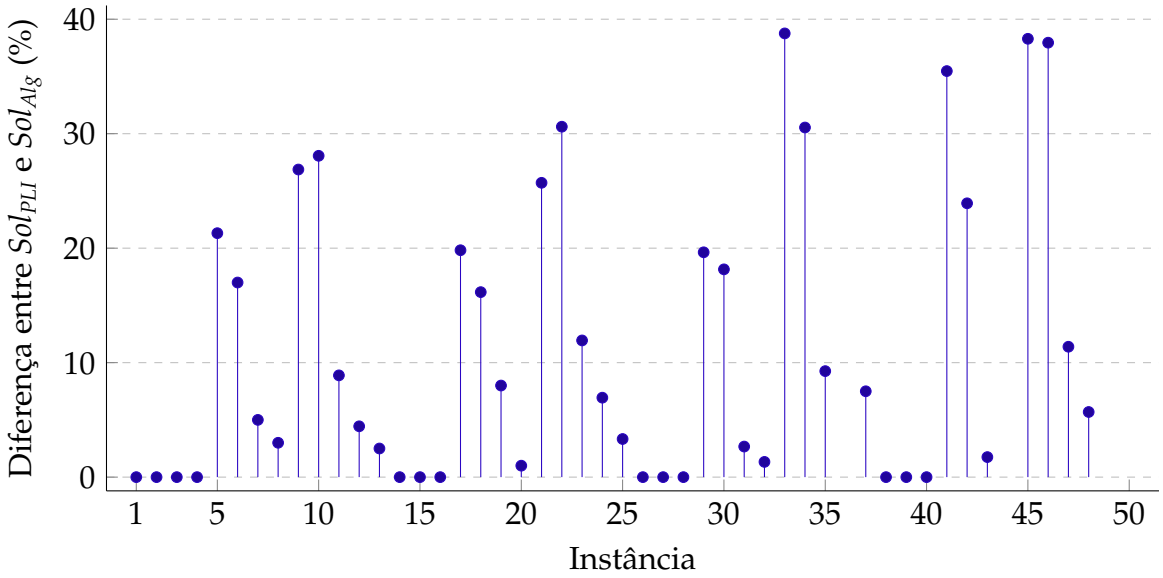


Figura 12 – Diferença entre a solução ótima e a solução do algoritmo para o MTPP para grafos simples em geral em instâncias aleatórias (%)

No gráfico da Figura 12, que apresenta a diferença entre a solução ótima e a solução encontrada pelo algoritmo, é importante destacar que, das instâncias que não

atingiram o tempo limite de execução, apenas cerca de 28% encontraram o valor ótimo. Porém, no pior caso, a diferença entre a solução ótima e a solução apresentada pelo algoritmo foi inferior a 39% entre os valores. Analisando tais resultados, é possível concluir que a heurística apresentada, em geral, bons limitantes inferiores em relação à solução ótima para o problema.

Ao observar o tempo de execução entre as instâncias que não atingiram o limite de tempo no modelo PLI, é óbvio notar que o tempo de execução do algoritmo apresenta uma velocidade muito superior. No algoritmo, o pior caso apresenta um tempo de execução cerca de 74% menor que no modelo PLI, e em quase 87% das instâncias, o tempo de execução do algoritmo foi ao menos 90% menor que no modelo PLI. Isso é uma prova empírica da ótima velocidade do algoritmo. O algoritmo mostrou-se muito eficiente em relação ao tempo de execução.

Conjunto de resultados 2 - Instâncias reais de grafos simples em geral

A Tabela 8 apresenta os resultados dos experimentos computacionais com o modelo PLI e o algoritmo para grafos simples em geral, aplicados em instâncias reais. Na tabela, as instâncias são identificadas pela coluna "Id"(identificador da instância), seguida dos dados da execução do modelo PLI representados pelas colunas "$|RL|$"(número de restrições lineares do modelo PLI), "T_{PLI}"(tempo de execução em segundos do modelo PLI), "W_{PLI}"(trabalho gasto pelo modelo PLI) e "Sol_{PLI}"(valor da solução ótima encontrada pelo modelo PLI). Por fim as colunas "T_{Alg}"(tempo de execução em segundos do algoritmo heurísticos), "Sol_{Alg}"(valor da solução encontrada pelo algoritmo heurístico), "GAP_T"(diferença entre o tempo de execução do modelo PLI e do algoritmo em porcentagem) e "GAP_Sol"(diferença entre a solução ótima e a solução encontrada pelo algoritmo heurístico em porcentagem).

Análise do modelo PLI para grafos simples em geral em instâncias reais

Ao analisar os resultados dos experimentos computacionais do modelo PLI para grafos simples em geral em instâncias reais, um primeiro ponto a considerar é que, assim como nas instâncias aleatórias, o número de restrições lineares aumenta de acordo com

Tabela 8 – Resultados dos experimentos computacionais (Instâncias Reais)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
R_1	671	0,72	0,19	8	0,00	8	99,60%	0,00%
R_2	702	2071,25	1279,03	32	0,02	20	100,00%	37,50%
R_3	733	143,73	74,02	50	0,02	26	99,99%	48,00%
R_4	12282	1,23	0,64	11	0,02	11	98,58%	0,00%
R_5	11566	1,10	0,64	11	0,02	11	98,62%	0,00%
R_6	4921	4,49	1,81	30	0,17	30	96,21%	0,00%
R_7	5040	3600,43	2520,99	147	1,86	130	-	-
R_8	5159	1183,24	770,74	254	1,77	197	99,85%	22,44%
R_9	5935	5,33	2,63	31	4,43	22	16,93%	29,03%
R_{10}	5963	16,65	10,67	56	4,41	36	73,53%	35,71%
R_{11}	5971	16,83	11,70	67	4,35	45	74,13%	32,84%
R_{12}	6213	31,33	30,98	143	4,56	62	85,46%	56,64%
R_{13}	15508	22,69	10,73	13	5,28	13	76,74%	0,00%
R_{14}	15517	70,93	45,61	22	5,37	22	92,43%	0,00%
R_{15}	8340	3601,63	4054,24	22	16,19	18	-	-
R_{16}	18089	1291,21	1428,66	228	254,77	218	80,27%	4,39%
R_{17}	19003	3605,61	3319,90	778	261,84	537	-	-
R_{18}	19917	3607,21	3240,26	1013	263,13	654	-	-
R_{19}	15743	3601,35	3835,90	218	107,37	127	-	-
R_{20}	16831	3621,91	3553,49	328	109,06	165	-	-
R_{21}	17919	3623,42	2558,56	340	98,30	170	-	-

Fonte – Igor Sampaio, 2023

um valor proporcional a n^2, onde n é o número de vértices da instância. Esse resultado confirma a consistência deste fator entre as instâncias aleatórias e as instâncias reais.

