UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

XIAO YONG KONG
YGOR LUIS M. P. DA HORA

ALGORITMO APROXIMATIVO PARA O PROBLEMA DE MAXIMIZAÇÃO DE INFLUÊNCIA PARA CONVEXIDADE P3

RIO DE JANEIRO
2023

XIAO YONG KONG
YGOR LUIS M. P. DA HORA

ALGORITMO APROXIMATIVO PARA O PROBLEMA DE MAXIMIZAÇÃO DE INFLUÊNCIA PARA CONVEXIDADE P3

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Daniel Sadoc Menasche
Co-orientador: Prof. Claudio Miceli e Prof. Vitor Ponciano

RIO DE JANEIRO
2023

Kong, Xiao Yong
K82a Algoritmo aproximativo para o problema de maximização de influência para convexidade P3 / Xiao Yong Kong e Ygor Luis Mesquita Pereira da Hora. – 2023.

56 f.

Orientador: Daniel Sadoc Menasche.
Coorientador: Claudio Miceli de Farias.
Coorientador: Vitor dos Santos Ponciano.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2023.

1. Grafos. 2. Convexidade. 3. P3. 4. Algoritmo. 5. Busca-binária. I. Hora, Ygor Luis Mesquita Pereira da. II. Menasche, Daniel Sadoc (Orient). III. Universidade Federal do Rio de Janeiro, Instituto de Computação. IV. Título.

XIAO YONG KONG
YGOR LUIS M. P. DA HORA

ALGORITMO APROXIMATIVO PARA O PROBLEMA DE MAXIMIZAÇÃO DE INFLUÊNCIA PARA CONVEXIDADE P3

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 31 de MARÇO de 2023

BANCA EXAMINADORA:

Daniel Sadoc Menasché
PhD (UFRJ)

Claudio Miceli de Farias
Dr (UFRJ)

Josefino Cabral Melo Lima
Docteur (UFRJ)

Vitor Ponciano
Mestre (UFRJ)

Dedicatória: ao meu pai, que comprava livros de matemática sem ter noção do quanto isso aguçaria minha curiosidade e a minha mãe que pacientemente me ensinava a interpretar e resolver seus problemas.

AGRADECIMENTOS

Primeiro gostaria de agradecer às nossas famílias pelo apoio desde o início, nossos pais por nos criarem e sempre nos apoiar, assim como nossos cônjugues.

Também tenho a agradecer ao nosso orientador e co-orientadores por nos acolher e dar todo o apoio e o material necessário para continuar com o projeto, obrigado também a todos os professores da UFRJ, que com paciência, construíram nosso caminho acadêmico e todo o conhecimento necessário para continuarmos até o final.

Um agradecimento especial ao nosso co-orientador Vitor, que nos apresentou o problema, explicou, mostrou outros projetos e seguiu nos apoiando até então, revisando e se reunindo. Sem você não teriamos chegado até aqui, amigo.

E por fim e não menos importante, obrigado ao projetos como GRIS, competição de algoritmos e as empresas que atuamos que nos ajudaram a criar um pensamento mais crítico e ajudou a nos tornamos melhores profissionais. Muito obrigado a todos.

"Few are those who see with their
own eyes and feel with their own hearts."

Albert Einstein

RESUMO

Esse trabalho aborda um estudo sobre convexidade P3 em grafos com uma aplicação na área de Sistema de Recomendação e influências. O objetivo é modelar e implementar um algoritmo que possa ser utilizado de modo real e mostrar como um conjunto mínimo de usuários pode influenciar os demais usuários associados com o máximo de aceitação.

Palavras-chave: grafos; TCC; convexidade; P3; algoritmo; busca-binaria; better-influence; cordal; grade; infecção-redes-sociais;

ABSTRACT

This work addresses a study on P3 convexity in graphs with an application in Recommender system and influences area. The objective is to model and implement a algorithm that can be used in a real way and show how a minimum set of users can influence the other associated users with maximum acceptance.

Keywords: graphs; TCC; convexidade; P3; algorithm; binary-search; better-influence; cordal; grade; social-media-infection;

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

"Há duas maneiras de viver a vida: a primeira é vivê-la como se os milagres não existissem. A segunda é vivê-la como se tudo fosse milagre".

Albert Einstein

Na atualidade os *sistemas de recomendação* estão entre as aplicações mais populares da ciência de dados. Eles são usados, na maioria das vezes, para prever a "classificação" ou "preferência" que um usuário daria a um item. Sistemas de Recomendação auxiliam, através algoritmos computacionais, em sugestões como livros, filmes, roupas e artigos. São ferramentas poderosas que têm o poder extrair dados significativos para empresas a partir de suas bases de dados. Quase todas as grandes empresas de tecnologia as aplicaram de alguma forma, por exemplo: a Amazon a utiliza para sugerir produtos aos clientes, o YouTube para decidir qual vídeo reproduzir na autoplay ou para recomendação de vídeo e o Facebook para recomendar páginas e pessoas a seguir.

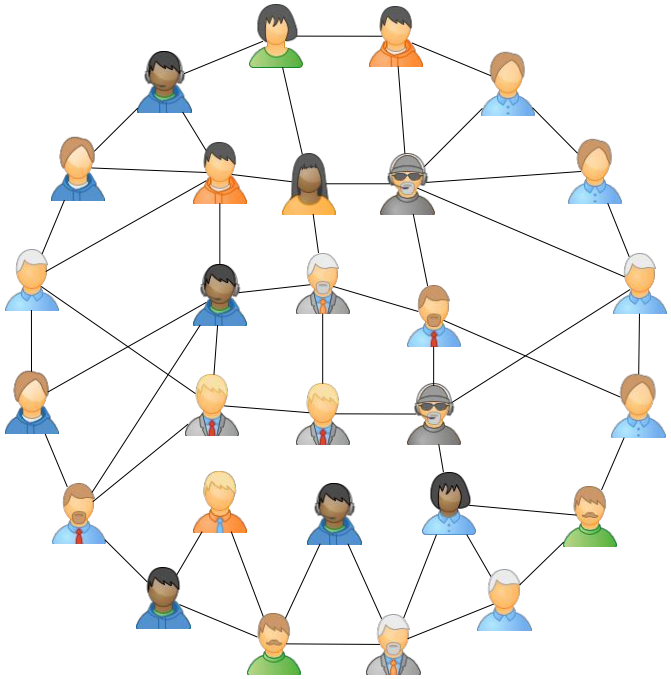

Figura 1 – Exemplos de Grafos

Um grafo é uma estrutura que serve para modelar o relacionamento de elementos de uma rede social. Grafos podem ser vistos como conjunto de pontos chamados vértices e os pares não ordenados destes pontos chamados arestas, cada aresta ligando um par de pontos (extremidades). O grafo pode servir de base para análise de diversas situações-problemas. Com grafos podemos modelar a disseminação de uma infecção, de uma informação ou de propaganda em uma rede social.

Campanhas publicitárias, de marketing e políticas são planejadas engajando vértices-influentes (são os vértice de maior potência de propagação na rede) de forma direta ou indireta. Tais situações muitas vezes podem ser pensadas como problemas em grafos, inclusive usando temas mais específicos como a convexidade em grafos, onde se começa com um conjunto inicial de vértices aos quais novos vértices são adicionados a esse conjunto sempre que vizinhos suficientes já estiverem dentro do conjunto (RAMOS; SANTOS; SZWARCFITER, 2014), imaginando uma espécie de contaminação na rede. Em particular, se um vértice é infectado quando pelo menos dois de seus vizinhos já estão infectados, temos a conhecida convexidade P_3. No caso atual de pandemia ou de disseminação fake news, podemos por como objetivo delimitar o espaço de alcance de um subconjunto particular inicial. Dessa forma a convexidade pode ser uma forma de conter a contaminação de uma rede real.

1.1 CONVEXIDADE EM GRAFOS

Nos últimos anos, surgiram muitas pesquisas sobre convexidade em grafos por trabalhos como (CENTENO, 2012b), (DOURADO; PROTTI; SZWARCFITER, 2010), (PALUGA; CANOY, 2007). Uma das primeiras discussões sobre convexidade em grafos se deu nos anos 70 com artigos de (ERDöS et al., 1972), (LEVI, 1951) e (MOON, 1972), essencialmente relacionados a torneios. A partir disso muitos artigos foram publicados explorando este tema. A complexidade computacional de vários parâmetros dessas convexidades, quando restrita a certos grafos, já foi e continua sendo muito estudada, inclusive muitos dos importantes problemas relacionados a esses parâmetros já se mostraram NP-difíceis, como pode ser visto no trabalho (DOURADO et al., 2012). Além da convexidade dos caminhos mínimos, a qual é a convexidade naturalmente considerada em grafos, e da convexidade dos espaços euclidianos, outras convexidades já se encontram consideradas na literatura da teoria dos grafos, como a já citada convexidade P_3 estudada em vários trabalhos, como no recente texto (CENTENO, 2012b), além da convexidade monofônica (DOURADO; PROTTI; SZWARCFITER, 2010).

1.1.1 Definições básicas sobre complexidade em grafos

Seja S um conjunto não vazio próprio de vértices do grafo, que assumiremos ser o conjunto de indivíduos inicialmente infectados. Seja $I(S)$ o conjunto dos vértices infectados por S. Obviamente temos que todo vértice inicialmente infectado também pertence ao conjunto $I(S)$ que é, por muitos, chamado de passo ou marcação infecciosa de S. No caso do conjunto $S = I(S)$, dizemos que o conjunto S é inofensivo para a infecção, ou seja, não infectará nenhum vértice do grafo, pois depois de infectado, S tem o mesmo conjunto S e, o nome desse conjunto será *convexo*. Observe que caso S seja convexo jamais teremos uma contaminação generalizada no grafo inteiro e isso é de muita importância, pois o

interesse de toda pesquisa nesse tipo de infecções é, em geral, descobrir modos e formas de proteger os indivíduos da contaminação, além de entender o processo combinatório e computacional envolvido na infecção.

1.1.2 Outras aplicações de convexidade em grafos

Além dos problemas essencialmente logísticos, citamos aqui o enorme uso de grafos em redes sociais, já que estamos vivenciando, hoje, em um mundo muito conectado, onde decisões importantes da vida humana já são tomadas à distância por duas ou mais pessoas apenas enviando ou recebendo uma mensagem de iMessage, Whatsapp ou utilizando o Facebook. Comunidades humanas com as suas mais variadas formas de relacionamentos são também descritas e modeladas por grafos, tendo assim diversos de seus questionamentos subordinados às leis combinatórias que descrevem várias propriedades dos agentes envolvidos. Em trabalhos como (RUFINO et al., 2018), (RUFINO et al., 2019) aplica-se algumas dessas descrições na resolução de problemas epidemiológicos. No trabalho (CHEN, 2009) foi estudado a propagação de influência através de uma rede social modelada por um grafo, onde foi definido, para cada nó do grafo, um valor fixado chamado de threshold desse nó, de modo que um nó do grafo adotará ou comprará um novo produto comercializado nessa rede, se esse nó tiver um número de vizinhos igual ao valor desse parâmetro chamado de threshold. O objetivo do estudo é encontrar um pequeno conjunto S de nós, tal que se o produto a ser vendido for consumido pelos nós de S, então tenderá a ser também comprado ou consumido por um grande número de outros nós do grafo e o principal resultado desse estudo foi que, tentar encontrar qual será o menor conjunto S que tem a propriedade de influenciar o consumo demasiado no grafo, é uma tarefa bem difícil do ponto de vista computacional, no caso tentar uma aproximação de uma solução minimizada de S já não seria fácil. Alguns outros resultados sobre influência em redes sociais podem ser encontrados nos trabalhos (KEMPE; KLEINBERG; TARDOS, 2003), (KEMPE; KLEINBERG; TARDOS, 2005), (MOSSEL; ROCH, 2007), (PASTOR-SATORRAS; VESPIGNANI et al., 2003).

1.2 RELAÇÃO ENTRE CONVEXIDADE EM GRAFOS E EPIDEMIAS

1.2.1 Conceitos gerais

A convexidade em grafos e o problema da disseminação de epidemias em grafos estão relacionados porque a teoria da convexidade pode ser usada para modelar a disseminação de epidemias em grafos.

A teoria da convexidade em grafos é usada para estudar as propriedades dos conjuntos convexos em grafos, que são conjuntos de vértices que incluem todos os caminhos, satisfazendo determinada propriedade, entre quaisquer dois vértices do conjunto. A teoria da

convexidade em grafos é usada em várias aplicações práticas, incluindo a disseminação de epidemias em grafos.

No problema da disseminação de epidemias em grafos, o objetivo é determinar como uma doença se espalha em uma rede de contatos. O modelo mais comum para este problema é o modelo SIR (suscetível-infetado-recuperado), em que os indivíduos em um grafo são classificados em três categorias: suscetível, infetado e recuperado. O modelo considera que um indivíduo suscetível pode se infectar se estiver em contato com um indivíduo infetado, após o que ele se torna infetado, e eventualmente se recupera.

O que está fora do escopo deste trabalho. Neste trabalho consideramos apenas epidemias do tipo SI, ou seja, aquelas nos quais um indivíduos infectado permanece infectado para sempre. Epidemias do tipo SIR e variantes estão fora do escopo deste trabalho. Além disso, consideramos apenas transições entre estados, no qual um nó infecta vizinhos, mas não consideramos o tempo entre transições. Em particular, modelos em que tempos de disseminação são exponencialmente distribuídos estão fora do escopo deste trabalho.

O que está no escopo deste trabalho. A teoria da convexidade em grafos será usada para modelar a disseminação de epidemias em grafos. A ideia é que a epidemia se espalha a partir de um conjunto inicial de vértices para outros vértices do grafo, até que todo o grafo seja infectado. A teoria da convexidade em grafos pode ser usada para estudar as propriedades dos conjuntos cujo fecho convexo engloba todo grafo, e para prever a rapidez com que a epidemia se espalhará em todo o grafo, sendo o tempo de propagação medido em número de transições de estados.

1.2.2 Conjunto de vértices mínimo para que epidemia se propague por todo grafo

A teoria da convexidade pode ser usada para determinar o número mínimo de nós a serem contaminados no grafo, necessários para fazer com que a epidemia se propague pelo grafo todo. O número de elementos neste conjunto de vértices mínimo é chamado de *número envoltório*. Diz-se que o conjunto de vértices mínimo *cobre* o grafo todo, ou seja, o *fecho convexo* do conjunto de vértices é o grafo todo, permitindo que a epidemia se propague a partir deste conjunto para todos os outros vértices do grafo.

