Detecção de Comunidades em Redes Sociais

João Guilherme Lopes Alves da Costa, Matheus Leão de Carvalho

Universidade Federal do Rio Grande do Norte - UFRN / Departamento de Informática e Matemática Aplicada - DIMAp

Resumo

Este presente trabalho se prontifica a apresentar o uso da modelagem em grafos na detecção de comunidades em redes sociais. Dado que cada nó representa um usuário da rede e as arestas sejam os relacionamentos entre os usuários, é possível obter a detecção de comunidades nesse grafo através do problema de agrupamento de grafos (*graph clustering*). Porém, a grande maioria dos algoritmos pode ser ineficiente para resolver o problema dada a quantidade de dados que temos nas redes sociais atualmente. Dessa forma, será apresentado o algoritmo de Xingwang Zhao para redes sociais de larga escala através da compressão de grafos.

Palavras-chave: grafos; detecção de comunidades; redes sociais; agrupamento de grafos; compressão de grafos

1. Introdução

Dado o crescimento vertiginoso do uso de redes sociais, é de interesse científico a análise de redes para se entender sua estrutura. A técnica de detecção de comunidades é útil para se determinar a estrutura de relacionamentos em uma rede qualquer, mas neste trabalho será focado na aplicação para redes sociais.

As comunidades se caracterizam como uma propriedade de uma rede em que os nós de uma comunidade estão fortemente conectados (possuem um maior relacionamento entre esses nós em relação a outros do grafo). Fazendo a detecção de comunidades em uma rede, é possível, por exemplo, saber qual melhor tipo de conteúdo apresentar ao usuário, ou para fazer recomendação para um usuário de outras contas/usuários com o mesmo tipo de interesse.

2. Descrição do problema real

Em redes sociais, é de interesse da plataforma manter o usuário cada vez mais conectado e fazer com que ele passe mais tempo utilizando a plataforma. Uma estratégia para isso, é fornecer conteúdo de usuários do mesmo interesse dele. O problema se situa em como detectar qual o interesse do usuário, e quais desses interesses são de maior relevância para o usuário. Como exemplo, a rede social chinesa Tiktok teve um crescimento muito grande de usuários em um curto espaço de tempo, em grande parte devido ao seu algoritmo de recomendação [1].

É bastante comum em redes sociais, ao criar uma nova conta, fornecer uma lista de possíveis interesses do usuário para que ele selecione. Isso é uma forma de facilitar o fornecimento de assuntos de interesse do usuário. Conforme o usuário vai utilizando mais a rede, ele vai se engajamento mais em determinados tipos de assuntos e tendo menos interesse em outros, e a rede social deve acompanhar essa mudança para saber qual o assunto preferido do usuário em determinados momentos, com o intuito que ele se mantenha mais tempo na plataforma.

Uma comunidade pode ser observada por usuários que estejam mais conectados entre si, em relação a outros. Não necessariamente uma comunidade deve estar relacionada a um 'assunto' de interesse, podendo ser uma comunidade de amigos, de familiares, entre outros. Inicialmente, o foco dos pesquisadores para fazer a detecção de comunidades era em obter comunidades disjuntas. No entanto, no contexto de redes sociais é possível que um usuário esteja em mais de uma comunidade, e o problema seria em identificar quais comunidades o usuário pertence.

3. Modelo em grafos

A forma mais comum de modelar o problema de encontrar comunidades em redes sociais é representando as pessoas (usuários, perfis ou etc.) como nós (vértices) e representando as relações entre elas como ligações (arestas). Adicionalmente se pode dar um caráter de intensidade para as relações (geralmente chamado de grau de afinidade), sendo representado como uma rotulação das ligações.