Entre as instâncias reais, há instâncias com variação muito maior no número de vértices do que nas instâncias aleatórias, incluindo instâncias com até 2719 vértices. Por esse motivo, a maioria das instâncias com mais de 1000 vértices atingiu o tempo limite de execução.

Além disso, pode-se destacar que, entre as instâncias em que foi possível obter a solução ótima dentro do tempo de execução limite, o número de vértices, arestas e cores não são os únicos fatores que afetam o tempo de execução e o trabalho gasto. Possivelmente, a estrutura do grafo também interfere nesses fatores. Por exemplo, a instância R_2 apresentou tempo de execução de 2071,25 segundos e trabalho gasto de 1279,03, mesmo com apenas 77 vértices, 254 arestas e 39 cores.

Análise do algoritmo heurístico para o MTPP para grafos simples em geral em instâncias reais

Serão analisadas as instâncias reais que também são instâncias de grafos simples em geral. Para esse caso, a heurística executa inicialmente o algoritmo *Grafo Cacto* e, em seguida, o algoritmo *Tropical Cacto*. Serão avaliados dois fatores principais relacionados ao desempenho do algoritmo em relação aos valores ótimos: o tempo de execução e a precisão dos resultados obtidos.

No caso de instâncias reais, é igualmente importante avaliar a relação entre a solução ótima e a solução apresentada pelo algoritmo. Verifique a figura 13.

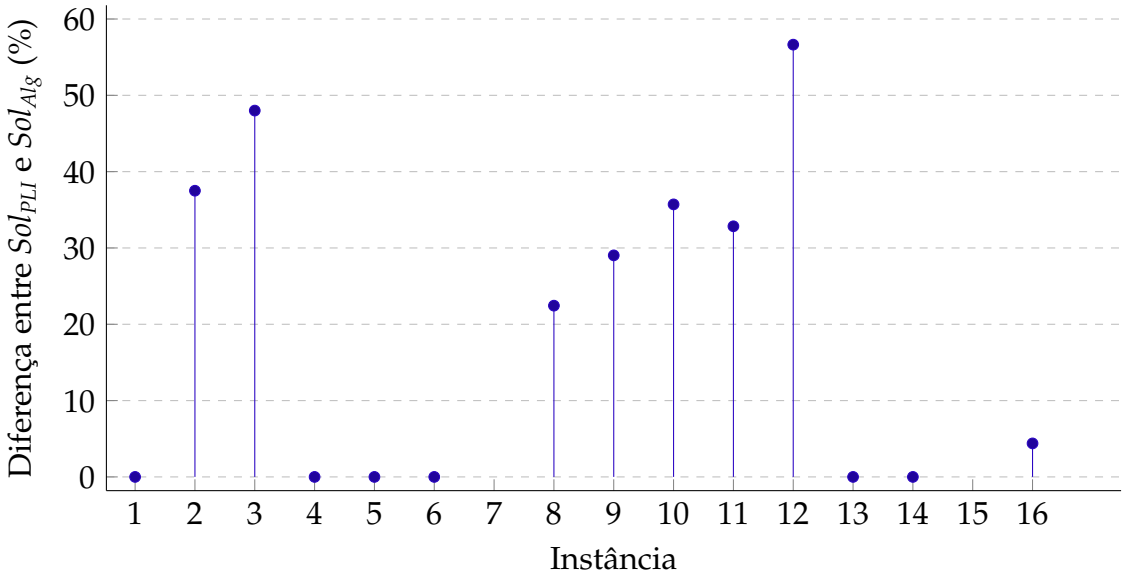


Figura 13 – Diferença entre a solução ótima e a solução do algoritmo para o MTPP para grafos simples em geral em instâncias reais (%)

O gráfico da Figura 13 revela que há uma porcentagem menor de instâncias que não atingiram o tempo limite e uma diferença maior entre a solução ótima e a solução apresentada pelo algoritmo no pior caso. Embora 57% das instâncias que não atingiram o tempo limite não tenham encontrado o valor ótimo, o algoritmo conseguiu encontrar valores inferiores adequados. No pior caso, a diferença entre a solução ótima e a solução apresentada pelo algoritmo foi inferior a 57% dos valores.

Os resultados dos experimentos computacionais mostram uma redução no tempo de execução superior a 90% em 57% das instâncias apresentadas. Isso pode tornar a heurística apropriada para aplicações que priorizam a velocidade de execução em detrimento da busca pela solução ótima.

Conjunto de resultados 3 - Instâncias aleatórias de DAGs

As Tabelas de 9 a 12 apresentam os resultados dos experimentos computacionais do modelo PLI simplificado para DAGs em instâncias aleatórias, e estão organizadas de acordo com o número de vértices das instâncias. Em cada tabela, as instâncias são identificadas pela coluna "Id"(identificador da instância), seguida das informações referentes à execução do modelo PLI, representadas pelas colunas $|RL|$, referente ao número de restrições lineares do modelo PLI, T_{PLI}, referente ao tempo de execução em segundos do modelo PLI, e W_{PLI}, referente ao trabalho gasto pelo modelo PLI.

Tabela 9 – Resultados dos experimentos computacionais para DAGs (Instâncias com 200 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}
A_{111}	421	0,15	0,02	8
A_{112}	421	0,13	0,03	10
A_{113}	421	0,41	0,27	20
A_{114}	421	0,32	0,28	20
A_{121}	501	0,12	0,03	6
A_{122}	501	0,15	0,03	10
A_{123}	501	0,28	0,19	85
A_{124}	501	0,29	0,23	93
A_{131}	581	0,14	0,03	6
A_{132}	581	0,16	0,04	10
A_{133}	581	0,24	0,17	114
A_{134}	581	0,26	0,21	146

Fonte – Igor Sampaio, 2023

A primeira tabela apresenta instâncias com 200 vértices, e em todas elas o valor ótimo foi encontrado em menos de 0,5 segundos. Esses são resultados excelentes, especialmente quando comparados com as instâncias de grafos simples com o mesmo tamanho.

Nas instâncias com 400 vértices, é possível observar, de forma mais consistente, o crescimento proporcional do tempo de execução e do trabalho gasto em relação ao aumento do número de arestas em instâncias com o mesmo número de cores.