1.3 OBJETIVO

Neste trabalho, apresentamos aplicações de convexidade P_3 em grafos na área de Sistema de Recomendação. O objetivo é modelar a propagação de uma recomendação em uma rede de modo que um conjunto mínimo de usuários possam recomendar itens aos demais usuários associados com o máximo de aceitação e menor perda de informação repassada. De forma mais ampla, visamos fazer uma previsão de como um conjunto de indivíduos pode afetar os demais.

1.4 CONTRIBUIÇÕES

Esse trabalho é continuação do artigo sobre Sistemas de Recomendação usando influência coletiva (PONCIANO, 2019) onde os autores criaram um modelo matemático para o problema que vamos abordar. Nosso trabalho foi desenvolver um modelo computacional em Python, usando um algoritmo randomizado e aproximativo, para achar o mínimo número de nós a ser infectado no grafo para que a epidemia se propague pelo grafo todo, tendo em vista que problema estudado na monografia é NP-completo.

Todos os algoritmos implementados ao longo deste trabalho de conclusão de curso estão publicamente disponíveis no Github:

- <https://github.com/xiaoyongkong/TCC>

1.5 ROTEIRO

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 apresentamos noções básicas sobre grafos e complexidade. No Capítulo 3 apresentamos resultados númericos obtidos com os algoritmos implementados, e o Capítulo 4 conclui. Finalmente, nos apêndices apresentamos observações finais sobre trabalhos relacionados.

2 NOÇÕES PRELIMINARES

Bem-aventurado o homem que acha sabedoria e o homem que adquire conhecimento;

Provérbios 3:13

Alguns conceitos serão apresentados neste capítulo que facilitarão o entendimento do problema estudado. Como referência básica sobre grafos temos o livro (SZWARCFITER, 2018; BONDY; MURTY, 2011; WEST, 2017) e sobre convexidade (PELAYO, 2013). Vamos considerar apenas grafos finitos, conexos e simples.

2.1 CONCEITOS E DEFINIÇÕES INICIAIS DE GRAFO

Podemos dizer que um grafo G é uma estrutura composta por dois subconjuntos finitos neste trabalho: $V(G)$ é o subconjunto de *vértices*, e $E(G)$ é subconjunto de pares não ordenados de elementos tomados de $V(G)$, de *arestas*. Uma aresta $e = (u, v) \in E(G)$ é formada pelo par de vértices $u, v \in V(G)$, neste caso u e v são vértices *adjacentes*. Dizemos também que e é *aresta incidente* a u e v. Denotamos a *tamanho* de $|V(G)| = n$ e $|E(G)| = m$. Outra definição de *vizinhança aberta* de um vértice $v \in V(G)$ é denotada $N(v) = \{u \in V(G) | (u, v) \in E(G)\}$. Já a *vizinhança fechada* de um vértice $v \in V(G)$ é denotada $N[v] = N(v) \cup \{v\}$. Um grafo orientado às vezes é também chamado de dígrafo. O *grau de um vértice* v, denotado por $d(v)$, corresponde ao número de vértices adjacentes a v, ou seja, a cardinalidade de $|N(v)|$. O *grau máximo* de um grafo G é denotado por $\Delta(G) = max\{d(v) | v \in V(G)\}$. De forma similar o *grau mínimo* é denotado por $\delta(G) = min\{d(v) | v \in V(G)\}$.

Dado um grafo G, e um vértice $v \in V(G)$, o grafo $G \backslash \{v\}$ é obtido a partir de G retirando-se o vértice v de seu conjunto de vértices, e retirando-se também todas arestas de $E(G)$ incidentes a v. De forma semelhante, dada uma aresta $e \in E(G)$, o grafo $G \backslash \{e\}$ é obtido a partir de G retirando-se a aresta e de $E(G)$.

Dizemos que $G' = (V', E')$ é um *subgrafo* de um grafo $G = (V, E)$, quando $V' \subseteq V$ e $E' \subseteq E$. Quando o subgrafo G' contém todas as arestas de E cujas extremidades estão contidas em V', então G' é o *subgrafo induzido* de G por V'.

Um grafo G é um *ciclo*, que denotaremos por C_n, se ele é uma sequência de vértices $v_1, \ldots, v_n, v_1$ distintos, onde $v_i \neq v_j$ para $i \neq j$ e $(v_i, v_{i+1}) \in E(G)$, tal que $n \geq 3$.

Um grafo que não possui ciclos é dito *acíclico*.

Um caminho $\mathcal{P}$ de tamanho k, ou de ordem $k + 1$ entre dois vértices v_0 e v_k é uma sequência de vértices distintos $v_0, v_1, ..., v_k$, onde $v_i v_{i+1} \in E$, para $0 \leq i < k$. Dizemos, nesse caso, que v_0 e v_k são as extremidades desse caminho e que v_i , para $0 < i < k$ são os vértices internos de P.

Um grafo G é *conexo* se existe um caminho entre qualquer par de vértices de G. Um grafo é uma *árvore* quando é acíclico e conexo. Um subgrafo conexo de uma árvore é dito *subárvore*. A distância entre dois vértices u e v, denotada por $d_G(u, v)$, é o tamanho do menor caminho entre u e v.

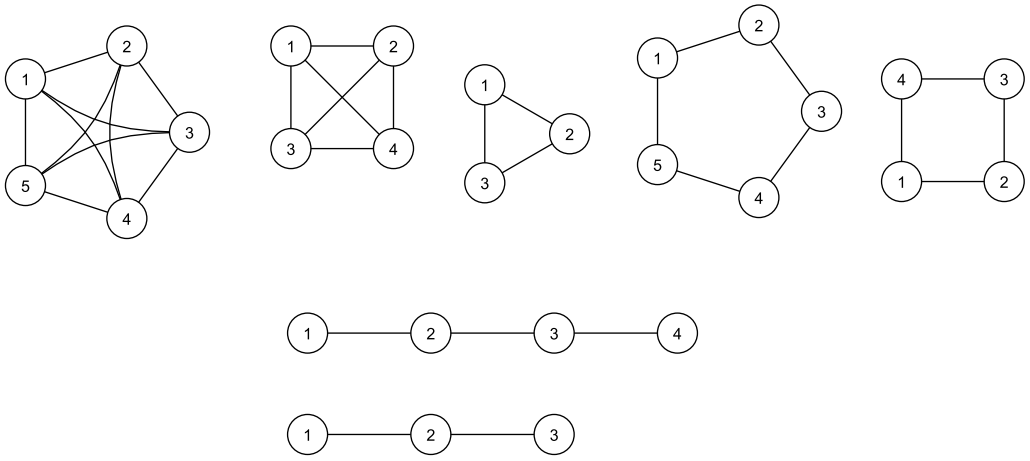

Figura 2 – Exemplos de Grafos

2.2 CONVEXIDADE EM GRAFOS

Na geometria, um conjunto do plano euclidiano é definido convexo quando o seguimento de reta ligando quaisquer dois pontos desse conjunto estiver, inteiramente, contido nesse conjunto.

Vamos agora definir a convexidade em grafos. Todas as definições citadas aqui, inclusive outras correlatas, podem ser encontradas em (PELAYO, 2013).

Definição 1 (Espaço Convexo)**.** *Uma coleção $\mathcal{C}$ de partes de um conjunto não vazio e finito V é dita uma convexidade sobre V quando:*

1) $\emptyset, V \in \mathcal{C}$.

2) $\mathcal{C}$ *é fechada para interseção.*

Um espaço convexo é um par $(V, \mathcal{C})$, onde V é um conjunto não vazio e $\mathcal{C}$ uma convexidade sobre V. Os elementos da coleção $\mathcal{C}$ são chamados de conjuntos ***convexos****.*

Exemplo 1. *Dado o conjunto $V = \{a, b, c, d, e\}$ e a família de partes de V, dada por $\mathcal{C} = \{\emptyset, \{a\}, \{b\}, \{a, b\}, V\}$. Temos que a família $\mathcal{C}$ verifica as condições mencionadas anteriormente, ou seja, $\mathcal{C}$ é uma convexidade sobre V.*

Seja G um grafo e $u, v \in V$, um vértice x é dito ser *uv-gerado* ou simplesmente *gerado* pelo par de vértices $\{u, v\}$, quando x pertencer a algum uv-caminho *considerado pela convexidade em questão*. O intervalo $I[u, v]$ dos vértices u e v é o conjunto formado por esses vértices juntamente com todos os vértices que forem uv-gerados.

$$I[u, v] = \{u, v, w; \text{onde } w \text{ é vértice } uv\text{-gerado}\}$$

Seja $S \in V$. O intervalo $I[S]$ sobre o subconjunto de vértices S é definido como a união dos intervalos de todos os pares de vértices de S, ou seja.

$$I[S] = \bigcup_{u,v \in S} I[u, v]$$

Observe que, da própria definição, sempre temos $S \subset I[S]$, no entanto a recíproca dessa inclusão pode não ser verdadeira, já que um subconjunto de vértices do grafo pode não conter outros vértices gerados por seus pares de vértices

No caso em que tal igualdade ocorra $I[S] = S$, o conjunto S é dito ser *convexo*, observe que essa convexidade ocorre sobre o conjunto dos vértices $V(G)$ do grafo. Na mesma figura 3 temos que o conjunto $S_2 = \{c, d, e, f, g\}$ marcado no grafo à esquerda é um conjunto convexo, onde o intervalo considera os caminhos geodésicos, ou seja, o caminho de comprimento mínimo entre pares de vértices u e $v \in S$.

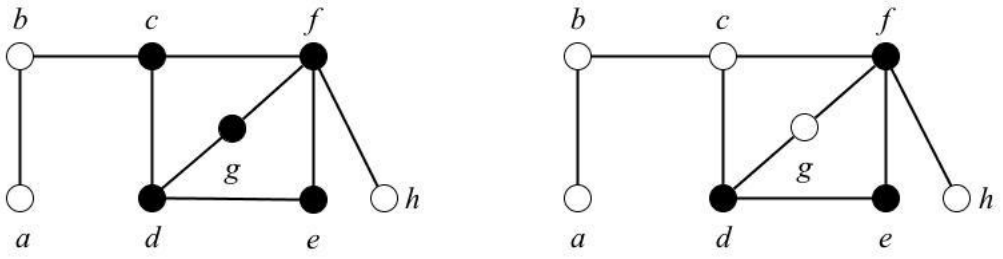

Figura 3 – O conjunto de vértices $S_2 = \{c, d, e, f, g\}$ marcado no grafo à esquerda é convexo quando se considera o intervalo dos caminhos geodésicos no grafo. Já o conjunto $S_1 = \{d, e, f\}$ marcado no mesmo grafo à direita, não é convexo sob as mesmas considerações, pois $I[S_1] = \{c, d, e, f, g\} \neq S_1$.

A partir da definição de conjunto convexo sobre V, é possível também definir um outro importante parâmetro do grafo relacionado à convexidade, denominado de *número de convexidade* do grafo, ver (CHARTRAND; WALL; ZHANG, 2002) e (DOURADO et al., 2012), que é o tamanho do maior subconjunto convexo próprio de $V(G)$.

- **Fecho convexo:** O fecho convexo $H[S]$ de um subconjunto de vértices S de um grafo, é definido como o menor conjunto convexo que contem S
- **Conjunto envoltório:** Quando $H[S] = V(G)$ o conjunto S é chamado de *conjunto envoltório* do grafo

- **Conjunto envoltório mínimo:** Um conjunto envoltório cujo número de vértices é menor ou igual a qualquer outro conjunto envoltório

- **Número envoltório:** O tamanho do menor conjunto envoltório de um grafo é um outro importante parâmetro de convexidade, conhecido como *número envoltório* do grafo (DOURADO et al., 2009).

Observe que esses parâmetros definidos acima, estão todos atrelados e relacionados a uma convexidade em questão no grafo, pois para sua definição usamos noções de intervalo e fecho convexo que, por sua vez, se relacionam com alguma convexidade em questão, fazendo com que, dessa forma, mudando a convexidade possa se mudar todo o processo computacional para a busca do parâmetro estudado. Uma das mais conhecidas e estudadas convexidades em grafos é definida abaixo. Essa convexidade, a convexidade P3, é uma convexidade de caminhos, ou seja, o espaço de convexidade em questão é formado por uma coleção de caminhos do grafo.

2.2.1 A Convexidade P3

A *convexidade* P_3 é um tipo importante e muito estudado de convexidade considerada em um grafo. Essa convexidade se caracteriza quando os caminhos considerados para dois vértices gerarem um terceiro vértice, na computação do intervalo, e forem caminhos de tamanho dois, ou seja, caminhos do tipo P_3. Desse modo, um subconjunto $S \subset V$ é dito ser convexo na convexidade P_3 ou ser P_3-convexo, quando todo vértice pertencente a caminhos P_3 entre pares de vértices de S, estiver também em S, ou seja, quando todo vértice de fora de S tem no máximo um vizinho em S, dito de outro modo $I[S]_{P_3} = S$, onde $I[S]_{P_3}$ é o operador de computação do intervalo para esta convexidade (CENTENO, 2012b).

Em 2011 foi mostrado no trabalho (CENTENO et al., 2011) que a determinação do *número envoltório*, na convexidade P_3 para grafos gerais é NP-difícil.

2.3 CLASSE DE GRAFOS

- Passeio: é uma sequência alternante de vértices e arestas $v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k$, que começa e termina em vértices.

- Caminho: é um passeio na qual todos os vértices são distintos.

- Ciclo Simples: Quando, o primeiro vértice e o último vértice do caminho tem arestas entre si, dizemos que é um ciclo simples.