Dessa forma, podemos representar a rede como um grafo $G = (V,E)$, onde V é o conjunto de n vértices e $E \subset V \times V$ é o conjunto de m arestas. No caso de redes ponderadas (em que as ligações entre os nós possuem pesos) representamos a rede como um grafo $G = (V,E,W)$ em que $W : E \rightarrow R_+$ é uma função que retorna o peso de uma ligação e V e E são como descrito anteriormente. Para representar a estrutura da rede, é possível utilizar mais de um TAD, aqui trataremos de uma em específico (matriz de adjacência $n \times n$): cada nó é representado como um número natural de 1 a n (ou 0 a $n-1$), e cada célula $e_{(i,j)}$ representa a ligação entre os nós i e j. Células com valor nulo representam ligações inexistentes, enquanto as células não nulas representam uma ligação. Em redes não ponderadas (redes em que as ligações apresentam rótulos ponderados, ou seja rotulados com a intensidade da ligação), cada célula não nula terá valor 1, enquanto que em uma rede ponderada, cada célula terá a intensidade da ligação.

O problema se trata de encontrar "*clusters*"("agrupamentos"em português), uma vez que uma comunidade em uma rede social é um conjunto de pessoas (usuários, perfis ou etc.) que têm mais coesão entre os membros do que com as pessoas de fora, e um cluster é um conjunto de nós mais densamente conectados do que com os demais. Note que a resolução do problema em grafos corresponde ao problema real, uma vez que ao encontrar os clusters de um grafo, se encontrou as comunidades em uma rede social. Desse modo, temos como objetivo a resolução de um problema em grafos bem conhecido e explorado na literatura.

A detecção de comunidades é um problema não somente relacionado a redes sociais, podendo ser generalizado para os mais diversos tipos. M. Girvan e M. E. J. Newman são dois pesquisadores famosos na área de detecção de comunidades. No artigo *Community structure in social and biological networks* [2], eles definem outra propriedade comum entre muitas redes, a **estrutura de comunidade**, e fazem um paralelo entre a rede biológica e a rede social.

4. Estado-da-Arte

Há uma série de algoritmos propostos na literatura para resolver o problema de detecção de comunidades, normalmente relacionados a detecção de *clusters* em grafos.

Os primeiros algoritmos propostos foram pensados para resolver o problema em redes 'pequenas', enquanto as redes que temos atualmente tiveram aumento exponencial na quantidade de dados.

4.1. Algoritmo de Louvain e Modularidade

Um dos algoritmos mais rápidos para detecção de comunidades, com complexidade $O(n \log n)$, é o *Louvain Community Detection*, proposto em 2008 na Universidade Católica de Louvain, por Blondel *et al* [3]. Este algoritmo é baseado na otimização da **modularidade** da rede. Modularidade mede a densidade de ligação entre nós dentro de uma comunidade quando comparado a ligação de nós entre comunidades. A modularidade é definida (para grafos não-ponderados, o peso das arestas poderá ser 0 ou 1) por:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

- A_{ij} representa o peso da aresta entre os nós i e j;
- k_i e k_j são as somas dos pesos das arestas ligadas aos nós i e j, respectivamente;
- m é a soma de todos os pesos das arestas no grafo;
- c_i e c_j são as comunidades dos nós;
- δ é a função Delta de Kronecker ($\delta(x,y) = 1$ se $x = y$, 0 caso contrário).

Entretanto, a otimização de modularidade máxima consiste em um problema **NP-dificil**, de forma que seja necessário utilizar heurísticas para esse algoritmo. O algoritmo proposto por Clauset *et al* [4] é a mais rápida aproximação de um algoritmo para otimização de modularidade em redes de larga escala, tendo a complexidade de $O(n \log^2 n)$ para redes esparsas.

O algoritmo de Louvain consiste em 2 etapas que são repetidas iterativamente:

1. O algoritmo inicia com uma rede ponderada de V vértices.
   (i) Nessa primeira fase, é atribuído uma comunidade para cada nó do grafo;
   (ii) Depois, para cada nó é considerado seus vértices adjacentes e analisado o ganho de modularidade removendo o nó analisado de sua comunidade e adicionando-o a comunidade de seu adjacente;

(iii) Um nó mudará para a comunidade de seu adjacente se o ganho de modularidade for positivo e máximo. O nó permanecerá na mesma comunidade caso não haja ganho de modularidade positiva;

(iv) Esta primeira fase irá terminar quando um máximo local de modularidade é alcançada, ou seja, quando nenhum movimento individual pode melhorar a modularidade.