Os pontos anteriormente destacados mantêm-se verdadeiros ao observar as instâncias com 600 vértices. No entanto, outro ponto que pode ser notado em todas as instâncias até o momento é que instâncias com menos cores tendem a ter um tempo de execução e trabalho gasto maior dentro de instâncias com a mesma quantidade de vértices e arestas.

Tabela 10 – Resultados dos experimentos computacionais para DAGs (Instâncias com 400 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}
A_{211}	841	0,24	0,08	7
A_{212}	841	0,32	0,11	11
A_{213}	841	2,13	2,55	40
A_{214}	841	2,65	3,17	40
A_{221}	1001	0,38	0,10	6
A_{222}	1001	0,37	0,11	13
A_{223}	1001	1,16	1,10	176
A_{224}	1001	1,27	1,31	187
A_{231}	1161	0,49	0,13	6
A_{232}	1161	0,46	0,13	12
A_{233}	1161	1,11	0,96	221
A_{234}	1161	1,23	1,11	282

Fonte – Igor Sampaio, 2023

Tabela 11 – Resultados dos experimentos computacionais para DAGs (Instâncias com 600 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}
A_{311}	1261	0,78	0,16	7
A_{312}	1261	0,73	0,20	12
A_{313}	1261	9,47	9,04	60
A_{314}	1261	11,45	10,54	60
A_{321}	1501	0,97	0,22	6
A_{322}	1501	0,98	0,24	11
A_{323}	1501	4,58	3,52	258
A_{324}	1501	4,39	3,91	287
A_{331}	1741	1,38	0,28	7
A_{332}	1741	1,18	0,29	11
A_{333}	1741	3,14	2,33	325
A_{334}	1741	4,09	3,04	423

Fonte – Igor Sampaio, 2023

As instâncias com 800 vértices concluem o conjunto de instâncias aleatórias para DAGs, e todos os pontos anteriormente destacados foram confirmados pelas instâncias com 800 vértices.

Análise da simplificação do modelo PLI para DAGs em instâncias aleatórias

Também é possível analisar a simplificação do modelo por meio do número de desigualdades adicionadas ao modelo, e é possível observar isso por meio dos experimentos computacionais. A Figura 14 mostra a relação entre o crescimento do

Tabela 12 – Resultados dos experimentos computacionais para DAGs (Instâncias com 800 vértices)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}
A_{411}	1681	1,15	0,29	8
A_{412}	2001	1,70	0,41	10
A_{413}	1681	16,46	15,02	80
A_{414}	1681	31,14	34,34	80
A_{421}	2001	1,60	0,38	7
A_{422}	2001	1,70	0,41	11
A_{423}	2001	9,44	8,00	342
A_{424}	2001	10,45	9,98	378
A_{431}	2321	2,35	0,48	7
A_{432}	2321	2,28	0,49	11
A_{433}	2321	6,53	4,96	440
A_{434}	2321	7,83	6,89	544

Fonte – Igor Sampaio, 2023

número de restrições lineares no modelo e o crescimento na quantidade de vértices do modelo simplificado para DAGS.

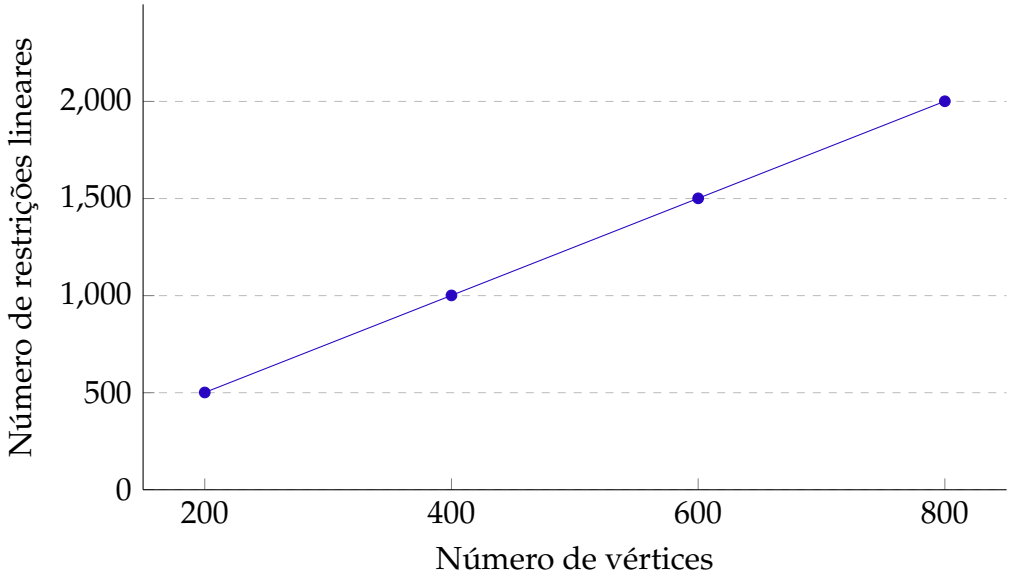


Figura 14 – Relação entre o crescimento do número de restrições lineares no modelo e o número de vértices da instância para DAGs.

Pela figura e pelos valores apresentados, é possível constatar que o número de restrições lineares em instâncias do mesmo tamanho cresce de forma muito inferior ao modelo original. É possível calcular que o número de restrições lineares de uma instância segue a fórmula de $2n + c + 1$, onde n é o número de vértices da instância e c é o número de cores. Isso define um crescimento linear do número de restrições lineares

em relação ao número de vértices da instância. Isso é uma prova empírica de que a característica que apresenta maior complexidade para tratar no problema de grafos simples em geral é evitar circuitos com a maior qualidade possível.

Nesta simplificação, conseguimos obter soluções ótimas inteiras para instâncias com mais de 3000 vértices em menos de 1 hora (3600 segundos) de execução. Entretanto, foi decidido apresentar instâncias com tamanhos equivalentes aos utilizados para o modelo PLI original. Dentre as 48 instâncias com no máximo 800 vértices, 100% dos experimentos computacionais resultaram em tempo de execução inferior a 32 segundos para obter soluções ótimas inteiras.

A grande diferença de tempo de execução, cerca de 99% mais rápido, entre o modelo original e o modelo simplificado para DAGs, serve como prova empírica de que evitar circuitos no problema original é custoso em relação ao tempo de execução, e também demonstra a possível eficiência da simplificação do modelo para um possível uso prático.