- *Árvore*: é um grafo conexo (existe caminho entre quaisquer dois de seus vértices) e acíclico (não possui ciclos)

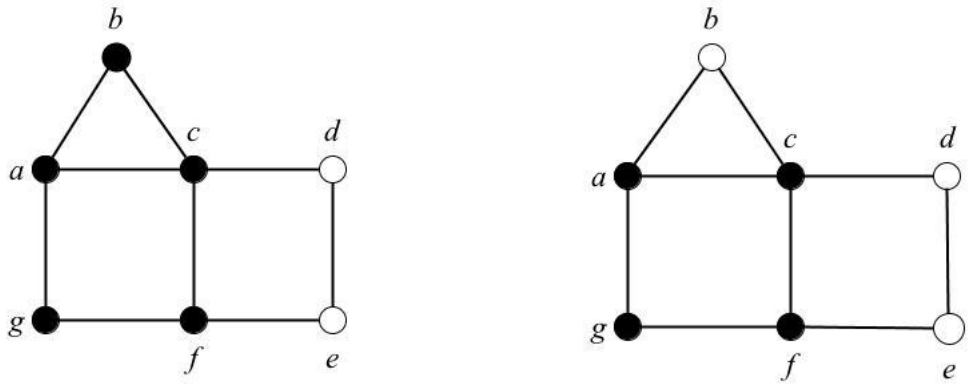

Figura 4 – O conjunto $S_1 = \{a, b, c, g, f\}$ marcado no grafo à esquerda é P_3-convexo, equanto que o conjunto $S_2 = \{a, c, g, f\}$ marcado no grafo à direita não é P_3-convexo, pois o vértice b está em um caminho P_3 de extremos a e c sem pertencer a S_2.

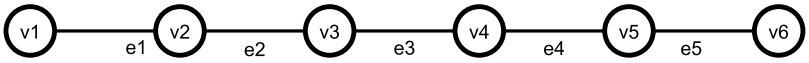

Figura 5 – Ilustração da caminho simples

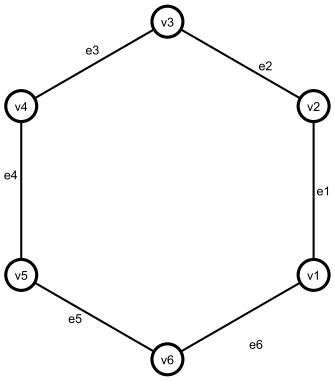

Figura 6 – Ilustração da ciclo simples

- Grade: uma grade de dimensão p×q é um grafo cujos vértices são os pontos de coordenadas inteiras (x, y) tal que $1 \leq x \leq p$ e $1 \leq y \leq q$ e tal que dois vértices são adjacentes se, e somente se, a distância entre os pontos é igual a 1.
- Cordal: Um grafo é cordal se todo ciclo de comprimento pelo menos 4 tem uma corda.

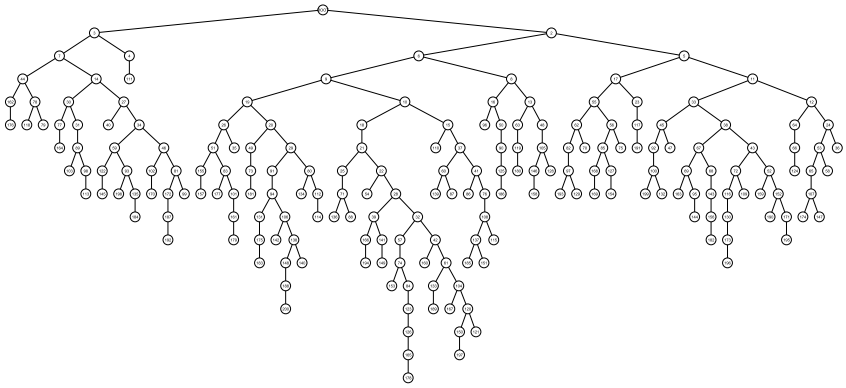

Figura 7 – Ilustração de árvore

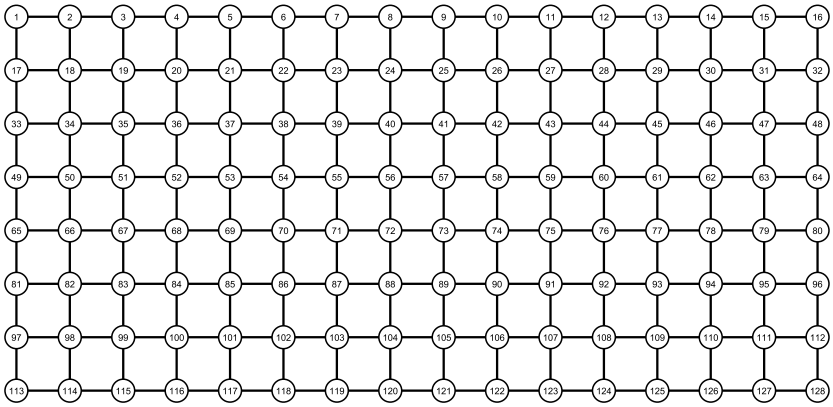

Figura 8 – Ilustração de grade

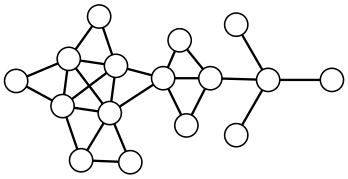

Figura 9 – Ilustração de grafo cordal

2.4 COMPLEXIDADE DE ALGORITMOS

Um *problema computacional* é um par (I, Q), onde I é o conjunto de todas as possíveis entradas para o problema chamado de conjunto das *instâncias* e de uma questão sobre essas instâncias Q. Dessa forma, resolver um problema computacional é desenvolver um algoritmo correto cuja entrada é uma instância do problema e cuja saída é uma resposta à questão levantada no problema. Um problema computacional é dito ser de *decisão* quando exigir uma resposta do tipo sim ou não, apenas. Dizemos que um algoritmo é *polinomial* ou de *tempo polinomial*, quando sua complexidade de tempo, que pode ser pensada como a medida do número de passos que o algoritmo executa na busca pela solução do problema, for limitada por uma função polinomial, no tamanho da entrada.

Os problemas de decisão para os quais existem algoritmos polinomiais, que os resolvem, formam uma classe de complexidade chamada de classe *P*. Um problema de decisão é dito ser *não determinístico polinomial* quando possuir um algoritmo polinomial que certifique a resposta SIM. É importante que essa certificação ocorra em um tempo sucinto no tamanho da entrada do problema. Os problemas de decisão com esta característica mencionada formam uma classe de complexidade denominada de *NP*. A classe Co-NP é formada pelos problemas que possuem um certificado sucinto para as instâncias que produzem resposta NÃO.

Sejam $\Pi_1 = (I_1, Q_1)$ e $\Pi_2 = (I_2, Q_2)$ dois problemas computacionais de decisão. Uma *redução polinomial* do problema Π_1 no problema $\Pi_2 = (I_2, Q_2)$ é uma $f : I_1 \longrightarrow I_2$ com as seguintes propriedades:

1) f pode ser calculada em tempo polinomial.

2) Toda instância I de I_1 produz uma resposta SIM para Π_1 se, e somente se $f(I)$ produzir resposta SIM para I_2.

Um problema de decisão Π pertence à classe *NP-completo* quando as seguintes condições forem satisfeitas:

1) $\Pi \in NP$.

2) Todo problema $\Pi' \in NP$ puder ser reduzido polinomialmente a Π.

Um problema computacional Π que pertença à classe dos NP-completos é dito ser um problema *NP-Completo*. E caso este problema não necessariamente pertença a classe NP, então o mesmo passa a se chamar de *NP-difícil* e pertencerá a uma classe chamada de classe dos problemas *NP-difíceis* (SZWARCFITER, 1988).

2.5 ALGORITMO DE BUSCA BINÁRIA

A pesquisa ou busca binária (em inglês binary search algorithm ou binary chop) é um algoritmo de busca em vetores que segue o paradigma de divisão e conquista. Ela parte do pressuposto de que o vetor está ordenado e realiza sucessivas divisões do espaço de busca comparando o elemento buscado (chave) com o elemento no meio do vetor. Se o elemento do meio do vetor for a chave, a busca termina com sucesso. Caso contrário, se o elemento do meio vier antes do elemento buscado, então a busca continua na metade posterior do vetor. E finalmente, se o elemento do meio vier depois da chave, a busca continua na metade anterior do vetor.

Dado uma lista A de n elementos com os valores $A_0, A_1, AA_2, ..., A_{n-1}$ ordenada de tal modo que $A_0 \leq A_1 \leq A_2 \leq ... \leq A_{n-1}$, e um valor para pesquisa T, a seguinte rotina usa pesquisa binária para achar o índice de T em A.

1. Defina L para 0 e R para $n-1$
2. Se $L \geq R$ a pesquisa termina sem sucesso
3. Defina m (o índice do meio da lista) para $\frac{L+R}{2}$ arredondado
4. Se $A_m < T$, defina L para $m+1$ e volte ao segundo passo
5. Se $A_m > T$, defina R para $m-1$ e volte ao segundo passo.
6. Se $A_m = T$, a pesquisa está feita, o índice de T é m.

2.5.1 Envoltória Convexa

O intervalo P3 entre dois vértices u e v, $I[u, v]$, consiste de u, v e todos os vértices dos caminhos de comprimento dois entre o par de vértices u, v. Sendo assim, o intervalo P3 de um conjunto de vértices S , $I[S]$, é a união de todos $I[u, v]$ para $u, v \in S$.

Considere agora que a operação intervalo P3 será aplicada sucessivas vezes,$I[S]$; ou seja, dado um conjunto S achamos $I[S]$, em seguida, $I[I[S]]$, e mais uma vez, $I[I[I[S]]]$, continuamos esta operação até que não seja possível adicionar mais vértices ao conjunto. Isto acontece porque obtivemos um conjunto $P3$ convexo, este conjunto é chamado de envoltória $P3$ convexa de S. Definimos então a envoltória $P3$ convexa, $hull(S)$, como sendo o menor conjunto P3 convexo contendo S. Quando $hull(S) = V(G)$, temos uma envoltória $P3$. Chamaremos o conjunto de vértices S de conjunto da envoltória P3. O número de envoltória $P3$ é a cardinalidade do conjunto da envoltória $P3$ mínimo, e será chamado de $P3 - hull$ set. Exemplo na Figura 16.

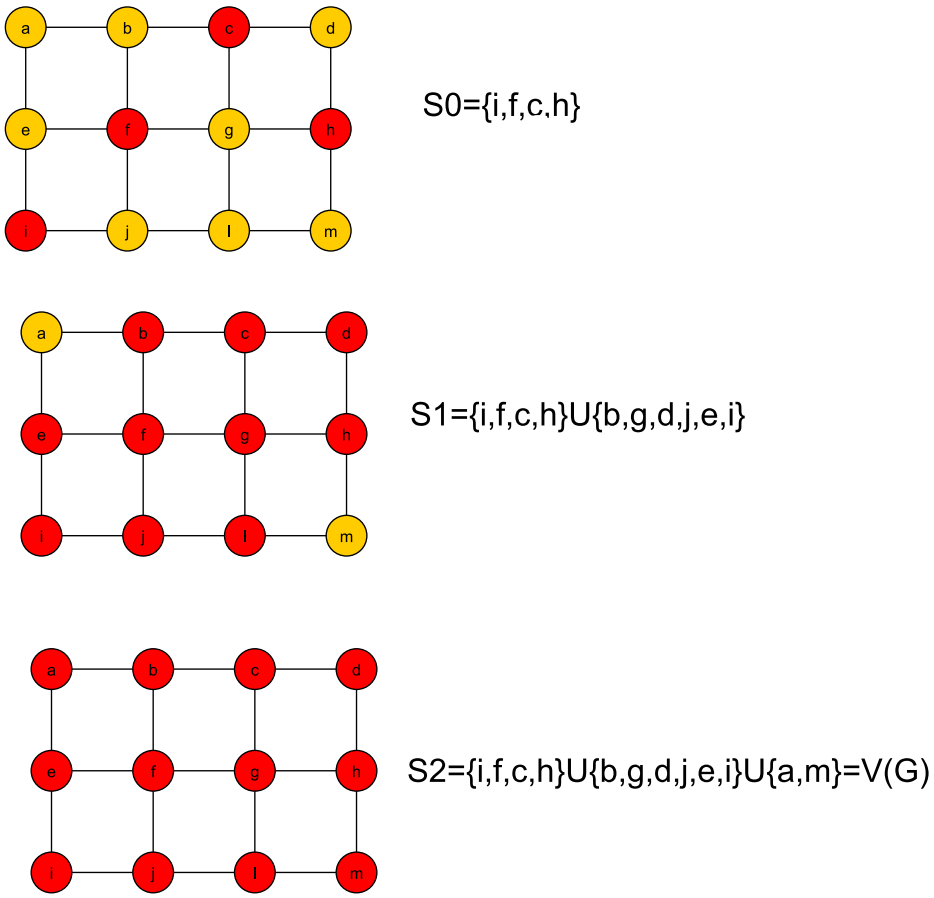

Figura 10 – Na figura o conjunto S0 é um conjunto de envoltória P3 do grafo. Cada vértice de S1 é contaminado, pois cada um tem dois vizinhos em S0 que conjunto contaminação inicial. Em cada interação um vértice contaminado passa a ser um contaminador, assim sucessivamente até que todos vértices sejam contaminados.

3 TESTES COMPUTACIONAIS E APROXIMATIVOS

Segundo os autores do artigo Número envoltório na convexidade p3: Resultados e aplicações(NASCIMENTO; FERREIRA; COELHO, 2020) os estudos da convexidade $P3$ em grande parte são teóricos. Nosso objetivo nesse capítulo é a partir dos estudos teóricos fazer estudos empíricos do problema para verificar possibilidades de solução de problemas reais no sentido de analisar a efetividade e os impactos do conceito de influência de modo prático, assim como nos artigos threshold influence propagation model (KEMPE; KLEINBERG; TARDOS, 2015) e On Finding Small Sets That Influence Large Networks (Cordasco; Gargano; Rescigno, 2016), quando os autores descrevem cenários fictícios onde os conceitos de propagação de influencias poderiam ser aplicados e também realizam experimentos em data sets de redes sociais reais como Delicious, Facebook, Flikr, Higgs-twitter, Last.fm, Livemocha, YouTube, dentre outras.

O algoritmo sempre encontrará um conjunto de envoltória, porém nem sempre será o de menor cardinalidade. A aproximação do resultado deste algoritmo será avaliada nas próximas seções em que serão comparadas a aproximação de cada resultado individualmente, categorizado por classes específicas de grafos.