2. A segunda fase consiste em construir uma nova rede em que os nós serão as comunidades encontradas na primeira fase. Para construir essa nova rede, o peso de uma aresta entre os novos nós será dada pela **soma dos pesos das arestas entre os nós das duas correspondentes comunidades**.

Além de sua velocidade, este algoritmo se destaca por sua fácil implementação. No entanto, uma de suas limitações é devido ao grande uso de memória na execução.

4.2. Detecção de Comunidades Sobrepostas em Redes

Dado que é comum em redes sociais a detecção de comunidades sobrepostas, foi feito um estudo comparativo dos algoritmos por Jierui *et al* no artigo *Overlapping Community Detection in Networks: The State-of-the-Art and Comparative Study* [5].

Nesse artigo são analisados 14 diferentes tipos de algoritmos. Os autores categorizam em 5 tipos de classes em como são identificados as comunidades: **Clique Percolação**, **Gráfico de Linhas e Particionamento de links**, **Expansão e Otimização Local**, **Detecção Fuzzy** e **Algoritmos baseados em agentes e dinâmicos**.

Algorithm	Reference	Complexity	Imp
CFinder	[Palla et al. 2005]	-	C++
LFM	[Lancichinetti et al. 2009]	$O(n^2)$	C++
EAGLE	[Shen et al. 2009a]	$O(n^2 + (h+n)s)$	C++
CIS	[Kelley 2009]	$O(n^2)$	C++
GCE	[Lee et al. 2010]	$O(mh)$	C++
COPRA	[Gregory 2010]	$O(vm\log(vm/n))$	Java
Game	[Chen et al. 2010b]	$O(m^2)$	C++
NMF	[Psorakis et al. 2011]	$O(kn^2)$	Matlab
MOSES	[McDaid and Hurley 2010]	$O(en^2)$	C++
Link	[Ahn et al. 2010]	$O(nk^2_{max})$	C++
iLCD	[Cazabet et al. 2010]	$O(nk^2)$	Java
UEOC	[Jin et al. 2011]	$O(ln^2)$	Matlab
OSLOM	[Lancichinetti et al. 2011]	$O(n^2)$	C++
SLPA	[Xie et al. 2011; Xie and Szymanski 2012]	$O(tm)$	C++

Figura 1: Algoritmos incluídos no experimento.

Os resultados obtidos foram:

- Para redes com baixa densidade de sobreposição, os algoritmos que ofereceram melhor performance em relação aos outros foram: SLPA, OSLOM, Game, and COPRA.
- Para redes com alta densidade de sobreposição e alta diversidade de sobreposição, tanto SLPA e Game forneceram desempenho relativamente estável.
- Um último resultado observado, foi que uma característica comum sobre vários algoritmos em redes do mundo real é que possui um número relativamente pequeno de nós sobrepostos (normalmente 30%), e cada um pertencendo a poucas comunidades, em média 2 ou 3.

4.3. Outros algoritmos

Podem ser citados outros algoritmos como: *Surprise Community Detection*, *Leiden Community Detection* e *Walktrap Community Detection*. Também pode se citar o uso do **Problema da Clique** para detecção de comunidades. Clique é um subgrafo completo em um grafo, obtendo Cliques, tem-se comunidades mais fortemente conectadas. No entanto, encontrar Clique Máxima em grafos é um problema NP-difícil.

5. Revisão bibliográfica do problema real e do modelo em grafos

Há um vasto campo de pesquisa que cresce cada vez mais para estudar como tornar os algoritmos de detecção de comunidades melhores. Apesar da pesquisa ter aumento principalmente pelas redes sociais, os resultados desses estudos podem ser usadas em diversas aplicações de inteligência artificial. A detecção de comunidades, por exemplo, tem aplicações não só em redes sociais, mas para diferentes tipos de rede, por exemplo redes biológicas.