A relação entre o tempo de execução em segundos e o número de vértices é demonstrada na figura 15, que mostra como o tempo de execução cresce de forma linear com o número de vértices.

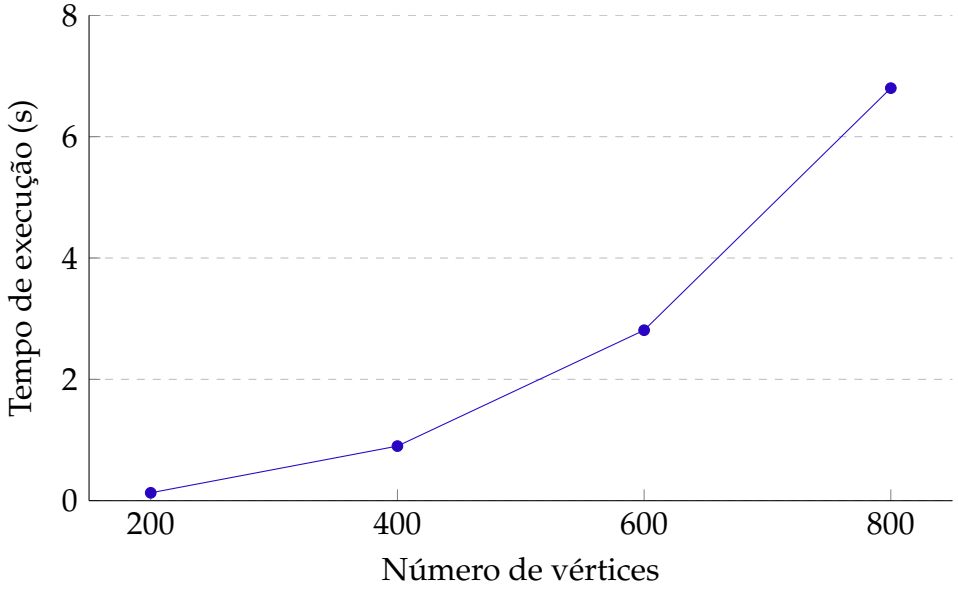


Figura 15 – Relação entre o crescimento do número de vértices da instância e seu tempo de execução para DAGs.

Assim como no modelo original, observando a figura 15, é possível concluir que o tempo de execução cresce diretamente com o aumento do número de vértices.

O crescimento no número de cores e arestas se mostraram bem menos significativos em relação ao impacto no tempo de execução do que o crescimento do número de vértices. No entanto, algumas características podem ser observadas na análise dos resultados dos experimentos.

- O aumento no número de cores não tem uma relação diretamente proporcional com o tempo de execução, apresentando maior tempo de execução e tempo gasto em instancias com menos cores.
- Além disso, o número de arestas tem uma relação proporcional com o tempo de execução e trabalho gasto, ou seja, o tempo de execução e trabalho gasto normalmente aumenta conforme o número de arestas aumenta.

Conjunto de resultados 4 - Instâncias reais de DAGs

A Tabela 13 apresenta os resultados dos experimentos computacionais do modelo PLI simplificado para DAGs em instâncias reais. Na tabela, as instâncias são identificadas pela coluna "Id"(identificador da instância), seguida dos dados da execução do modelo PLI representados pelas colunas "$|RL|$"(número de restrições lineares do modelo PLI), "T_{PLI}"(tempo de execução em segundos do modelo PLI) e "W_{PLI}"(trabalho gasto pelo modelo PLI).

Análise da simplificação do modelo PLI para DAGs em instâncias reais

As instâncias reais confirmam os pontos destacados pelas instâncias aleatórias. As instâncias reais seguem a fórmula de $2n + c + 1$, onde n é o número de vértices da instância e c é o número de cores, assim como as instâncias aleatórias. O crescimento do número de restrições lineares no modelo se mantém linear em relação ao crescimento do número de vértices.

Entre as instâncias reais, há uma variação muito maior no número de vértices em comparação às instâncias aleatórias, incluindo instâncias com até 2719 vértices. Diferentemente da versão para grafos simples, todas as instâncias tiveram seu valor ótimo encontrado em menos de 30 segundos.

Tabela 13 – Resultados dos experimentos computacionais para DAGs (Instâncias Reais)

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}
R_1	163	0,02	0,01	8
R_2	194	0,02	0,01	21
R_3	225	0,03	0,01	25
R_4	484	0,30	0,26	11
R_5	488	0,29	0,21	11
R_6	625	1,35	0,30	25
R_7	744	0,46	0,11	37
R_8	863	0,47	0,10	42
R_9	1711	1,26	0,38	20
R_{10}	1739	1,43	0,35	25
R_{11}	1747	1,24	0,35	30
R_{12}	1989	1,73	0,43	32
R_{13}	2584	6,21	2,57	10
R_{14}	2593	3,68	1,14	14
R_{15}	2864	3,23	0,91	9
R_{16}	4797	8,32	2,36	29
R_{17}	5711	12,20	3,10	32
R_{18}	6625	17,70	3,77	32
R_{19}	5711	22,14	3,03	11
R_{20}	6799	22,27	4,30	11
R_{21}	7887	29,23	5,30	11

Fonte – Igor Sampaio, 2023

Mesmo com tempos de execução muito curtos, é possível notar uma consistência na proporção de tempo de execução e trabalho gasto em relação ao tamanho da instância, confirmando uma proporção na relação entre o crescimento do número de vértices, cores e arestas e o crescimento do tempo de execução consistentes entre os experimentos com instâncias aleatórias e instâncias reais.

Conjunto de resultados 5 - Instâncias aleatórias de grafos cacto

A Tabela 14 apresenta os resultados dos experimentos computacionais com o modelo PLI e o algoritmo para DAGs, aplicados em instâncias aleatórias. Na tabela, as instâncias são identificadas pela coluna "Id"(identificador da instância), seguida dos dados da execução do modelo PLI representados pelas colunas "$|RL|$"(número de restrições lineares do modelo PLI), "T_{PLI}"(tempo de execução em segundos do modelo PLI), "W_{PLI}"(trabalho gasto pelo modelo PLI) e "Sol_{PLI}"(valor da solução ótima encontrada pelo modelo PLI). Por fim as colunas "T_{Alg}"(tempo de execução em

segundos do algoritmo heurísticos), "Sol_{Alg}"(valor da solução encontrada pelo algoritmo heurístico), "GAP_T"(diferença entre o tempo de execução do modelo PLI e do algoritmo em porcentagem) e "GAP_Sol"(diferença entre a solução ótima e a solução encontrada pelo algoritmo heurístico em porcentagem).