3.1 PROPONDO UMA NOVA MODELAGEM DO PROBLEMA

Queremos encontrar o menor conjunto *better influencer* S que infecta $V(G)$ em tempo t, utilizando caminhos P_3, ou seja, queremos o P_3-*hull set*. Um exemplo de iteração pode ser encontrado na Figura 11.

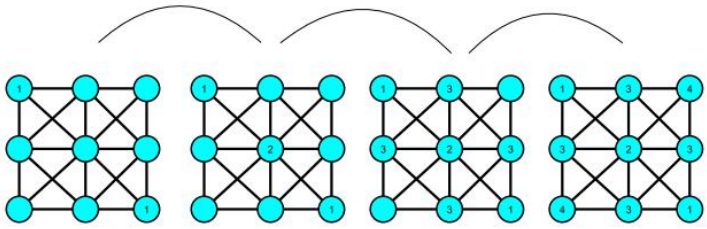

Figura 11 – Ilustração da contaminação P_3 com tempo $t = 4$.

Dado um conjunto S inicial é possível validar se trata-se de um P_3-*hull set* em tempo polinomial $O(N + M)$, onde N é o número de vértices do grafo G e M é o número de aresta deste mesmo grafo. Queremos encontrar o menor S possível.

Perceba também que existirá um conjunto F de vértices de G que estarão obrigatoriamente no P_3-*hull set* pois caso existam vértices isolados, ou seja, de grau 0, ou ainda vértices de grau 1, estes tipos necessariamente precisam originar um caminho para contaminação total do grafo. Essa é a solução inicial trivial mas geralmente não suficiente para o menor P_3-*hull set* que satisfaça a infecção total de G.

A solução backtracking para esse problema é, geralmente, muito custosa. Pois dado o restante dos vértices do conjunto $W = V(G) - F$ seria necessário validar todos os subconjuntos W' de W tal que a cardinalidade $W' + F$ fosse a menor possível. O número de subconjuntos de W dado que seu tamanho seja K seria 2^K, sendo necessário uma validação na complexidade de tempo $O(N + M)$ para cada um desses subconjuntos. No caso mais geral, onde não há conjunto F inicial trivial, nota-se que $W = V(G)$, ou seja, $K = N$ e a busca pelo subconjunto desejado com backtracking exigiria a complexidade $O((N + M) * 2^N)$, a qual cresce exponencialmente, sendo inviável para tratar problemas reais desse tipo mesmo para grafos de tamanho reduzido.

A primeira proposta aqui não é melhorar a complexidade de tempo de validação $O(N + M)$, apesar do caso em que haja avanços na melhoria deste tempo de validação o algoritmo aqui descrito também contará com melhorias. Por outro lado podemos melhorar a complexidade de tempo de busca, anteriormente 2^K, por um tempo logarítmico. Podemos notar que para a modelagem proposta o que importa é achar o conjunto W' de menor cardinalidade, assim podemos realizar uma busca binária pelo tamanho do conjunto final mantendo o intervalo de busca inicial em $[0, K]$. Fixado um iterador da busca binária J tal que $0 <= J <= K$, para chegar a resposta ótima seria preciso gerar todos os subconjuntos de W' com tamanho J o que ainda seria exponencial. O que acrescentamos aqui é uma variável Y que representa o número de amostras de subconjuntos de tamanho J com intuito de encontrar uma solução suficientemente prática, não necessariamente a ótima. Embora não seja garantido encontrar a solução ótima, acrescentando essa variável Y será mostrado que para algumas classes de grafos teremos um bom grau de aproximação. A estratégia ao acrescentar validar em Y amostras de tamanho J é uma heurística para tornar o número de subconjuntos linear em Y.

Podemos ver na figura a seguir um exemplo do algoritmo de busca binária especificado acima. Imagine que o grafo tenha 9 vértices. A princípio podemos ter uma envoltória convexa em que todos os vértices do grafo sejam necessários. Porém não é razóavel. Então podemos tentar encontrar conjuntos aleatório de tamanho 5, ou seja, a metade do número de vértices do grafo, que constituam um conjunto mínimo para alcançar o grafo inteiro com a operação de convexidade. Realizamos Y amostragens, verificando a validade disso. Como não foi encontrado nas Y amostras conjuntos que alcancem todo o grafo, então sabemos que o conjunto mínimo, possivelmente, tem tamanho maior que 5, e eliminamos as buscas por conjuntos de vértices de tamanho menor. Após isto, verificamos que o tamanho do conjunto estará entre 6 e 9, sendo o elemento do meio o 8. Tentamos

averiguar novamente Y amostras de tamanho 8 vértices que alcancem o grafo inteiro pela operação de convexidade. Uma vez encontrado um subconjunto entre as Y amostras sabemos que o tamanho do menor subconjunto estará possívelmente entre 6 e 8, sendo agora o elemento intermediário deste intervalo igual a 7. Testamos o alcance da operação com Y amostras de 7 elementos e novamente encontramos um conjunto, o que significa que o tamanho do menos subsconjunto será entre 6 e 7. Testamos conjuntos amostrais de tamanho 6 e não encontramos nenhum que alcance todo o grafo pela operação de convexidade. Logo, neste caso, o menor subconjunto será o de tamanho 7.

Figura 12 – Iterações da busca binária no tamanho do conjunto de vértices

A escolha da busca binária não é aleatória. Uma busca linear já seria melhor que a solução em backtracking limitando-se a Y amostras. Porém, acrescentando-se essa variável Y perdemos muita precisão em encontrar a solução ótima mas poderia-se melhorar isso executando-se tantas buscas a mais quantas fossem necessárias, mantendo-se a linearidade no tamanho de número de buscas. Isso acrescentaria um grau polinomial a mais na complexidade de buscas. Para aproveitar-se desse fator (executar-se tanto mais buscas) sem aumentar necessariamente o grau polinomial da complexidade, o que é um problema prático, utilizar um algoritmo logarítmico como a busca binária é um artifício complementar ótimo.

O processo de execução de outras buscas binárias relatados ocorreriam como na imagem a seguir. A diferença entre o exemplo de busca anterior e este é que a cada novo subconjunto encontrado o tamanho do subconjunto mínimo a ser avaliado volta a ser ao

menos 1 vértice do grafo. Isso é o equivalente a dizer que há mais oportunidade de busca pelo conjunto de tamanho mínimo. Por exemplo, ao averiguarmos via amostragem que não existem conjuntos de tamanho 5 que contaminem o grafo inteiro é razoável dizer que o intervalo de busca nos tamanhos de 1 a 5 são inviabilizados. Mas, no momento em que encontramos um subconjunto de tamanho 8 que alcance o grafo inteiro recomeçamos o intervalo da busca binária possibilitando que conjuntos de tamanho 1 a 5 possam ser testados novamente pois podem ter havido problemas devido ao processo amostral. Esse tipo de algoritmo provou-se nos testes mais promissor pois encontrava conjuntos de tamanhos menores.

Figura 13 – Iterações da busca binária, com recomeço, no tamanho do conjunto de vértices

Além da amostragem tornar o número de subconjuntos linear temos outra vantagem: a possibilidade de paralelizar a validação das Y amostras. Nas análises desse trabalho não entraremos nesse detalhe que impacta principalmente no tempo real de execução, mas uma das propostas principais fica mantida: a possibilidade de escalar o algoritmo de modo prático.

Por fim, o processo que tornou o algoritmo ainda melhor foi extrair informações do processo de amostragem. Por exemplo, a cada amostragem conseguimos entender a importância de um vértice no resultado final pois testamos se com aquele vértice amostrado o alcance do grafo melhorou de tal maneira a contaminar todos os vértices fora do conjunto amostral. Se isso ocorre, é possível atribuir um certo peso a este vértice na solução final, embora seja possível que esse vértice ainda não esteja na solução ótima. Se ex-

trairmos essa informação de importância do vértice a cada amostragem, a cada iteração da busca binária teremos um processo mais direcionado para realizar novas amostragens, pois se escolhermos esse vértice com base no seu peso atribuído, estamos trabalhando na convergência para uma solução ótima. E isso é exatamente o que o algoritmo faz. A cada iteração de busca binária existe, para cada vértice, uma probabilidade deste vértice estar na solução ótima. Então é realizada um processo de amostragem, sem repetição e com pesos para selecionar melhores conjuntos amostrais. Essa ideia reflete a noção de que em possíveis soluções locais possa existir parte de uma solução ótima, como em (SALES GUILHERME DA COSTA, 2019), o que ajuda os vértices melhores a se manterem sendo escolhidos nesse processo evolutivo probabilístico.

A forma prática com que fazemos a atualização de pesos é da seguinte forma: dado que seja encontrado um subconjunto de vértices que contamine todo grafo, damos a esse subconjunto, X por cento de probabilidade de reaparecer, exatamente, numa amostragem em uma próxima iteração de busca binária.

3.2 ALGORITMO PROPOSTO

Dado que o problema de decisão do número envoltório é NP-completo (CENTENO et al., 2011), obter um algoritmo aproximativo pode ser viável para aplicações desse parâmetro em rede sociais. Com este viés, a abordagem de um algoritmo de busca binária para o problema foi produzido nos seguintes moldes:

```
Chamada inicial: BUSCA-BINÁRIA(G, 0, n)

BUSCA-BINÁRIA(G, início, fim)
    se (inicio = fim) entao
        encontrada resposta, fim
    senão
       i = (início + fim) / 2
       se (EXISTE-ENVOLTORIA(G, i)) então
          faça a BUSCA-BINÁRIA(G, inicio, i - 1)
       senão
          faça a BUSCA-BINÁRIA(G, i + 1, fim)
       fimse
    fimse


EXISTE-ENVOLTORIA(G,i)
    sorteie de forma uniforme e aleatoria s subconjuntos de tamanho i de G
    para cada subconjunto S, verificar se hull(S) = V(G)
```

```
caso exista ao menos um subconjunto S tal que hull(S) = V(G),
     retornar VERDADEIRO
caso contrário,
     retornar INCONCLUSIVO
```

3.2.1 Explicando o algoritmo

A ideia do algoritmo é realizar sucessivas divisões a fim de encontrar o nosso melhor resultado. Ao aplicar o método "BUSCAS-BINÁRIA" no qual veremos o nosso resultado ótimo quando o início e o fim forem iguais. Caso não sejam, iremos dividir o nosso vetor pela metade (i) e verificar se o grafo possui o envoltório, através do método "EXISTE-ENVOLTORIA". Caso o envoltório encontrado esteja depois do nosso valor i, iremos continuar na parte superior do grafo e assim iremos realizar uma nova busca, caso contrário, a buscar será na parte inferior. Ao realizar uma nova busca, o lado que escolhemos será o nosso novo grafo e assim iremos realizar novamente a busca seguindo o mesmo processo de forma sucessiva até encontrarmos nosso valor ótimo.

```
def optimize(graph, flexible = False):
    # when flexible is True run a reset on minimum to restart binary search, it is equivalent to run binary search multiple times
    minimum = 1
    maximum = len(graph.available_hull())
    n = math.ceil(maximum / 2)
    first_hull_time = True
    first_hull = True
    while True:
        print('minimum: {}, maximum: {}, n: {}'.format(minimum, maximum, n))

        if os.getenv('PARALLEL') == 'True':
            hull, indexes = run_samples_parallel(graph, n) # on hold for future
        else:
            hull, indexes = run_samples(graph, n)

        if reach_threshold(hull, len(graph)):
            if first_hull_time or (hull.time <= hull_time.time and len(hull.initial_hull()) < len(hull_time.initial_hull())):
                first_hull_time = False
                hull_time = hull
            if first_hull or len(hull.initial_hull()) < len(hull_best.initial_hull()) or (len(hull.initial_hull()) == len(hull_best.initial_hull()) an
                if not first_hull and os.getenv('WITH_WEIGHT') == 'True':
                    graph.available_hull().update_weights(indexes)
                first_hull = False
                hull_best = hull
                print("tamanho do MELHOR FECHO INICIAL: {}".format(len(hull_best.initial_hull())))
                print("numero de vertices alcancados pelo MELHOR FECHO INICIAL: {}".format(len(hull_best)))
                print("tempo do MELHOR FECHO INICIAL: {}".format(hull_best.time))
                print()
                if flexible:
                    minimum = 1
            maximum = n
            n = (maximum + minimum) // 2
        else:
            minimum = n
            n = n + math.ceil((maximum - minimum) / 2) # maximum - max(math.ceil((maximum - n) / 2), 1) # (maximum - n) // 2
        if maximum - minimum <= 1:
            break
    return hull_best, hull_time
```

Figura 14 – Na figura temos parte do código do algoritmo binário

```
def infect(self, vcontaminant, vcontaminated):
    if vcontaminated in self.infection:
        if self.infection[vcontaminated] is None:
            return False
        if len(self.infection[vcontaminated]) < int(os.getenv('CONTAMINANTS')) and vcontaminant not in self.infection[vcontaminated]:
            self.infection[vcontaminated].add(vcontaminant)
            if len(self.infection[vcontaminated]) == int(os.getenv('CONTAMINANTS')):
                return True
    else:
        self.infection[vcontaminated] = set()
        self.infection[vcontaminated].add(vcontaminant)
        if len(self.infection[vcontaminated]) == int(os.getenv('CONTAMINANTS')):
            # só existe essa condição para no futuro podermos evoluir pra qualquer N >= 1, aqui no caso N=2
            return True
    return False
```

Figura 15 – Na figura temos parte do código do Existência Combinatória

3.2.2 Refinando o algoritmo

Para refinarmos os resultados do nosso algoritmo, utilizamos pesos na amostragem, mencionando inspirados em algoritmos genéticos ou Markov Chain Monte Carlo (MCMC)

Na literatura temos a discussão de processos de amostragem. Temos, por exemplo, o processo de amostragem por pesos sem repetição relatado em (EFRAIMIDIS; SPIRAKIS, 2006). Esse processo permite que para um dado conjunto W' seja possível em tempo linear priorizar com uma probabilidade acumulada de P um subconjunto de W', pela realocação de pesos. Um exemplo, caso tenhamos uma tupla $(1, 2, 3, 4)$ representando o conjunto $1, 2, 3, 4$, e seus respectivos pesos sejam $(1, 3, 3, 1)$, numa amostragem de 2 elementos desse conjunto de tamanho 4, teremos a probabilidade de 0.75 do conjunto $2, 3$ ser selecionado em uma amostragem por pesos sem repetição, pois o somatório do peso desses dois elementos $3 + 3 = 6$ representa 0.75 do total $(1 + 3 + 3 + 1 = 8)$.