5.1. Problema real: recomendação de conteúdo em redes sociais

A respeito do problema real, existem vários estudos sobre como o algoritmo de recomendação é crucial no sucesso de uma rede social. Por exemplo, a rede social Tiktok é uma grande referência quando o assunto é recomendação, e normalmente é atribuído esse fato ao seu grande sucesso [1]. Outra rede social que há bastante artigo a respeito é o algoritmo do Twitter [6] [7], visto que é uma rede social mais estabelecida e ainda possui um algoritmo de recomendação bastante sofisticado. É utilizado modelagem em grafos e o problema de detecção de comunidades.

Apesar de não pertencer ao escopo de 'rede social', pode-se citar o exemplo da Netflix, que é a plataforma de *streaming* mais famosa na atualidade. Além do fato de ter sido pioneira no mercado de *streaming*, um dos motivos pela manutenção de seu sucesso perpassa por seu algoritmo de recomendação [8] [9] [10].

5.2. Modelo em grafos: detecção de comunidades em grafos

Tratando do modelo em grafos, existem várias estratégias para resolver o problema de detecção de comunidades. O mais comum encontrado na literatura é a técnica de *Clustering*. Ademais, o mais comum encontrado são algoritmos que detectam comunidades sem sobreposição, ou seja, o caso em que um vértice não pode estar em mais de uma comunidade. Nesses casos, o principal interesse dos algoritmos é encontrar conjuntos disjuntos entre o conjunto de vértices do grafo.

Na pesquisa, é encontrado um comparativo entre alguns algoritmos de detecção de comunidades com sobreposição. Porém, seu resultado conclui que a utilização desses algoritmos em redes reais (com grande quantidade de dados) ainda não é um problema

totalmente resolvido, visto que foi obtido uma quantidade de nós sobrepostos relativamente pequena (normalmente inferior a 30%).

6. Casos de teste ou geradores de casos de teste encontrados na literatura

A biblioteca Python CDlib - Community Discovery Library, pode ser usada para extrair, comparar e avaliar comunidades de redes complexas. Ela oferece a implementação de métodos como Algoritmo de Louvain, Surprise, Leiden, Walktrap, Modularidade de Girvan e Newman, entre outros. Outra com proposta semelhante, também em Python, é a biblioteca communities.

Para trabalhar em conjunto com a CDlib é recomendado a NetworkX, pois ela possui casos de teste disponíveis para o usuário, tais como o Clube de Karatê do Zachary e outros exemplos.

```python
from cdlib import (algorithms, evaluation)
import networkx as nx

G = nx.karate_club_graph()
coms1 = algorithms.louvain(G, weight='weight', resolution=1., randomize=False)
coms2 = algorithms.surprise_communities(G)
coms3 = algorithms.leiden(G)
mod = evaluation.newman_girvan_modularity(G, coms3)
```

Figura 2: Exemplo de uso das bibliotecas Python CDlib e NetworkX.

6.1. Clube de Karatê do Zachary

Relacionado a parte de casos de teste, um dos mais famosos encontrados é o *Zachary's Karate Club* ("**Clube de Karatê do Zachary**"). O Clube de Karatê do Zachary é uma rede social que modela um clube de Karatê universitário, e foi descrito no artigo *An Information Flow Model for Conflict and Fission in Small Groups* de Wayne W. Zachary. No entanto, sua popularidade maior se deve ao seu uso por Girvan e Newman em seu artigo *Community structure in social and biological networks* [2].

Um conjunto de dados gratuito do Clube de Karatê do Zachary pode ser encontrado gratuitamente na internet 7.2, ou mesmo por meio da biblioteca Python NetworkX.