Tabela 14 – Resultados dos experimentos computacionais - Algoritmo para grafos cacto

Id	$\|RL_{PLI}\|$	T_{PLI}	$\|W_{PLI}\|$	Sol_{PLI}	T_{Alg}	Sol_{Alg}	GAP_T	GAP_{Sol}
A^c111	839	29,52	17,79	19	0,154	19	99,48%	0,00%
A^c112	859	6,30	4,42	20	0,005	20	99,93%	0,00%
A^c121	919	8,23	5,56	33	0,090	33	98,90%	0,00%
A^c122	939	23,42	12,07	38	0,180	37	99,23%	2,63%
A^c131	999	11,67	6,05	35	0,122	35	98,95%	0,00%
A^c132	1019	18,37	9,45	35	0,121	35	99,34%	0,00%
A^c211	1679	3600,49	2399,89	29	0,364	29	-	-
A^c212	1719	3600,36	1730,61	32	0,447	33	-	-
A^c221	1839	123,91	74,00	41	0,360	41	99,71%	0,00%
A^c222	1879	666,00	426,71	48	0,374	48	99,94%	0,00%
A^c231	1999	19,60	10,08	48	0,383	48	98,05%	0,00%
A^c232	2039	32,26	15,97	51	0,378	46	98,83%	9,80%
A^c311	2519	3600,45	1895,26	40	0,954	39	-	-
A^c312	2579	3601,02	1996,05	39	0,850	42	-	-
A^c321	2759	1791,34	954,95	56	0,848	56	99,95%	0,00%
A^c322	2819	3600,86	1901,53	51	0,872	55	-	-
A^c331	2999	1519,61	851,91	49	0,824	49	99,95%	0,00%
A^c332	3059	3600,77	2050,42	58	0,832	58	-	-
A^c411	3359	3603,95	1733,79	45	1,575	46	-	-
A^c412	3439	3601,14	1738,87	34	1,809	45	-	-
A^c421	3679	3601,69	1230,25	57	1,683	57	-	-
A^c422	3759	3601,67	1237,91	65	2,211	66	-	-
A^c431	3999	3601,99	1253,40	69	1,758	69	-	-
A^c432	4079	3601,47	1256,90	81	1,970	77	-	-

Fonte – Igor Sampaio, 2023

Um ponto importante a ser destacado é que, apesar das instâncias de grafo cacto terem tamanho máximo de 800 vértices, assim como nos experimentos para grafos simples, muitas instâncias não obtiveram a solução ótima dentro do tempo de execução determinado. Isso pode indicar que a estrutura de grafos cacto traz maior dificuldade para o modelo PLI.

Análise do algoritmo heurístico para o MTPP para grafos cacto em instâncias aleatórias

Para grafos cacto, a heurística para o problema MTPP roda diretamente o algoritmo *Caminho Tropical*, sem a necessidade de adaptação do grafo. Serão avaliados dois principais fatores relacionados ao desempenho do algoritmo em relação aos valores ótimos: o tempo de execução e a qualidade dos resultados obtidos.

Novamente, para grafos cacto, é avaliado o desempenho do algoritmo por meio da relação entre a solução ótima e a solução apresentada pelo algoritmo. Observe a figura 16.

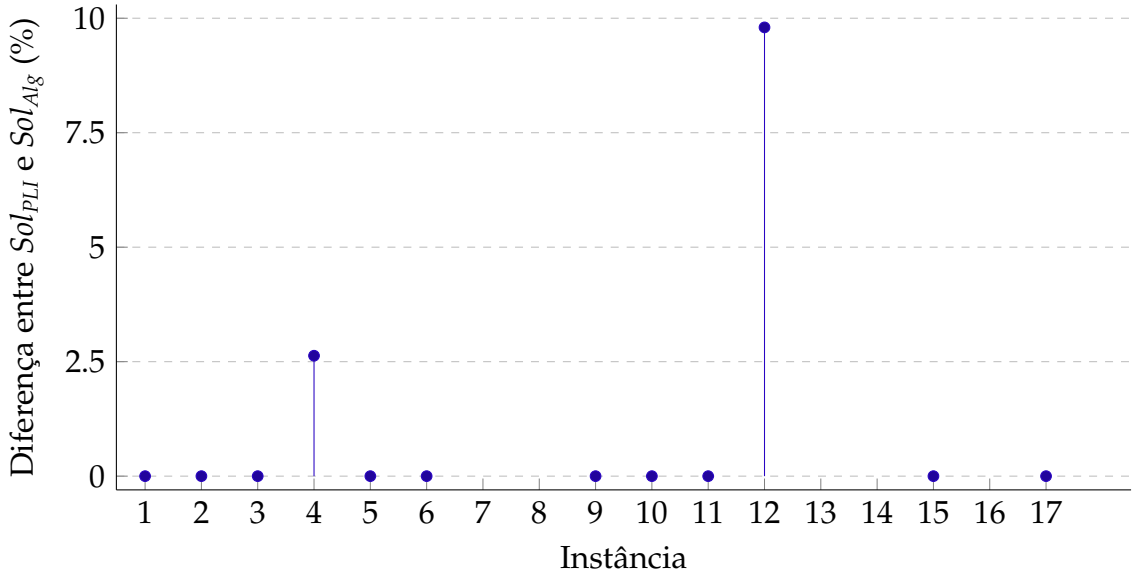


Figura 16 – Diferença entre a solução ótima e a solução do algoritmo para o MTPP para grafos cacto em instâncias aleatórias (%)

Com instâncias de grafos cacto, que não precisam de adaptação, é notável, ao observar o gráfico da Figura 16, que a heurística fornece limitantes inferiores muito melhores e encontra a solução ótima em mais casos do que para grafos simples em geral. Observando apenas as instâncias apresentadas que não atingiram o tempo limite na execução do modelo PLI, é possível notar que menos de 17% das instâncias não encontraram o valor ótimo, um valor muito inferior ao observado para grafos simples em geral. Melhor ainda é o fato de que, no pior caso, a diferença entre a solução ótima e a solução apresentada pelo algoritmo foi inferior a 10% entre os valores. Conclui-se que a heurística fornece limitantes inferiores ótimos para grafos cacto e, em muitos casos, pode ser uma boa alternativa para aplicações práticas.