Este tipo de artifício probabilístico é interessante pois podemos utilizar uma amostragem guiada em que partes de uma boa solução poderão pertencer a solução ótima no problema de convexidade $P3$. No algoritmo da modelagem proposta é mostrado que W' é uma amostra aleatória de W. Mas no processo de busca binária, uma vez conhecido o menor fecho convexo Z é possível selecionar com maior probabilidade elementos de $W \cap Z$ o que num longo processo de amostragem se mostrou interessante pois de fato boas soluções possuiam partes de soluções ótimas. Para grafos inflados, por exemplo, onde o problema é difícil ser tratado somente com o algoritmo de modelagem proposta apresentado, esse fator de amostragem com pesos sem repetição convergiu para soluções ótimas.

A escolha do W' inicial ainda é aleatória mas o que é mostrado aqui é ainda que houvesse um método heurístico para determinar um bom W' inicial que fosse o menor fecho

convexo $P3$ de G poderíamos utilizá-lo como etapa inicial dessa amostragem probabilística para possível melhoria.

```
def run_samples(graph, n):
    first = True
    for cnt in range(0, int(os.getenv('LENGTH_SAMPLE'))):
        random_hull, idx = graph.available_hull().random_subset(n, os.getenv('WITH_WEIGHT') == 'True')
        hull = graph.mandatory_hull() + random_hull
        hull = graph.hull_algorithm(hull)
        if first or (len(hull) > len(hull_best)) or (len(hull) == len(hull_best) and hull.time < hull_best.time):
            if not first and os.getenv('WITH_WEIGHT') == 'True':
                graph.available_hull().update_weights(indexes, True)
            first = False
            hull_best = hull
            indexes = idx
        if os.getenv('STOP_ON_FIRST_BEST_SAMPLE') == 'True' and reach_threshold(hull, len(graph)):
            break
    return hull_best, indexes
```

Figura 16 – Na figura temos parte do código da amostragem com peso

3.2.3 Comentários sobre o desempenho

Verificamos empiricamente que o algoritmo é capaz de encontrar a menor envoltória em inúmeros casos.

- grade: surpreendentemente, esse foi um dos casos mais desafiadores para algoritmos clássicos, e nosso algoritmo foi capaz de encontrar uma solução em tempo viável
- *small world*: num grafo em que temos várias comunidades fortemente conexas conectadas por pontes, nosso algoritmo foi capaz de identificar que em cada comunidade é necessário estabelecer um vértice que funcione como *hub* (Figura 22)

3.2.4 Comentários adicionais

No processo de iteração garantimos a finalização através da busca binária e sobre reinicializações dessas somente quando encontrado um conjunto W'', e consequentemente W' de menor tamanho. No pior caso, as amostragens não tem efeitos e o conjunto Z será igual a todos os vértices v do grafo, contudo nos resultados isso não ocorre, principalmente devido ao fator de convergência que será descrito a seguir e que complementa o algoritmo agora descrito.

Quanto a complexidade podemos assumir que a principal etapa será uma busca binária sobre o tamanho de G. Como etapa central da busca binária realiza-se Y amostras de tamanho J onde cada uma será checada para averiguar se é um fecho convexo de G, assim o tamanho médio das Y amostras será $2 * N$. Logo teremos $Y * ((2 * N) + M)$ para

uma rodada de busca binária. No máximo teremos N rodadas de buscas, logo teremos $N * Y * ((2 * N) + M) = N * Y * (N + M)$. De modo geral a complexidade final ficará por conta do Y mas, para $1 < Y <= N$, a solução ficará entre N^2 e N^4, ou seja, em tempo polinomial. Nota-se que paralelizar as amostragens também é uma solução muito boa pois a complexidade depende também de Y.

3.3 CONVERGÊNCIA APLICADA AO ALGORITMO PROPOSTO

A modelagem apresentada para esse algoritmo ainda não é a ideal. Embora a variável de amostragem Y introduzida melhore o tempo de busca do algoritmo, sendo relevante para o cálculo da sua complexidade, amostragens aleatórias independentes não nos fornecem uma boa solução ainda. O objetivo de fornecer antecipadamente a modelagem, antes de levantarmos a seguinte proposta é justamente fornecer de modo direto as etapas do algoritmo que se manterão mesmo após aplicada a indução e apresentar a complexidade do algoritmo baseada nessas etapas.

Durante os estudos da convexidade $P3$ e implementação do algoritmo percebemos que para classes de grafos que tem poucas soluções ótimas, como a classe de inflados ou grafos escada, a convergência era muito difícil. Podemos imaginar que se um grafo possui N vértices e precisamos escolher J deles, tal que $0 < J <= N$, e só exista um conjunto Z de tamanho J que seja o menor fecho convexo, escolher justamente esse conjunto será uma tarefa não propícia para poucas amostragens, ou seja, Y pequeno. E mais, será quase impossível escolher aleatoriamente este conjunto. Isso é verdade para a maioria dos casos, mas não se a escolha dos conjuntos for guiada de modo probabilístico.

Um algoritmo genético possui um fator importante de convergência, por exemplo. O crossover entre os melhores indivíduos de uma população tende a gerar indivíduos melhores ao longo das evoluções. Poderíamos ter aplicado esse tipo de algoritmo aqui mas a desvantagem é que por vezes o crossover considera a codificação/decodificação da solução, e nota-se que um crossover ingênuo entre dois fechos convexos de tamanho J nem sempre será um fecho convexo do grafo. Mas inspirada nessa possibilidade de convergência, que existe em outros tipos de algoritmo de otimização, propomos uma nova solução.

3.4 RESULTADOS

Os resultados e código do algoritmo proposto pode ser encontrado no repositório GitHub. Foram testados também diversas configurações para execução do mesmo algoritmo proposto. Essas configurações foram armazenadas em forma de variáveis ambiente no projeto e poderão ser alteradas conforme demonstrado no arquivo *README.md* do repositório. Por exemplo, é possível desabilitar a etapa de amostragens aleatórias por peso através da variável $WITH\_WEIGHT = False$. Outro ponto relevente é que os resultados

das execuções são salvos em formato CSV e para cada grafo testado é gerado um arquivo resultado no formato **.gexf* pronto para ser aberto no software open source Gephi.

As variáveis estão definidas no arquivo .env

- INITIAL_GRAPH: o número que representa o arquivo que contém o grafo. Por exemplo, se o arquivo tiver o nome 01.txt então esta variável deverá ter valor 01.
- CONTAMINANTS: o número de vizinhos que devem estar contaminados para contaminar um vértice no problema do fecho convexo.
- LENGTH_SAMPLE: o número de conjuntos aleatórios que deverão ser gerados para buscar uma melhor solução durante a fase de otimização.
- STOP_ON_FIRST_BEST_SAMPLE: se definido como True não precisará gerar LENGTH_SAMPLE conjuntos para encontrar o melhor. No primeiro conjunto que for melhor que a melhor solução atual o algoritmo de geração de conjuntos será interrompido para prosseguir para a próxima tentativa da fase de otimização.
- FLEXIBLE_BINARY_SEARCH: se definido como True irá realizar o reset do tamanho do conjunto mínimo para 1 sempre que encontrar uma melhor solução na busca binária. É equivalente a rodar uma nova busca binária sempre que encontrar uma nova solução melhor.
- WITH_WEIGHT: se definido como True fará a seleção do conjunto de vértices pertencente ao fecho inicial aleatório baseado em pesos.
- VELOCITY: multiplicador do peso de um vértice ao encontrar uma nova solução na fase de geração de conjuntos aleatórios.

Conforme relatado diversas configurações foram testadas, listamos abaixo as configurações utilizadas como default de execução:

- CONTAMINANTS = 2
- LENGTH_SAMPLE = 100 ou 500
- STOP_ON_FIRST_BEST_SAMPLE = False
- FLEXIBLE_BINARY_SEARCH = True
- WITH_WEIGHT = True
- VELOCITY = 1
- ONE_IN = 10 ou 100

grafo	classe	tamanho	tamanho de Z	Y	ONE_IN	melhor execução?
001.tgf	caminho	100	57	500	10	Sim
001.tgf	caminho	100	57	500	100	Não
001.tgf	caminho	100	58	100	10	Não
001.tgf	caminho	100	61	100	100	Não
002.tgf	caminho	1000	569	500	100	Sim
002.tgf	caminho	1000	575	100	100	Não
002.tgf	caminho	1000	575	500	10	Não
002.tgf	caminho	1000	599	100	10	Não
003.tgf	árvore	100	52	500	10	Sim
003.tgf	árvore	100	53	100	10	Não
003.tgf	árvore	100	54	500	100	Não
003.tgf	árvore	100	55	100	100	Não
004.tgf	árvore	1000	536	500	10	Sim
004.tgf	árvore	1000	538	500	100	Não
004.tgf	árvore	1000	546	100	100	Não
004.tgf	árvore	1000	551	100	10	Não
005.tgf	grade	100	10	100	10	Sim
005.tgf	grade	100	11	100	100	Não
005.tgf	grade	100	11	500	100	Não
005.tgf	grade	100	12	500	10	Não
006.tgf	grade	900	32	500	100	Sim
006.tgf	grade	900	33	100	10	Não
006.tgf	grade	900	33	500	10	Não
006.tgf	grade	900	34	100	100	Não
007.tgf	inflado	128	32	100	10	Sim
007.tgf	inflado	128	32	100	100	Não
007.tgf	inflado	128	32	500	10	Não
007.tgf	inflado	128	32	500	100	Não
008.tgf	inflado	1024	252	500	10	Sim
008.tgf	inflado	1024	252	500	100	Não
008.tgf	inflado	1024	254	100	100	Não
008.tgf	inflado	1024	255	100	10	Não
009.tgf	ciclo	1000	562	500	100	Sim
009.tgf	ciclo	1000	568	500	10	Não
009.tgf	ciclo	1000	578	100	100	Não
009.tgf	ciclo	1000	591	100	10	Não
010.tgf	ciclo	100	55	100	10	Sim
010.tgf	ciclo	100	56	500	100	Não
010.tgf	ciclo	100	58	100	100	Não
010.tgf	ciclo	100	76	500	10	Não
011.tgf	cordal	4917	1216	500	100	Sim
011.tgf	cordal	4917	1244	500	10	Não

A seguir mostramos a tabela de resultados da execuções do algoritmo para diversas classes de grafos e utilizando diversas configurações:

grafo	classe	tamanho	tamanho de Z	Y	ONE_IN	melhor execução?
011.tgf	cordal	4917	1248	100	100	Não
011.tgf	cordal	4917	1336	100	10	Não
012.tgf	grade	3600	64	500	100	Sim
012.tgf	grade	3600	65	500	10	Não
012.tgf	grade	3600	66	100	100	Não
012.tgf	grade	3600	68	100	10	Não
013.tgf	inflado	1092	364	500	10	Sim
013.tgf	inflado	1092	364	500	100	Não
013.tgf	inflado	1092	366	100	10	Não
013.tgf	inflado	1092	366	100	100	Não
014.tgf	indefinida	20000	1927	100	10	Sim
015.tgf	árvore	1999	1111	100	10	Sim
015.tgf	árvore	1999	1137	100	100	Não
016.tgf	árvore	3999	2200	100	100	Sim
016.tgf	árvore	3999	2258	100	10	Não
018.tgf	árvore	9999	5548	100	100	Sim
018.tgf	árvore	9999	6520	100	10	Não
019.tgf	M(200,3)	400	108	100	100	Sim
019.tgf	M(200,3)	400	113	100	10	Não
020.tgf	M(200,4)	400	82	100	10	Sim
020.tgf	M(200,4)	400	84	100	100	Não
021.tgf	M(200,5)	400	65	100	10	Sim
021.tgf	M(200,5)	400	67	100	100	Não
022.tgf	M(200,6)	400	57	100	100	Sim
022.tgf	M(200,6)	400	58	100	10	Não
023.tgf	grade	3364	127	100	100	Sim
023.tgf	grade	3364	130	100	10	Não
024.tgf	3-regular	1000	275	100	10	Sim
024.tgf	3-regular	1000	275	100	100	Não
025.tgf	indefinida	1000	83	100	10	Sim
025.tgf	indefinida	1000	85	100	100	Não
026.tgf	4-regular	1000	32	100	100	Sim
026.tgf	4-regular	1000	36	100	10	Não
027.tgf	5-regular	1000	18	100	100	Sim
027.tgf	5-regular	1000	19	100	10	Não
028.tgf	6-regular	1000	10	100	100	Sim
028.tgf	6-regular	1000	12	100	10	Não
029.tgf	7-regular	1000	7	100	10	Sim
029.tgf	7-regular	1000	9	100	100	Não
030.tgf	8-regular	1000	8	100	10	Sim
030.tgf	8-regular	1000	9	100	100	Não
031.tgf	caminho	10000	5823	100	100	Sim
031.tgf	caminho	10000	8484	100	10	Não

grafo	classe	tamanho	tamanho de Z	Y	ONE_IN	melhor execução?
035.tgf	indefinida	10000	5894	100	100	Sim
035.tgf	indefinida	10000	7146	100	10	Não
050.tgf	inflado	210	15	100	10	Sim
050.tgf	inflado	210	15	100	100	Não
051.tgf	inflado	2450	50	100	100	Sim
051.tgf	inflado	2450	51	100	10	Não
052.tgf	inflado	480	160	100	10	Sim
052.tgf	inflado	480	160	100	100	Não
053.tgf	inflado	800	200	100	10	Sim
053.tgf	inflado	800	202	100	100	Não
055.tgf	inflado	1500	507	100	100	Sim
055.tgf	inflado	1500	510	100	10	Não
056.tgf	inflado	240	26	100	10	Sim
056.tgf	inflado	240	26	100	100	Não
058.tgf	inflado	3000	1010	100	100	Sim
058.tgf	inflado	3000	1037	100	10	Não

Para os resultados onde o tamanho da amostra Y verificada foi verificado tanto para 500 quanto para 100 amostras, que são as amostras de 001 até 013, verificamos que quanto maior o tamanho da amostra maiores as chances de encontrarmos o um fecho convexo de tamanho menor. Desses 13 itens, 10 dos melhores resultados foram obtidos com tamanho de amostra 500 enquanto somente 3 dos resultados foram obtidos com tamanho de amostra 100. Há assim uma tendência muito clara de que quanto mais amostragens maiores as probabilidades de obtermos o P_3-*hull set*.