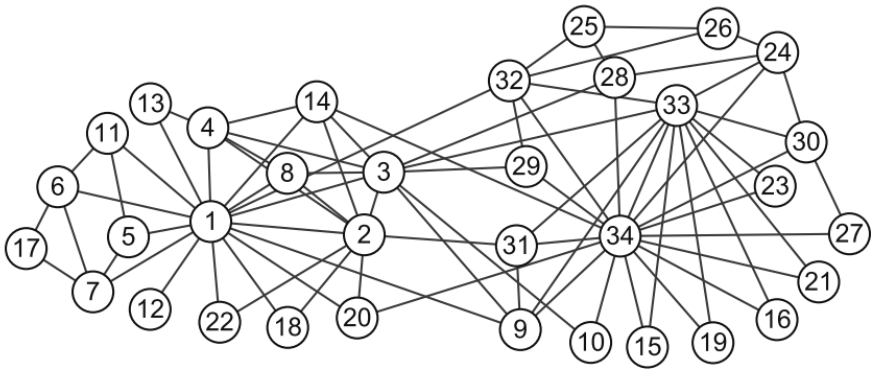

Figura 3: Ilustração do Grafo da Rede do Clube de Karatê do Zachary

6.2. Casos de teste para o Algoritmo de Zhao

No artigo *A community detection algorithm based on graph compression for large-scale social networks* por Zhao *et al* [11], ele utiliza de diversos casos de teste conhecidos na literatura, são esses:

Datasets	# Vertices	# Arestas	Comunidades
Karate [2]	34	78	2
Dolphin [12]	62	159	2
Football [2]	115	613	12
Polbooks [13]	105	441	3
Polblogs [2]	1,490	16,718	2
Email [14]	1,133	5,451	NA
PGP [15]	10,681	24,316	NA
CA_AstroPh [16]	18,772	396,160	NA
CA_CondMat [16]	23,133	186,936	NA
Email_Enron [16]	36,692	183,831	NA
soc_Epinions [17]	75,879	508,837	NA
Email_EuAll [16]	265,214	420,045	NA
com_Youtube [18]	1,134,890	2,987,624	NA
WikiTalk [18]	2,394,385	5,021,410	NA

Tabela 1: Redes sociais utilizadas como casos de teste no Algoritmo de Zhao

7. Links (mais importantes) da internet relacionados ao problema real e ao modelo em grafos

7.1. Algoritmo do Twitter Open-source

`https://github.com/twitter`: Recentemente, com a mudança de gestão do Twitter, foi feito a divulgação dos seus algoritmos gratuitamente no Github. Nesse link, é possível ter acesso a todos os projetos do Twitter divulgados no Github.

`https://blog.twitter.com/engineering/en_us/topics/open-source/2023/twitter-recommendation-algorithm`: Esse link é um artigo do Twitter abordando sobre seu algoritmo de recomendação.

`https://github.com/twitter/the-algorithm`: Este projeto no Github contém o código fonte dos algoritmos de recomendação do Twitter.

Em SimClusters é possível ver uma descrição da modelagem da rede em grafos bipartidos e como é feito a detecção de comunidades.

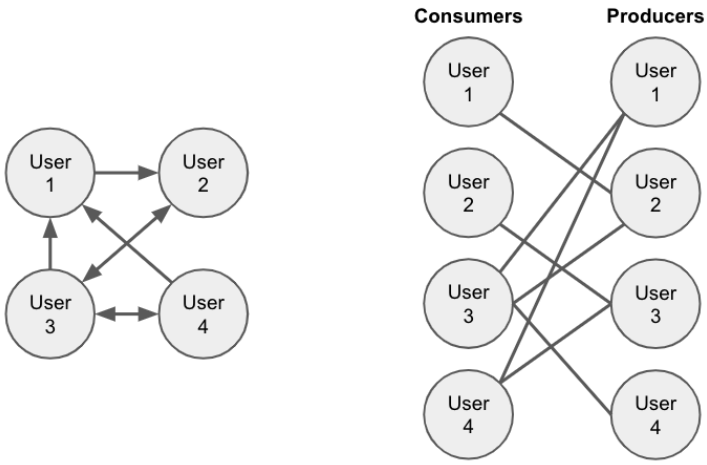

Figura 4: Modelagem em grafos da rede do Twitter.

7.2. Clube de karatê do Zachary

`http://konect.cc/networks/ucidata-zachary/`: Um site que possui um dataset gratuito do Clube de Karatê do Zachary, uma rede social que descreve bem a estrutura de comunidade.

7.3. Bibliotecas para Detecção de Comunidades

Links para algumas bibliotecas Python para trabalhar com detecção de comunidades em redes: `https://cdlib.readthedocs.io/en/latest/`, `https://networkx.org/` e `https://pypi.org/project/communities/`.