Um fator que se mantém constante na análise da heurística é a qualidade do tempo de execução. Observando apenas as instâncias em que é possível obter valores ótimos, o tempo de execução do algoritmo é, em todos os casos, pelo menos 98% melhor do que o tempo de execução do modelo PLI. O algoritmo se mostrou muito eficiente em relação ao tempo de execução, ainda mais pelo fato de que instâncias de grafos cacto terem, normalmente, uma construção que traz dificuldades em termos de tempo de execução do modelo PLI.

7 Considerações finais

O estudo de um problema NP-Difícil exige uma compreensão profunda das técnicas e algoritmos que existem para tentar explorar esse tipo de problema, bem como uma habilidade para formular modelos matemáticos e criar algoritmos heurísticos eficientes. No entanto, esses modelos e algoritmos podem ser extremamente complexos e difíceis de analisar teoricamente, o que torna a tarefa de estudar esses problemas um desafio significativo.

Além disso, em se tratando de problemas NP-Difícil, vale observar que a realização de experimentos computacionais em instâncias reais é de extrema importância para avaliar o desempenho de um modelo de programação linear inteira e concomitantemente de algoritmos heurísticos. Tal desempenho pode variar se for comparado com instâncias geradas aleatoriamente, embora construir um banco de instâncias reais não seja uma tarefa trivial quando estas precisam ser tratadas.

Em suma, esta dissertação estudou o problema do caminho tropical em grafos coloridos nos vértices e apontou algumas direções para explorá-lo. Foi desenvolvido um modelo de programação linear inteira, bem como uma simplificação desse modelo para grafos acíclicos direcionados (DAGs). Além disso, foi criado um algoritmo heurístico em que dado um grafo e uma coloração, este devolve um caminho colorido nos vértices.

A partir dos resultados experimentais foi possível notar que a formulação do modelo PLI para o MTPP em grafos encontrou solução ótima para mais de 95% das instâncias com até 800 vértices e para algumas instâncias com mais 2 mil vértices dentro do tempo limite pré definido de 60 minutos.

Para o caso especial de DAGs, o modelo PLI obteve solução ótima para todas as instâncias aleatórias geradas com até 800 vértices em no máximo 32 segundos. Além disso, o modelo PLI encontrou soluções ótimas para todas as instâncias reais em menos de 30 segundos. Vale observar que instâncias maiores poderiam ser geradas aleatoriamente para DAGs, no entanto, os grafos do tipo DAGs foram construídos a partir das instâncias geradas aleatoriamente para grafos.

Em relação aos resultados dos experimentos com o algoritmo heurístico para o conjunto de instâncias de grafos simples em geral, foi observado que 28% das instâncias aleatórias e 43% das instâncias reais alcançaram o valor ótimo. No pior caso, o algoritmo

obteve uma diferença máxima de 57% entre o valor ótimo e o valor obtido pelo algoritmo. Já em grafos cacto, o valor ótimo foi alcançado em mais de 83% das instâncias. Para as instâncias que não atingiram o valor ótimo, a diferença entre o valor ótimo e o obtido pelo algoritmo foi inferior a 10%.

Os resultados obtidos pelos experimentos computacionais mostraram o potencial do algoritmo em encontrar soluções ótimas para algumas instâncias do problema, em especial instâncias de grafos cacto. É importante considerar o potencial do uso da heurística para a definição de limitantes inferiores do problema, o que pode ser útil para futuras abordagens.

Em geral, este estudo contribuiu para a compreensão do problema do caminho tropical em grafos coloridos nos vértices e apresentou soluções úteis para resolvê-lo. A análise empírica realizada permitiu avaliar o desempenho dessas soluções em diferentes cenários, o que pode orientar a escolha da melhor abordagem para resolver o problema em situações específicas.

Trabalhados futuros

Uma possibilidade interessante para futuros estudos é explorar o problema do caminho tropical em grafos coloridos nos vértices por meio de métodos de combinatória poliédrica. Essa abordagem pode permitir uma compreensão mais profunda da estrutura matemática subjacente ao problema e levar a soluções mais eficientes em termos de tamanho da instância e tempo de execução.

Um exemplo de estudo futuro nesse sentido seria investigar as propriedades do poliedro associado ao problema do caminho tropical em grafos simples. É possível que exista algumas inequações que definam facetas e tais inequações possam ser usadas em um modelo de PLI relaxado para que soluções inteiras ótimas sejam encontradas em um tempo consideravelmente menor.

Outra possibilidade seria explorar a conexão entre o problema do caminho tropical e outros problemas clássicos de combinatória poliédrica, como o problema da árvore geradora mínima (MST) e o problema do fluxo máximo em redes. Essa conexão pode permitir o uso de técnicas e algoritmos já existentes para resolver o problema do caminho tropical.

Em resumo, há muitas possibilidades de estudos futuros para explorar o problema do caminho tropical em grafos coloridos nos vértices usando métodos de combinatória poliédrica. Tais estudos podem levar a soluções mais eficientes e a uma compreensão mais profunda da estrutura matemática subjacente ao problema.

Referências

AKBARI, S.; LIAGHAT, V.; NIKZAD, A. Colorful paths in vertex coloring of graphs. *the electronic journal of combinatorics*, p. P17–P17, 2011. Citado na página 30.

ALON, N.; YUSTER, R.; ZWICK, U. Color-coding. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 42, n. 4, p. 844–856, 1995. Citado 2 vezes nas páginas 30 e 43.

ARORA, S.; BARAK, B. *Computational complexity: a modern approach*. [S.l.]: Cambridge University Press, 2009. Citado na página 26.

BABU, J.; BASAVARAJU, M.; CHANDRAN, L. S.; FRANCIS, M. C. On induced colourful paths in triangle-free graphs. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 61, p. 69–75, 2017. Citado na página 31.

BESSY, S.; BOUSQUET, N. Colorful paths for 3-chromatic graphs. *Discrete Mathematics*, Elsevier, v. 340, n. 5, p. 1000–1007, 2017. Citado na página 31.

BONDY, J.; MURTY, U. *Graph Theory*. [S.l.]: Springer, 2008. Citado na página 18.

BRUCKNER, S.; HÜFFNER, F.; KOMUSIEWICZ, C.; NIEDERMEIER, R. Evaluation of ilp-based approaches for partitioning into colorful components. In: SPRINGER. *International symposium on experimental algorithms*. [S.l.], 2013. p. 176–187. Citado na página 18.