É importante notar que para grafos com um alto número de vértices, quanto maior o número de amostragens Y melhor o fecho convexo encontrado ao final. Porém, como auferido, a complexidade do algoritmo depende diretamente do número de amostragens, embora num cenário real as amostragens e verificação se são fecho convexo de G entre uma iteração de busca binária e outra poderiam ser paralelizadas. Esta consideração também é um avanço neste trabalho.

Além disso, dando foco as 10 melhores amostras de tamanho 500, verificamos que o fator de probabilidade (ONE_IN) numa amostragem sem reposição e por pesos em 5 dessas amostras foi de 10% enquanto nas outras 5 amostras foi de 1% o que pode indicar que a escolha de um fator de probabilidade para chegar-se ao P_3-*hull set* pode também ser aleatório. Se considerarmos também todas as 38 execuções, filtrando somente o melhor resultado de cada uma, e colocarmos em evidência o fator de probabilidade para amostragem, encontramos que 20 delas foram com 10% e outras 18 foram com 1% o que parece ainda indicar que o escolha do fator de probabilidade inicial pode ser aleatória. Isto indica que a técnica de amostragem sem reposição por pesos pode ser utilizada em contextos mais amplos onde se preza pela escolha de um subconjunto desconhecido em que pretende-se convergir para o ótimo sem precisarmos antecipadamente escolher qual

a probabilidade deste subconjunto ser a solução atual. Precisa-se de mais estudos para concluir a respeito disso, mas parece corroborar também a ideia empírica de que uma boa solução pode conter partes de uma solução ótima.

Utilizar o ONE_IN permite o trade-off entre os conceitos de *exploration* e *exploitation* pois quanto menor for esse valor, maior a probabilidade do resultado da busca ser considerado o ótimo pois haverá menor chance de exploração entre as amostragens. Já quanto maior esse valor, maiores as possibilidades de exploração por uma nova solução enquanto é pouco assumido que o resultado de busca se trata de um ótimo. O conceito de exploration e exploitation pode ser melhor entendido em (EFRAIMIDIS; SPIRAKIS, 2006). Este artifício técnico de amostragens por pesos sem repetição para obtenções de subconjuntos na busca pela manutenção de um propriedade, neste caso a convexidade em grafos, não parece ter sido utilizada antes e pode ser objeto de estudos posteriores.

Vejamos também nas execuções apresentadas na tabela acima, para classes de grafos onde o tamanho do P_3-*hull set* é conhecido:

grafo	classe	solução ótima	tamanho de Z	Aproximação
001.tgf	caminho	50	57	1,1
002.tgf	caminho	500	569	1,1
003.tgf	árvore	50	52	1,0
004.tgf	árvore	500	536	1,1
005.tgf	grade	10	10	1,0
006.tgf	grade	30	32	1,1
007.tgf	inflado	32	32	1,0
009.tgf	ciclo	500	562	1,1
010.tgf	ciclo	50	55	1,1
012.tgf	grade	60	64	1,1
013.tgf	inflado	364	364	1,0
015.tgf	árvore	1036	1111	1,1
016.tgf	árvore	2074	2200	1,1
018.tgf	árvore	5178	5548	1,1
023.tgf	grade	120	127	1,1
050.tgf	inflado	15	15	1,0
051.tgf	inflado	50	50	1,0
053.tgf	inflado	200	200	1,0
055.tgf	inflado	500	507	1,0
056.tgf	inflado	26	26	1,0
058.tgf	inflado	1000	1010	1,0

Como mostrado na tabela acima o algoritmo é em diversas soluções 1,1 aproximativo em suas melhores soluções encontradas. Isto é um ótimo resultado pois demonstra que para diferentes classes de grafos seria possível aplicá-lo sem necessidade de encontrar uma heurística eficiente. Para redes reais, onde se avalia contaminação ou influência, este resultado sugere uma nova abordagem técnica para o tratamento prático da convexidade $P3$.

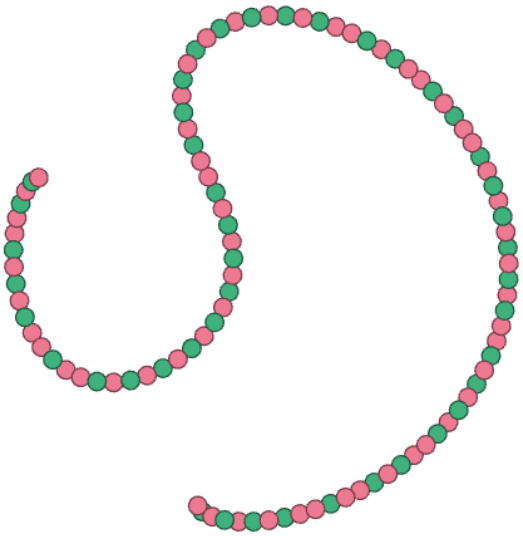

Figura 17 – Exemplo de uma solução encontrada para o grafo caminho 001.tgf. Fecho inicial em verde.

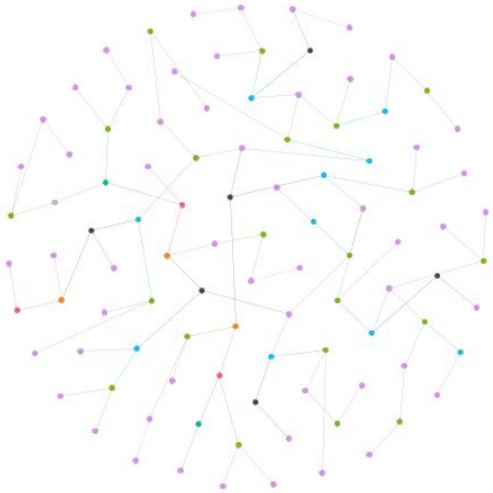

Figura 18 – Exemplo de uma solução encontrada para o grafo árvore 003.tgf. Fecho inicial em rosa.

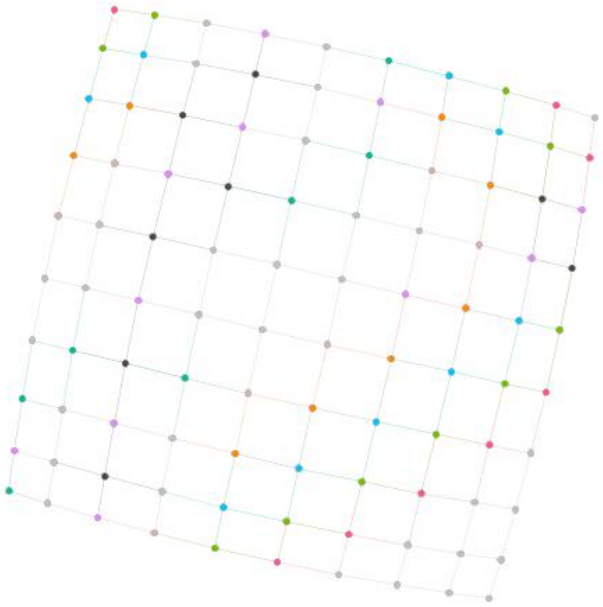

Figura 19 – Exemplo de uma solução encontrada para o grafo árvore 005.tgf. Fecho inicial em rosa.

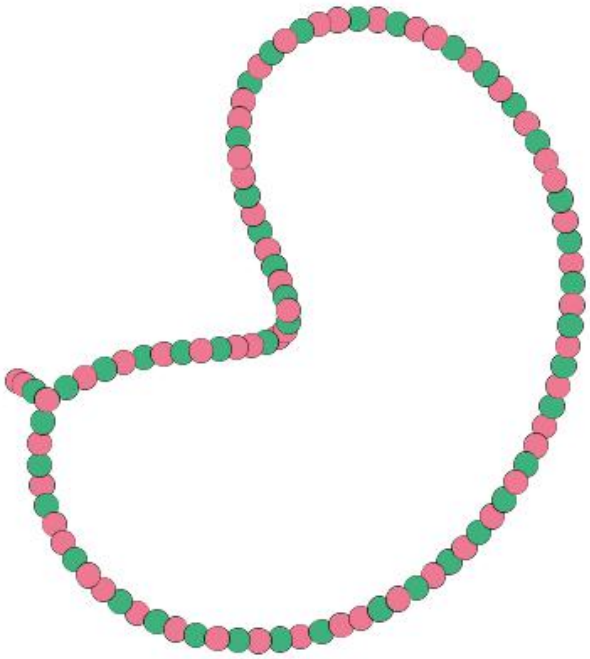

Figura 20 – Exemplo de uma solução encontrada para o grafo ciclo 010.tgf. Fecho inicial em verde.

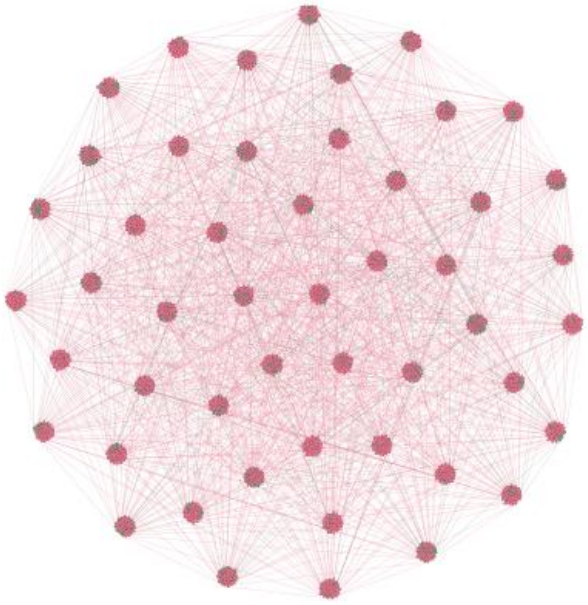

Figura 21 – Exemplo de uma solução encontrada para o grafo inflado 051.tgf. Fecho inicial em verde.

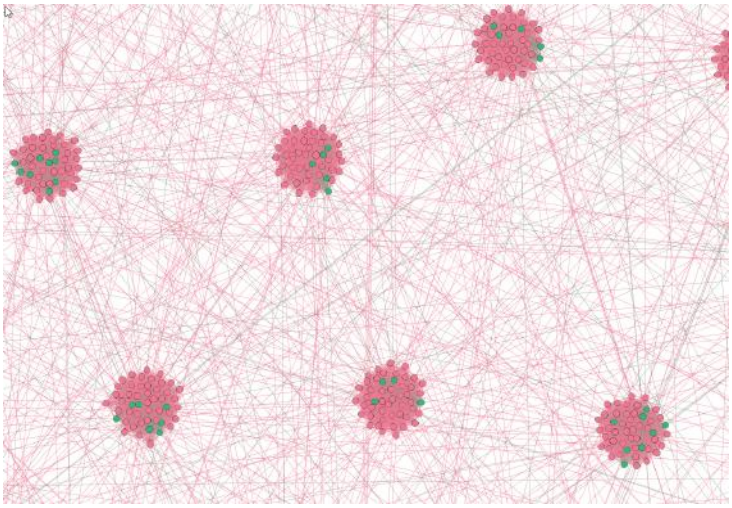

Figura 22 – Zoom in numa solução encontrada para o grafo inflado 051.tgf. Fecho inicial em verde.

4 CONCLUSÃO E TRABALHOS FUTUROS

Concluímos este trabalho com a esperança de utilização destes resultados nas diversas áreas de aplicabilidade deste estudo como biologia, web, sistemas de recomendação, entre outras. Em todas essas redes temos propagação epidêmica, seja de vírus ou ideias.

Sabemos que o problema do número envoltório P3 é NP-completo. Logo, o algoritmo produzido é aproximativo, utilizando a busca binária e escolhas probabilísticas. Esse algoritmo pode ser usado para calcular a envoltória convexa em qualquer classe de grafos. Nos resultados que tivemos, conseguimos empiricamente soluções ótimas para os grafos analisados na classe dos grafos inflados, grades e cordais e aproximações para demais classes de grafos.

4.1 TRABALHOS FUTUROS

Em trabalhos futuros podemos implantar uma rede distribuída para execução e verificação das amostragens checando na prática o tempo de execução para obter-se uma boa solução. Outro ponto interessante seria encontrar, por propriedades das classes de grafos, o número de amostras Y ideal para executarmos o algoritmo proposto. O processo mais evidente é que para grafos muito densos o tempo de contaminação deverá ser menor e assim o tamanho do conjunto de vértices no menor fecho convexo tende a ser menor. O número de amostragens nesse caso também tenderia a ser menor, mas não sabemos ainda o quanto. O fator probabilístico de amostragem por pesos e sem repetição também é uma incógnita deste problema. Testamos um número reduzido para permitir maiores explorações mas também permitir a chance de melhores soluções serem buscadas mas pode-se tentar induzir esse fator pelas propriedades de um grafo averiguadas em $O(N + M)$.

REFERÊNCIAS

BONDY, A.; MURTY, U. S. R. Graph theory. Springer London, 2011. Disponível em: <https://books.google.com.br/books?id=HuDFMwZOwcsC>.

CENTENO, C. C. **A convexidade P_3 para grafos não direcionados**. Tese (Doutorado) — UFRJ, 2012.

CENTENO, C. C. **Sobre convexidade P3**. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2012.

CENTENO, C. C. et al. Irreversible conversion of graphs. **Theoretical Computer Science**, p. 3693 – 3700, 2011.

CHARTRAND, G.; WALL, C. E.; ZHANG, P. The convexity number of a graph. **Graphs and Combinatorics**, v. 18, n. 2, p. 209–217, May 2002. Disponível em: <https://doi.org/10.1007/s003730200014>.

CHEN, N. On the approximability of influence in social networks. **SIAM Journal on Discrete Mathematics**, SIAM, v. 23, n. 3, p. 1400–1415, 0 2009.

Cordasco, G.; Gargano, L.; Rescigno, A. A. On Finding Small Sets that Influence Large Networks. **arXiv e-prints**, p. arXiv:1610.04838, out. 2016.

DOMINGOS, P. M.; RICHARDSON, M. Mining the network value of customers. In: **KDD '01**. [S.l.: s.n.], 2001.