7.4. Detecção de Comunidades

Community Detection - Overview (ScienceDirect): Este link contém uma página do site ScienceDirect, um plataforma para divulgação de revistas científicas, que exibe vários artigos relevantes relacionados a Detecção de Comunidades.

`https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae`: Essa matéria na plataforma Medium por Thamindu Dilshan Jayawickrama traz de forma resumida e com linguagem mais simplificada a temática de detecção de comunidade e alguns algoritmos populares.

`https://senseable.mit.edu/community_detection/`: Neste site é possível ter acesso a outra proposta de otimização de modularidade chamada Combo.

`https://www.revistasg.uff.br/sg/article/view/V9N4A6`: Este é um experimento da aplicação do problema de detecção de comunidades em redes sociais em um grafo de relacionamento de alunos de uma instituição de ensino superior.

8. Proposta de Abordagem Algorítmica

Utilizaremos o algoritmo descrito no artigo *A community detection algorithm based on graph compression for large-scale social networks* [11] por Zhao *et al.*

8.1. Descrição do Algoritmo Implementado

O algoritmo de Zhao *et al.* é dividido em 4 partes:

- Compressão (*Graph compression*);
- Determinação de sementes de comunidade (*Community seed determination*);
- Expansão da semente (*Seed expansion*);
- Propagação da estrutura de comunidade (*Community structure propagation*).

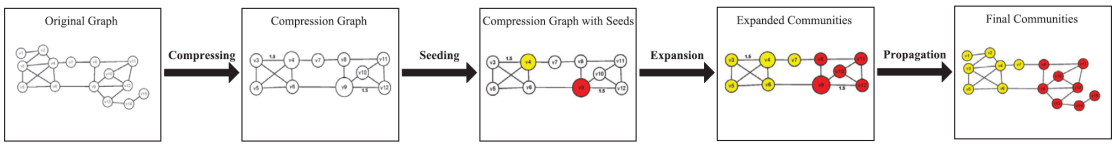

Figura 5: Visão geral do algoritmo proposto.

8.1.1. Compressão

Essa parte consiste na compressão de partes do grafo para o mesmo nó, que será chamado de super-nó. O algoritmo é iniciado copiando o grafo para um grafo comprimido, sobre o qual as operações serão feitas. Nele, enquanto houverem nós de grau 1

ou 2 (que não sejam pontes - nós de grau 2, os quais seus 2 vizinhos não são vizinhos entre si), eles serão incorporados em outros nós.

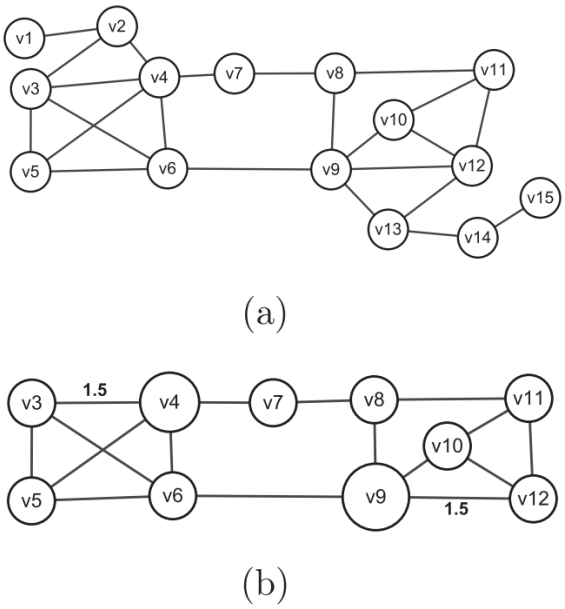

Figura 6: Exemplo de compressão de grafo utilizando o algoritmo: (a) grafo original; (b) o grafo comprimido.