BU, D.; ZHAO, Y.; CAI, L.; XUE, H.; ZHU, X.; LU, H.; ZHANG, J.; SUN, S.; LING, L.; ZHANG, N.; LI, G.-J.; CHEN, R. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic acids research*, v. 31, p. 2443–50, 06 2003. Citado na página 60.

CHAPELLE, M.; COCHEFERT, M.; KRATSCH, D.; LETOURNEUR, R.; LIEDLOFF, M. Exact exponential algorithms to find a tropical connected set of minimum size. In: SPRINGER. *International Symposium on Parameterized and Exact Computation*. [S.l.], 2014. p. 147–158. Citado 2 vezes nas páginas 16 e 31.

CHOR, B.; FELLOWS, M.; RAGAN, M. A.; RAZGON, I.; ROSAMOND, F.; SNIR, S. Connected coloring completion for general graphs: algorithms and complexity. In: SPRINGER. *International Computing and Combinatorics Conference*. [S.l.], 2007. p. 75–85. Citado na página 18.

COHEN, J.; ITALIANO, G. F.; MANOUSSAKIS, Y.; THANG, N. K.; PHAM, H. P. Tropical paths in vertex-colored graphs. *Journal of Combinatorial Optimization*, Springer, p. 1–23, 2019. Citado 6 vezes nas páginas 16, 27, 31, 32, 33 e 34.

COHEN, J.; MANOUSSAKIS, Y.; PHONG, H.; TUZA, Z. Tropical matchings in vertex-colored graphs. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 62, p. 219–224, 2017. Citado 2 vezes nas páginas 16 e 31.

CONFORTI, M.; CORNUéJOLS, G.; ZAMBELLI, G. *Integer programming*. [S.l.]: Springer, 2014. v. 271. Citado na página 26.

COREL, E.; PITSCHI, F.; MORGENSTERN, B. A min-cut algorithm for the consistency problem in multiple sequence alignment. *Bioinformatics*, Oxford University Press, v. 26, n. 8, p. 1015–1021, 2010. Citado na página 18.

CORMEN, T. H.; LEISERSON, C.; RIVEST, R.; STEIN, C. *Introduction to algorithms,|| Cambridge, Mass*. [S.l.]: MIT Press, 2009. Citado na página 23.

COSTA, L. d. F. Random and longest paths: Unnoticed motifs of complex networks. *arXiv preprint arXiv:0712.0415*, 2007. Citado na página 28.

D'AURIAC, J.-A. A.; BUJTÁS, C.; MAFTOUHI, A. E.; KARPINSKI, M.; MANOUSSAKIS, Y.; MONTERO, L.; NARAYANAN, N.; ROSAZ, L.; THAPPER, J.; TUZA, Z. Tropical dominating sets in vertex-coloured graphs. *Journal of Discrete Algorithms*, Elsevier, v. 48, p. 27–41, 2018. Citado 2 vezes nas páginas 16 e 31.

D'AURIAC, J.-A. A.; COHEN, N.; MAFTHOUI, H. E.; HARUTYUNYAN, A.; LEGAY, S.; MANOUSSAKIS, Y. Connected tropical subgraphs in vertex-colored graphs. 2016. Citado 2 vezes nas páginas 16 e 31.

DONDI, R.; HOSSEINZADEH, M. M. Finding colorful paths in temporal graphs. In: SPRINGER. *International Conference on Complex Networks and Their Applications*. [S.l.], 2021. p. 553–565. Citado na página 31.

EULER, L. *Mechanica Sive Motus Scientia Analytice Exposita: Instar Supplementi Ad Commentar. Acad. Scient. Imper*. [S.l.]: Ex typographia academiae scientiarum, 1736. v. 2. Citado 2 vezes nas páginas 18 e 29.

EUROSIS. *EuroSiS web mapping study*. 2009. Last accessed 27 Sep 2021. Disponível em: ⟨https://github.com/gephi/gephi/wiki/Datasets⟩. Citado na página 59.

FELLOWS, M. R.; FERTIN, G.; HERMELIN, D.; VIALETTE, S. Upper and lower bounds for finding connected motifs in vertex-colored graphs. *Journal of Computer and System Sciences*, Elsevier, v. 77, n. 4, p. 799–811, 2011. Citado na página 28.

FELLOWS, M. R.; HALLETT, M. T.; WAREHAM, H. T. Dna physical mapping: Three ways difficult. In: SPRINGER. *European Symposium on Algorithms*. [S.l.], 1993. p. 157–168. Citado na página 18.

FERREIRA, C. E.; WAKABAYASHI, Y. *Combinatória poliédrica e planos-de-corte faciais*. [S.l.]: UNICAMP-Instituto de Computacao, 1996. Citado 3 vezes nas páginas 26, 40 e 41.

FREIRE, A. da S.; LIMA, K. R.; ROJAS, D. I. Z. A system for motif search in biological networks. In: *Proceedings of the XIV Brazilian Symposium on Information Systems*. [S.l.: s.n.], 2018. p. 1–8. Citado 2 vezes nas páginas 18 e 28.

FUNG, T.-S. A colourful path. *The Mathematical Gazette*, Cambridge University Press, v. 73, n. 465, p. 186–188, 1989. Citado na página 30.

GABOW, H. N.; NIE, S. Finding a long directed cycle. *ACM Transactions on Algorithms (TALG)*, ACM New York, NY, USA, v. 4, n. 1, p. 1–21, 2008. Citado 3 vezes nas páginas 42, 43 e 44.

GEPHI. *Conjuntos de dados Gephi*. 2009. Last accessed 27 Sep 2021. Disponível em: ⟨https://github.com/gephi/gephi/wiki/Datasets⟩. Citado na página 58.

GOH, K.-I.; CUSICK, M. E.; VALLE, D.; CHILDS, B.; VIDAL, M.; BARABÁSI, A.-L. The human disease network. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 104, n. 21, p. 8685–8690, 2007. ISSN 0027-8424. Citado na página 60.

HASHIM, F. A.; MABROUK, M. S.; AL-ATABANY, W. Review of different sequence motif finding algorithms. *Avicenna journal of medical biotechnology*, Avicenna Research Institute, v. 11, n. 2, p. 130, 2019. Citado na página 27.

IOANNIDOU, K.; MERTZIOS, G. B.; NIKOLOPOULOS, S. D. The longest path problem is polynomial on interval graphs. In: SPRINGER. *International Symposium on Mathematical Foundations of Computer Science*. [S.l.], 2009. p. 403–414. Citado 2 vezes nas páginas 30 e 32.