DOURADO, M. C. et al. On the computation of the hull number of a graph. **Discrete Mathematics**, Elsevier, v. 309, n. 18, p. 5668–5674, 2009.

DOURADO, M. C. et al. On the convexity number of graphs. **Graphs and Combinatorics**, v. 28, n. 3, p. 333–345, May 2012. ISSN 1435-5914. Disponível em: <https://doi.org/10.1007/s00373-011-1049-7>.

DOURADO, M. C.; PROTTI, F.; SZWARCFITER, J. L. Complexity results related to monophonic convexity. **Discrete Applied Mathematics**, v. 158, n. 12, p. 1268 – 1274, 2010. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166218X0900479X>.

DREYER, P. A.; ROBERTS, F. S. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. **Discrete Applied Mathematics**, v. 157, n. 7, p. 1615–1627, 2009. ISSN 0166-218X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0166218X08004393>.

DUNBAR, J. E.; HAYNES, T. W. Domination in inflated graphs. **Congressus Numerantium**, UTILITAS MATHEMATICA PUBLISHING INC, p. 143–154, 1996.

EFRAIMIDIS, P. S.; SPIRAKIS, P. G. Weighted random sampling with a reservoir. **Information Processing Letters**, v. 97, n. 5, p. 181–185, 2006. ISSN 0020-0190. Disponível em: <https://www.sciencedirect.com/science/article/pii/S002001900500298X>.

ERDöS, P. et al. Some remarks on simple tournaments. **Algebra Universalis**, Springer, v. 2, n. 1, p. 238–245, 1972.

ESKANDANIAN, F.; SONBOLI, N.; MOBASHER, B. Power of the few: Analyzing the impact of influential users in collaborative recommender systems. In: . [S.l.: s.n.], 2019. p. 225–233.

FAVARON, O. Irredundance in inflated graphs. **Journal of Graph Theory**, Wiley Online Library, v. 28, n. 2, p. 97–104, 1998.

GONG, M. et al. An efficient shortest path approach for social networks based on community structure. **CAAI Transactions on Intelligence Technology**, v. 1, n. 1, p. 114–123, 2016. ISSN 2468-2322. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2468232216000123>.

GRAVIER, S.; KOVSE, M.; PARREAU, A. Generalized sierpinski graphs. In: . Posters at EuroComb, 2011. Disponível em: <http://www.renyi.hu/conferences/ec11/posters/parreau.pdf>.

HAFIENE, N.; KAROUI, W.; Ben Romdhane, L. Influential nodes detection in dynamic social networks: A survey. **Expert Systems with Applications**, v. 159, p. 113642, 2020. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420304668>.

JIANG, L. et al. An efficient algorithm for mining a set of influential spreaders in complex networks. **Physica A: Statistical Mechanics and its Applications**, v. 516, p. 58–65, 2019. ISSN 0378-4371. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0378437118313499>.

KANG, L.; SOHN, M. Y.; CHENG, T. E. Paired-domination in inflated graphs. **Theoretical computer science**, Elsevier, v. 320, n. 2-3, p. 485–494, 2004.

KEMPE, D.; KLEINBERG, J.; TARDOS, É. Maximizing the spread of influence through a social network. In: ACM. **Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2003. p. 137–146.

KEMPE, D.; KLEINBERG, J.; TARDOS, É. Influential nodes in a diffusion model for social networks. In: SPRINGER. **International Colloquium on Automata, Languages, and Programming**. [S.l.], 2005. p. 1127–1138.

KEMPE, D.; KLEINBERG, J.; TARDOS, E. Maximizing the spread of influence through a social network. **Theory of Computing**, Theory of Computing, v. 11, n. 4, p. 105–147, 2015. Disponível em: <https://theoryofcomputing.org/articles/v011a004>.

KIM, Y.; SRIVASTAVA, J. Impact of social influence in e-commerce decision making. In: . [S.l.: s.n.], 2007. v. 258, p. 293–302.

LEVI, F. W. On helly's theorem and the axioms of convexity. **The Journal of the Indian Mathematical Society**, v. 15, p. 65–76, 1951.

MOON, J. Embedding tournaments in simple tournaments. **Discrete Mathematics**, v. 2, n. 4, p. 389 – 395, 1972. ISSN 0012-365X. Disponível em: <http://www.sciencedirect.com/science/article/pii/0012365X72900167>.

MOSSEL, E.; ROCH, S. On the submodularity of influence in social networks. In: ACM. **Proceedings of the thirty-ninth annual ACM symposium on Theory of computing**. [S.l.], 2007. p. 128–134.

NASCIMENTO, J. R.; FERREIRA, D. J.; COELHO, E. M. M. Número envoltório na convexidade p3: Resultados e aplicações. **Revista de Sistema e Computação - RSC**, v. 9, n. 7, p. 238–244, 2020.

OLIVARES, R.; MUñOZ, F.; RIQUELME, F. A multi-objective linear threshold influence spread model solved by swarm intelligence-based methods. **Knowledge-Based Systems**, v. 212, p. 106623, 2021. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705120307528>.

PALUGA, E. M.; CANOY, J. S. R. Monophonic numbers of the join and composition of connected graphs. **Discrete Mathematics**, v. 307, n. 9, p. 1146 – 1154, 2007. ISSN 0012-365X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0012365X0600625X>.

PASTOR-SATORRAS, R.; VESPIGNANI, A. et al. Epidemics and immunization in scale-free networks. **Handbook of Graphs and Networks, Wiley-VCH, Berlin**, Wiley Online Library, 0 2003.

PELAYO, I. M. **"Geodesic Convexity in Graphs"**. [S.l.]: Springer New York, 2013. (SpringerBriefs in Mathematics).

PONCIANO, V. et al. Sistema de recomendação em grafos utilizando influência coletiva. In: **Anais do XLIX Seminário Integrado de Software e Hardware**. Porto Alegre, RS, Brasil: SBC, 2022. p. 200–205. ISSN 2595-6205. Disponível em: <https://sol.sbc.org.br/index.php/semish/article/view/20810>.

PONCIANO, V. dos S. **Estudo sobre convexidade em grafos**. Tese (Doutorado) — Tese de Doutorado-Programa de Pós-Graduação em Informática-UFRJ., 2019.

RAMOS, I. d. F.; SANTOS, V. F. d.; SZWARCFITER, J. L. Complexity aspects of the computation of the rank of a graph. **Discrete Mathematics and Theoretical Computer Science**, DMTCS, Vol. 16 no. 2 (in progress), n. 2, p. 73–86, set. 2014. Special issue PRIMA 2013. Disponível em: <https://hal.inria.fr/hal-01185615>.

RODRIGUES, H.; FONSECA, M. Viral marketing as epidemiological model. In: . [S.l.: s.n.], 2015.

RUFINO, V. et al. Contaminação epidêmica em redes: Imunidade coletiva e suas implicações frente a atacantes estratégicos. In: SBC. **17º Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPerformance 2018)**. [S.l.], 2018. v. 17, n. 1/2018.

RUFINO, V. et al. Dilemas frente epidemias estratégicas: Vacinar ou reiniciar? In: SBC. **Anais do XVIII Workshop em Desempenho de Sistemas Computacionais e de Comunicação**. [S.l.], 2019.

SALES GUILHERME DA COSTA, F. D. R. . I. G. Majority vote community detection with dynamic threshold and bootstrapped rounds. 2019. Disponível em: <https://pantheon.ufrj.br/handle/11422/14067>.

SZWARCFITER, J. L. **Grafos e algoritmos computacionais**. Brazil: Rio de Janeiro : Campus, 1988. ISBN 8570013418 (broch.).

SZWARCFITER, J. L. **Teoria computacional de grafos: os Algoritmos**. [S.l.]: GEN LTC, 2018. (Computação, Informática e Mídias Digitais).

VINSON, D. et al. Consumer and industrial buying behavior. **Journal of Marketing**, v. 42, p. 135, 07 1978.

WEST, D. **Introduction to Graph Theory**. Pearson, 2017. (Math Classics). Disponível em: <https://books.google.com.br/books?id=HuDFMwZOwcsC>.

ZHU, Z. Discovering the influential users oriented to viral marketing based on online social networks. **Physica A: Statistical Mechanics and its Applications**, v. 392, n. 16, p. 3459–3469, 2013. ISSN 0378-4371. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0378437113002689>.

APÊNDICES

APÊNDICE A – MARKETING VIRAL USANDO CONVEXIDADE

"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original".

Albert Einstein

A.1 MARKETING VIRAL E SISTEMA DE RECOMENDAÇÃO

Com o crescimento das redes sociais, temos os chamados influenciadores que são os usuários da rede capazes de emitirem informações ou ideias pela rede social e essas informações serem capaz de estimular outros usuário da rede que a receberam a compartilhá-las com seu amigos, temos um marketing viral. Segundo (RODRIGUES; FONSECA, 2015) Marketing Viral é uma estratégia de marketing que tem como objetivo explorar as conexões entre os indivíduos da rede para se espalhar e viralizar como ocorre em uma epidemia.

Durante as últimas décadas, com o surgimento do Youtube, Amazon, Netflix e muitos outros serviços da web, os sistemas de recomendação tomaram cada vez mais lugar em nossas vidas. Desde o e-commerce (sugerir aos compradores artigos que lhes possam interessar) à publicidade online (sugerir aos utilizadores os conteúdos certos, de acordo com as suas preferências), os sistemas de recomendação são hoje incontornáveis nas nossas jornadas online diárias. O objetivo das empresas é oferecer melhores direcionamentos de serviços e produtos com intuito de transformar o usuário da rede em cliente consumidor. Esses sistemas são usados para prever a classificação ou preferência que um usuário sinaliza a um item pesquisado na rede e, assim, recomendar produtos e/ou serviços com maior probabilidade de aceitação pelo usuário.

O objetivo neste capítulo é propor um modelo de recomendação representando a rede de usuários por um grafo e usando a convexidade P_3 através de conceitos conhecidos de convexidade em grafos.

A.1.1 Trabalhos Relacionados

No artigo (RODRIGUES; FONSECA, 2015), é apresentado um modelo epidemiológico SIR (Susceptible-Infected-Recovered) para estudar os efeitos de uma estratégia de marketing viral. É feita uma comparação entre os parâmetros da doença e o aplicativo de marketing, e são realizadas simulações utilizando o software Matlab.

Já em (HAFIENE; KAROUI; Ben Romdhane, 2020) são encontrados pontos chaves e desafios para o problema de maximização de influência e detecção de nós influentes

em redes sociais dinâmicas em contraponto a pesquisas anteriores que se concentram no desenvolvimento de algoritmos em redes sociais estáticas.

No texto de (ESKANDANIAN; SONBOLI; MOBASHER, 2019) uma pequena quantidade de usuários denotados como usuários influentes possuem grande relevância para o marketing viral, uma vez que tais usuários têm impacto sobre os outros usuários. Essa dinâmica pode afetar o comportamento de uma rede sendo esse um dos principais problemas de interesse da área.

Em (DREYER; ROBERTS, 2009) foram estudados modelos de disseminação de doenças ou opiniões por meio de redes sociais em grafos. Nos modelos citado neste artigo, os vértices possuem dois estados possíveis: 1 para infectado ou 0 para não infectado. A mudança de um estado para o outro é chamado de processo de limiar irreversível, onde um vértice entra no estado 1 se pelo menos seus vizinhos estiverem no estado 1. Além disso, um determinado vértice nunca sai do estado 1 uma vez que esteja nele. O objetivo deste estudo em questão é que todos os vértices da população sejam alcançados. Foram discutidas as maneiras de defesa contra tais conjuntos saturantes, como por exemplo, vacinando ou projetando topologias de rede. Foi resolvido o problema para caminhos, grafos completos, árvores e limites para grafos r-regulares.

Em (NASCIMENTO; FERREIRA; COELHO, 2020) apresenta uma revisão sistemática da literatura sobre os resultados e aplicações ao número envoltório na convexidade P_3 em grafos sobre estudos teóricos e aplicados sobre o número envoltório considerando a modelagem de fenômenos sociais. Os resultados da pesquisa mostram que o parâmetro é pouco explorado para a modelagem de fenômenos em redes sociais.

Em um estudo dado por (CHEN, 2009) uma rede social foi modelada pela estrutura de um grafo de modo que para cada usuário, representado por um vértice, foi fixado um valor denotado por *threshold*. Com isso, cada usuário adotará ou comprará um determinado produto comercializado na rede se tal vértice possuir um número de vizinhos igual ao valor do parâmetro *threshold*. Este estudo teve como objetivo encontrar um pequeno conjunto de vértices (usuários) S tal que, se o produto a ser vendido for consumido pelos usuários de S, então uma grande parte de outros usuários tenderá a se tornar comprador ou consumidor do produto em questão. O principal resultado desse estudo foi obter o menor conjunto S que tem a propriedade de influenciar o consumo demasiado no grafo sendo esta uma tarefa difícil do ponto de vista computacional. E, para este caso tentar uma aproximação de uma solução minimizada de S. Alguns outros resultados sobre influência em redes sociais podem ser encontrados nos trabalhos (KEMPE; KLEINBERG; TARDOS, 2003), (KEMPE; KLEINBERG; TARDOS, 2005) e (PASTOR-SATORRAS; VESPIGNANI et al., 2003).

Observa-se em (GONG et al., 2016) a importância do estudos de caminhos mínimos em redes sociais de larga escala. Segundo (KIM; SRIVASTAVA, 2007) as decisões de compra são muitas vezes fortemente influenciadas por pessoas que o consumidor conhece

e confia. Além disso, muitos compradores online tendem a esperar pelas opiniões dos primeiros usuários antes de tomar uma decisão de compra para reduzir o risco de comprar um produto fora das expectativas de consumo.

A.2 DEFINIÇÕES BÁSICAS

Chamamos de *iteração* do operador de influência I sobre um subconjunto de vértices S ao conjunto $I(I(...I(S))) = I^k(S)$. Esses e outros conceitos sobre grafos podem ser encontrados em (BONDY; MURTY, 2011).