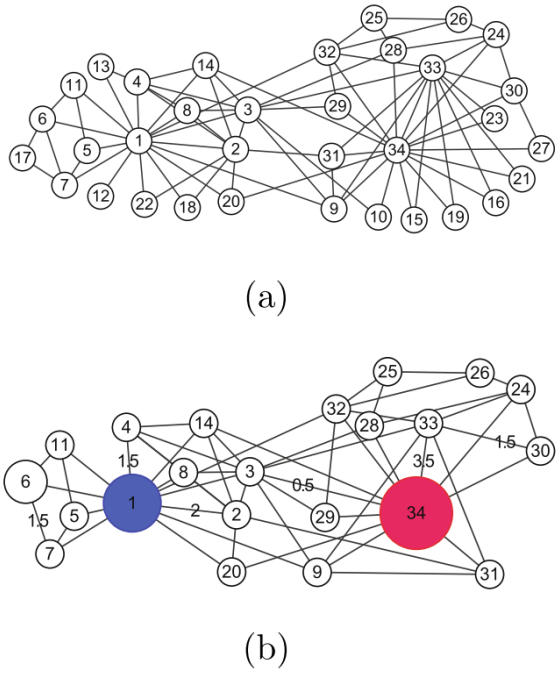

Figura 7: Exemplo de compressão de grafo na rede do Clube de Karatê utilizando o algoritmo: (a) grafo original do Clube de Karatê; (b) grafo comprimido do Clube de Karatê.

Algoritmo 1: O Algoritmo de Compressão de Grafos

Entrada: Grafo $G = (V, E, W)$.

Saída: Grafo Comprimido $G^c = (V^c, E^c, W^c)$.

Calcular os graus de cada vértice $v_i \in V$, $d(v_i) = |\{v_j | (v_i, v_j) \in E, v_j \in V\}|$;

Inicializar os conjuntos dos vértices inclusos para cada vértice $v_i \in V$ antes da compressão, $IV(v_i) = \{v_i\}$;

Inicializar os conjuntos de vértices com grau 1 e 2, respectivamente, $D1 = \{v_i | d(v_i) = 1, v_i \in V\}, D2 = \{v_i | d(v_i) = 2, v_i \in V\}$;

Inicializar o grafo comprimido $G^c = G$;

repita

  para cada $v_i \in D1$ **faça**

    $V^c = V^c - \{v_i\}$, $E^c = E^c - \{(v_i, v_j)\}$, onde $(v_i, v_j) \in E^c$;

    $IV(v_j) = IV(v_j) \cup \{v_i\}$;

    $d(v_i) = 0$, $d(v_j) = d(v_j) - 1$;

    Adicionar o vértice v_j em $D1$ ou $D2$ de acordo com seu novo grau;

    $D1 = D1 - \{v_i\}$;

  fim

  para cada $v_i \in D2$ **faça**

    se v_i *não é um nó ponte* **então**

      $V^c = V^c - \{v_i\}$, $E^c = E^c - \{(v_i, v_j), (v_i, v_k)\}$, $E^c = E^c \cup \{v_j, v_k\}$, onde $(v_i, v_j), (v_i, v_k) \in E^c$;

      se $W^c(v_j, v_k) > 0$ **então**

        $d(v_j) = d(v_j) - 1$, $d(v_k) = d(v_k) - 1$;

      fim

      $W(v_j, v_k) = W(v_j, v_k) + \frac{1}{2} \cdot W(v_i, v_j) \cdot W(v_i, v_k)$;

      $d(v_k) = 0$, $IV(v_j) = IV(v_j) \cup \{v_i\}$, onde $d(v_j) \geq d(v_k)$, $(v_i, v_j), (v_i, v_k) \in E^c$;

      Adicionar os vértices v_j e v_k em $D1$ ou $D2$ de acordo com seus novos graus;

    fim

    $D2 = D2 - \{v_i\}$

  fim

até $D1 = \varnothing$ *e* $D2 = \varnothing$;

retorna *Grafo Comprimido* $G^c = (V^c, E^c, W^c)$.

8.1.2. Determinação de sementes de comunidade

Essa parte consiste na determinação de quais nó serão sementes (ou centros) de uma comunidade utilizando como base o grafo comprimido obtido na primeira parte e pode ser dividida em duas ”sub-partes”:

- Cálculo do índice de centralidade:

  São calculadas densidade e qualidade de cada nó a partir do seu grau e quantidades de vértices e arestas que contém (caso seja um super-nó). Posteriormente os valores de densidade e qualidade são normalizados. É calculado um índice de centralidade como o produto de densidade e qualidade (normalizadas).