JOSHI, A.; PARYS, T. V.; PEER, Y. Van de; MICHOEL, T. Characterizing regulatory path motifs in integrated networks using perturbational data. *Genome biology*, Springer, v. 11, n. 3, p. 1–14, 2010. Citado na página 28.

KARGER, D.; MOTWANI, R.; RAMKUMAR, G. D. On approximating the longest path in a graph. *Algorithmica*, Springer, v. 18, n. 1, p. 82–98, 1997. Citado 2 vezes nas páginas 29 e 33.

KNUTH, D. E. *The Stanford GraphBase: a platform for combinatorial computing*. [S.l.]: ACM, 1993. Citado na página 58.

KOWALIK, Ł.; LAURI, J. On finding rainbow and colorful paths. *Theoretical Computer Science*, Elsevier, v. 628, p. 110–114, 2016. Citado na página 30.

LACROIX, V.; FERNANDES, C. G.; SAGOT, M.-F. Motif search in graphs: application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, IEEE, v. 3, n. 4, p. 360–368, 2006. Citado na página 18.

LI, H. A generalization of the gallai–roy theorem. *Graphs and Combinatorics*, Springer, v. 17, n. 4, p. 681–685, 2001. Citado na página 30.

LIN, C. Simple proofs of results on paths representing all colors in proper vertex-colorings. *Graphs and Combinatorics*, Springer, v. 23, n. 2, p. 201–203, 2007. Citado na página 30.

LINKFLUENCE. *CPAN authors*. 2009. Last accessed 27 Sep 2021. Disponível em: ⟨https://github.com/gephi/gephi/wiki/Datasets⟩. Citado na página 59.

MAKOLO, A. A comparative analysis of motif discovery algorithms. *Computational Biology and Bioinformatics*, Science Publishing Group, v. 4, n. 1, p. 1, 2015. Citado na página 27.

MARKOV, M.; ANDREICA, M. I.; MANEV, K.; TAPUS, N. A linear time algorithm for computing longest paths in cactus graphs. *Serdica Journal of Computing*, v. 6, n. 3, p. 287p–298p, 2012. Citado 2 vezes nas páginas 30 e 42.

MARX, D. Graph colouring problems and their applications in scheduling. *Periodica Polytechnica Electrical Engineering*, v. 48, n. 1-2, p. 11–16, 2004. Citado na página 18.

MCMORRIS, F. R.; WARNOW, T. J.; WIMER, T. Triangulating vertex-colored graphs. *SIAM Journal on Discrete Mathematics*, SIAM, v. 7, n. 2, p. 296–306, 1994. Citado na página 18.

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, ACM New York, NY, USA, v. 7, n. 4, p. 326–329, 1960. Citado 2 vezes nas páginas 35 e 38.

NEWMAN, M. E. The structure and function of complex networks. *SIAM review*, SIAM, v. 45, n. 2, p. 167–256, 2003. Citado na página 15.

REZENDE, S. de; WAKABAYASHI, Y. Caminhos mais longos em grafos. In: SBC. *Anais do XXVIII Concurso de Teses e Dissertações*. [S.l.], 2015. p. 49–54. Citado na página 29.

REZENDE, S. F. de. *Caminhos mais longos em grafos*. Dissertação (Mestrado) — Universidade de São Paulo (USP). Instituto de Matemática e Estatística (IME/SBI), 2014. Citado na página 42.

SANDVE, G. K.; DRABLØS, F. A survey of motif discovery methods in an integrated framework. *Biology direct*, BioMed Central, v. 1, n. 1, p. 1–16, 2006. Citado na página 27.

SCOTT, J.; IDEKER, T.; KARP, R. M.; SHARAN, R. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, Mary Ann Liebert, Inc. 2 Madison Avenue Larchmont, NY 10538 USA, v. 13, n. 2, p. 133–144, 2006. Citado 2 vezes nas páginas 18 e 28.

SHEN-ORR, S. S.; MILO, R.; MANGAN, S.; ALON, U. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, Nature Publishing Group, v. 31, n. 1, p. 64–68, 2002. Citado na página 18.

STEHLé, J.; VOIRIN, N.; BARRAT, A.; CATTUTO, C.; ISELLA, L.; PINTON, J.-F.; QUAGGIOTTO, M.; BROECK, W. Van den; RéGIS, C.; LINA, B.; VANHEMS, P. High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE*, Public Library of Science, v. 6, n. 8, p. 1–13, 08 2011. Citado na página 58.

TALBI, E.-G. *Metaheuristics: from design to implementation*. [S.l.]: John Wiley & Sons, 2009. Citado na página 42.

THEJASWI, S.; GIONIS, A.; LAURI, J. Finding path motifs in large temporal graphs using algebraic fingerprints. *Big Data*, Mary Ann Liebert, Inc., publishers 140 Huguenot Street, 3rd Floor New . . . , v. 8, n. 5, p. 335–362, 2020. Citado na página 28.

THIE, P. R.; KEOUGH, G. E. *An introduction to linear programming and game theory*. [S.l.]: John Wiley & Sons, 2011. Citado na página 23.

UEHARA, R.; UNO, Y. Efficient algorithms for the longest path problem. In: SPRINGER. *International symposium on algorithms and computation*. [S.l.], 2004. p. 871–883. Citado na página 30.

UEHARA, R.; VALIENTE, G. Linear structure of bipartite permutation graphs and the longest path problem. *Information Processing Letters*, Elsevier, v. 103, n. 2, p. 71–77, 2007. Citado 2 vezes nas páginas 30 e 32.

VALVERDE, S.; SOLÉ, R. V. Motifs in graphs. In: ____. *Encyclopedia of Complexity and Systems Science*. New York, NY: Springer New York, 2009. p. 5692–5702. ISBN 978-0-387-30440-3. Disponível em: ⟨https://doi.org/10.1007/978-0-387-30440-3_339⟩. Citado na página 15.

WEST, D. B. *et al. Introduction to Graph Theory*. [S.l.]: Prentice hall Upper Saddle River, 2001. v. 2. Citado na página 23.

WHITE, J.; SOUTHGATE, E.; THOMSON, J.; BRENNER, S. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, v. 314, n. 1165, p. 1—340, November 1986. ISSN 0962-8436. Citado na página 59.

WOLSEY, L. A. *Integer programming*. [S.l.]: John Wiley & Sons, 2020. Citado na página 24.