Seja S um subconjunto de vértices de um grafo G, suponhamos que S seja um grupo de influenciadores do grafo G. Partindo do conjunto S, iremos infectar o grafo através de sucessivas iterações de I passo a passo até obtermos um conjunto que não infecta mais nenhum outro vértice e esse conjunto é o fecho convexo de S, denotado por $H(S)$.

$$I : I^0(S) \subset I^1(S) \subset I^2(S) \subset ... \subset I^{j+1}(S) = H(S)$$

Caso $H(S) = V(G)$, ou seja, o conjunto S infecta todos os vértices do grafo em um número finito de passos de infecção, dizemos que S é um conjunto de fecho do grafo (em inglês, P_3-*hull set*). Alguns estudos de convexidade podem ser encontrados em (DOURADO et al., 2009) e (NASCIMENTO; FERREIRA; COELHO, 2020) .

A.3 O PROBLEMA DE MAXIMIZAÇÃO DE INFLUÊNCIA E BIBLIOGRAFIAS RELACIONADAS

O termo marketing de influência foi primeiramente utilizado no texto "Consumer and Industrial Buying Behavior"(VINSON et al., 1978). Marketing de influência é uma abordagem de marketing que consiste em praticar ações focadas em indivíduos que exerçam influência ou liderança sobre potenciais clientes de uma marca. Como benefício, os influenciadores interferem nas decisões de compra dos clientes a favor de uma determinada marca.

O objetivo do marketing viral na plataforma de redes sociais populares online é propagar rapidamente informações de marketing a um custo menor e aumentar as vendas. No artigo "Discovering the influential users oriented to viral marketing based on online social networks"(ZHU, 2013) os autores propõem um método para ajudar as empresas a identificar esses usuários como sementes para maximizar a difusão de informações no marketing viral.

Nas redes sociais em grande escala, o comportamento coletivo de grandes populações pode ser influenciado por um pequeno número de usuários (JIANG et al., 2019). Tais usuários são denominados *better influencers*. A identificação dos *better influencers* ajuda a controlar uma rede inteira ou uma grande parte da rede. O problema definido como

encontrar *better influencers* em uma rede é formalmente definido como a maximização da influência coletiva (PONCIANO, 2019).

De modo a sugerir meios eficientes de tratar o problema, iremos abordar alguns modelos encontrados na literatura assim como introduzir uma nova proposta de modelo computacional.

Domingos e Richardson propuseram um problema algorítmico fundamental para a identificação dos *better influencers* em (DOMINGOS; RICHARDSON, 2001). Suponha que um conjunto de dados em uma rede social tenha as estimativas da extensão em que os usuários influenciam uns aos outros, e ao se recomendar um novo produto o esperado seja que o produto seja adotado por grande fração da rede. É mencionado que princípio do marketing viral é que, ao visar inicialmente os *better influencers* da rede e, dando amostras grátis do produto, é possível desencadear uma cascata de influência pela qual os usuários ligados a ele recomende o produto a outros usuários fazendo com que diversos usuários experimentem o produto. A grande questão se concentra no fato da escolha adequada de usuários que entre neste processo da maneira mais eficiente.

Neste trabalho (OLIVARES; MUñOZ; RIQUELME, 2021),foi proposto um novo modelo de otimização para ambas as perspectivas em conflito, através do modelo.

Segundo o modelo apresentado em conhecido na literatura como *Linear Threshold Model*, cada vértice v do grafo é influenciado por cada um de seus vizinhos de acordo com peso $p_{v,w}$ de modo que a soma desses pesos seja igual a 1. Segundo o autor, a dinâmica de contaminação acontece da seguinte maneira: Cada vértice v escolhe um *threshold* (limite) θ_v de maneira uniformemente aleatória em um intervalo $[0, 1]$; isso representa o números de vizinhos de v que devem se tornar ativos para que v se torne ativo. Dada uma escolha aleatória de limites e um conjunto inicial de vértice ativos I_0, as interação $I(I(...I(S))) = I^t(S)$ são realizada em t etapas, onde todos os vértices que estavam ativos na tempo $t-1$ permanecem ativos e será ativado um vértice v para o qual o peso total de seus vizinhos ativos seja pelo menos θ_v.

Outro modelo que podemos citar que é apresentado em (KEMPE; KLEINBERG; TARDOS, 2003) é o modelo *Cascata Independente* que se inicia com um conjunto inicial de vértices ativos I_0, quando um vértice v se torna ativo pela primeira vez na etapa t, é dada uma única chance de ativar cada vizinho w dele que é inativo. O vértice v terá sucesso com uma probabilidade $p_{v,w}$ que é o parâmetro do sistema. Se o vértice v for bem-sucedido, então w se tornará ativo na etapa $t+1$; caso contrário, não poderá fazer outra tentativa de ativar o vértice w nas etapas posteriores.

A.4 UM EXEMPLO NO CONJUNTO DE *BETTER INFLUENCERS* RESTRITO A GRAFOS INFLADOS

Seja $G(V, E)$ um grafo. O *grafo inflado* G_I de G é obtido por G trocando todo vértice v de grau $d(v)$ por uma clique $K_{d(v)}$ e cada aresta uv por uma aresta entre dois vértices da clique correspondente $K_{d(u)}$ e $K_{d(v)}$ de G_I de tal forma que G_I que vem das arestas de G forma um emparelhamento de G_I. Os grafos inflados são introduzidos por (DUNBAR; HAYNES, 1996). O estudo de dominação e de outras variações do problema de dominação pode ser observado em (FAVARON, 1998; KANG; SOHN; CHENG, 2004). Seja $G = (V, E)$ um grafo simples. O conjunto $S \subseteq V(G)$ é uma dominação se para cada vértice $v \in V - S$. O vértice v tem pelo menos vizinho em S.

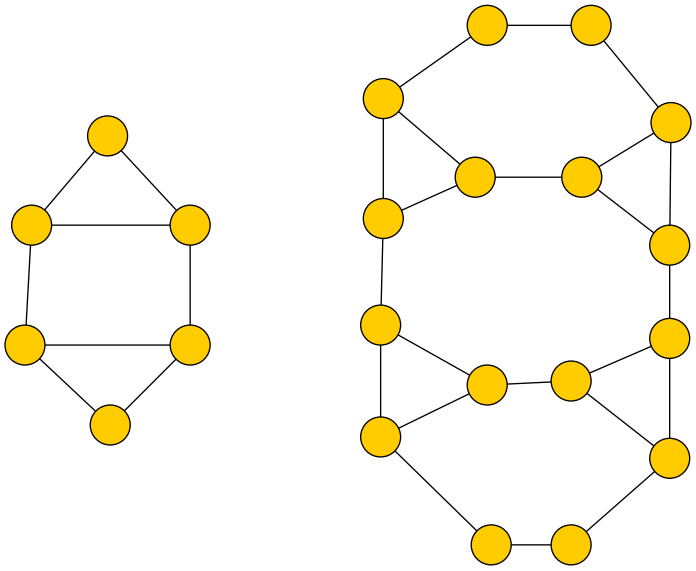

Figura 23 – Ilustração da contaminação G e seu grafo inflado G_I

O conjunto $S \subseteq V(G)$ é uma *dupla dominação* de G se para cada vértice $v \in V(G) - S$, $|S \cap N_G(v)| \geq 2$, ou seja, cada vértice v no grafo tem pelo menos dois vizinhos em S. O cálculo da dupla dominação mínima foi provado em (CENTENO, 2012a) que para grafos bipartidos, cordais, bipartidos cordais, a complexidade do problema é NP-completo. Note que o conjunto de *better influencers* para $t = 2$ é também uma dupla dominação.

Teorema 1. *(PONCIANO et al., 2022) O conjunto de better influencers para $t = 2$ (uma dupla dominação) é NP-completo para grafos inflados.*

Teorema 2. *(PONCIANO et al., 2022) Seja G_I um grafo inflado. O conjunto dos usuários considerados better influencers de G_I pode ser computado em tempo polinomial quando não temos as restrições para tempo t.*

A.5 DEMONSTRAÇÃO DO TEMPO DE GERAÇÃO POLINOMIAL PARA T=3

Provamos que podemos encontrar os *better influencers* de maneira polinomial para t=3 em uma classe restrita de inflados que mostram que é polinomial para classe de grafos de Sierpinski $S(n,k)$, conforme definido em (GRAVIER; KOVSE; PARREAU, 2011). Em particular, para $S(n,3)$ são grafos de Torre de Hanói.

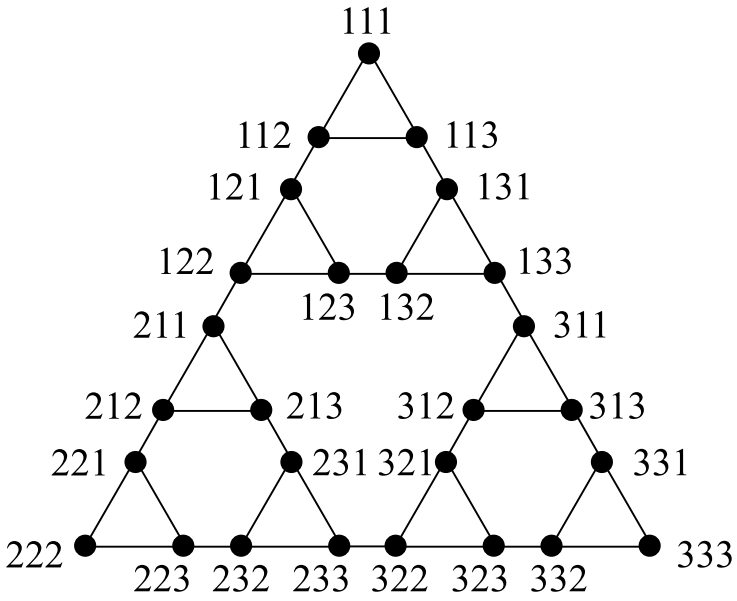

Figura 24 – Exemplo de $S(3, K_3)$.

Um subgrafo gerador S de um grafo G é chamado de $k-fator$ de G, se $d(u) = k$, $\forall u \in S$. No caso, particular de $k = 2$, S é a uma união disjunta de ciclos, chamado 2-fator.

Teorema 3. *Se G tem 2-fator, então G_I tem conjunto de better influencers de tamanho mínimo n com tempo de geração igual t=3.*

Demonstração. Suponha que raiz G de grafo inflado G_I tenha um 2-fator , então grafo G possui um subgrafo gerador que é a união disjunta de ciclos . Ao aplicar a operação G_I em G cada ciclo dobra de tamanho, tendo 2-vértices em cada clique de G_I. Como ciclo é par , tomamos metade dos vértices , um em cada clique. Aplicando o procedimento em todos ciclos disjuntos de G_I vindo do 2-fator temos um conjunto de *better influencers* de tamanho n, com tempo t=3, pois a outra metade dois vértices de cada clique é gerado no tempo t=2 e o restante dos vértices da clique em t=3. □

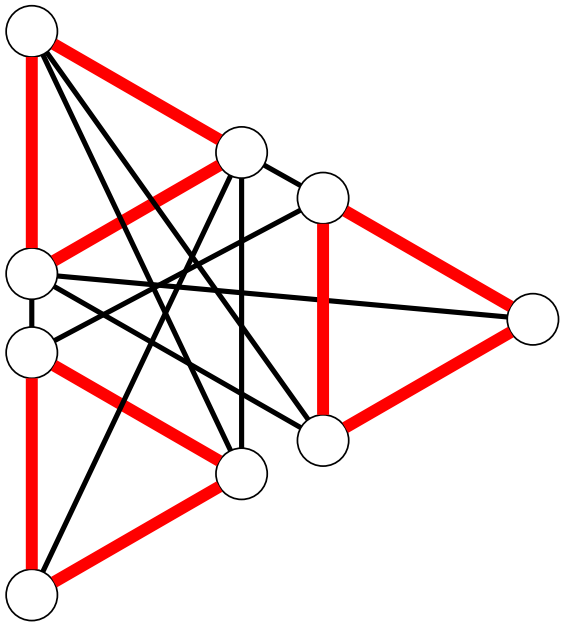

Figura 25 – Arestas vermelhas formam um 2-fator do grafo G

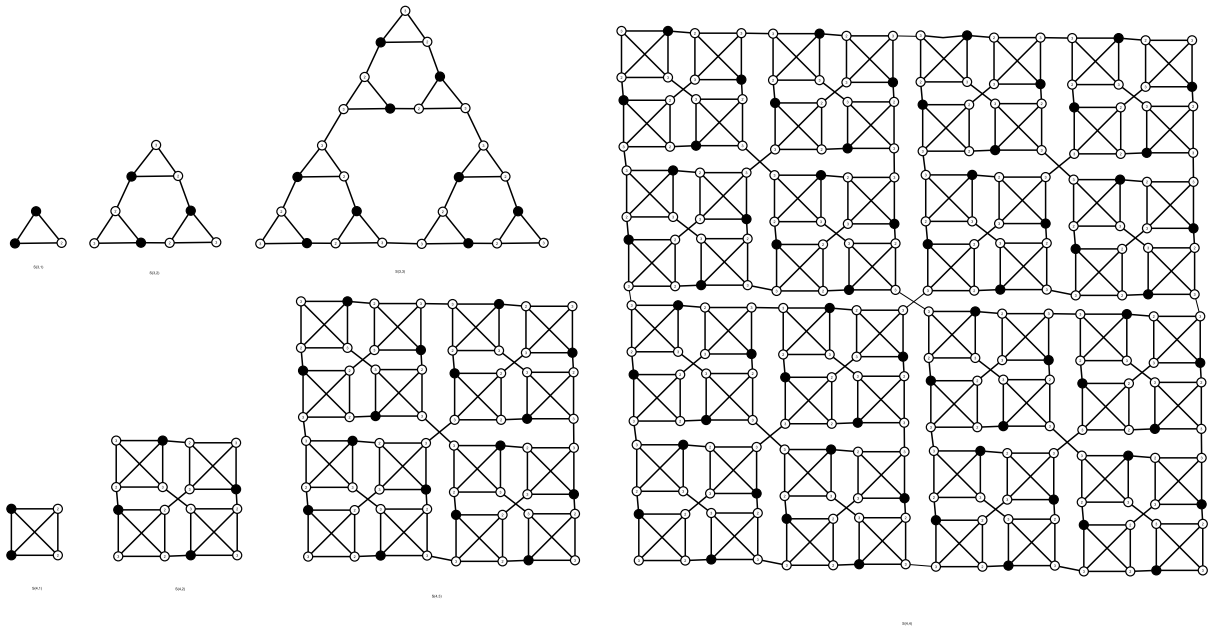

Figura 26 – Ilustração da contaminação t=3