- Determinação da semente:

  Os nós são ordenados pelo seu índice de centralidade e é calculada a segunda derivada, o ponto de pico é dado como a maior segunda derivada. Todos os nós que tem valor de índice de centralidade maior ou igual ao ponto de pico são marcados como semente. Posteriormente, para cada dupla de nós adjacentes que são sementes, um deles deixa de o ser, até que não hajam mais pares de sementes adjacentes.

Tome:

$$\rho(v_i) = d(v_1) \tag{1}$$

$$\mu(v_i) = |IV(v_i)| \tag{2}$$

$$\rho'(v_i) = \frac{\rho'(v_i)}{max\rho(v)_{v\in V^c}} \tag{3}$$

$$\mu'(v_i) = \frac{\mu'(v_i)}{max\mu(v)_{v\in V^c}} \tag{4}$$

$$\gamma(v_i) = \rho'(v_i)\cdot\mu'(v_i) \tag{5}$$

$$h_i = |(g_i - g_{i+1}) - (g_{i+1} - gi+2)| \tag{6}$$

$$kp = argmax_{i=n_c-2,\ldots,1}h_i \tag{7}$$

$$CS = v_1|\gamma(v_i) \geq g_{kp}, v_i \in V^c \tag{8}$$

Algoritmo 2: Algoritmo de determinação das sementes de comunidade

Entrada: grafo comprimido $G^c = (V^c, E^c, W^c)$ com n_c vértices.

Saída: Sementes de comunidade *CS*.

1 Compute a densidade e qualidade dos vértices $v_i \in V^c$ e os normalize de acordo com as equações 3 e 4;

2 Compute o índice de centralidade $\gamma(v_i)$ para $v_i \in V^c$ e construa a sequência decrescente $g_1 \geq g_2 \geq \cdots \geq g_{n_c}$ de acordo com a equação 5;

3 Compute a segunda derivada h de acordo com a equação 6;

4 Encontre o ponto de inflexão de h de acordo com a equação 7;

5 Determine as potencias sementes de comunidade *CS* de acordo com a equação 8;

6 **se** $W^c(v_i, v_j) \neq 0, v_i, v_j \in CS$ **então**

7 $\quad CS = CS - v_i$;

8 **fim**

9 **retorna** *Sementes de comunidade CS.*

8.1.3. Expansão da semente

Essa parte consiste em atribuir todos os nós que não são sementes a uma comunidade utilizando o resultado dos últimos 2 passos, e será dividida em e subpartes:

- Cálculo da similaridade

  A similaridade de um no u com uma comunidade é calculada como o somatório de todos os pesos das arestas de u a v mais o somatório dos inversos dos pesos das arestas v' e v'' tais que v'' são todos os vizinhos de v', v' são os adjacentes de u e v simultaneamente e v são todos os vizinhos de u que fazem parte da comunidade.

- Determinação de comunidade

  Para cada nó, ele vai ser inserido na comunidade que tiver mais afinidade (note que é possível que tenha afinidades iguais com mais de uma comunidade, caso isso ocorra para a afinidade máxima, deve ser usado algum critério para escolher a comunidade).

Tome:

$$sim(u,C) = \sum_{v \in N(u) \bigcap C} W^c(u,v) + \sum_{v \in N(u) \bigcap C} \sum_{v' \in N(u) \bigcap N(v)} \frac{1}{\sum_{v'' \in N(v')} W^c(v',v'')}$$

Algoritmo 3: Algoritmo de expansão da semente

Entrada: Grafo comprimido $G^c = (V^c, E^c, W^c)$ e suas sementes de comunidade $CS = cs_1, cs_2, \cdots, cs_k$.

Saída: Comunidades $\mathscr{C}^c$.

Inicialize as comunidades $\mathscr{C}^c = C_1, c_2, \cdots, C_k$ tais que $C_i = cs_i$ para $i = 1, 2, \cdots, k$;
Inicialize o conjunto de vértices não rotulados $UL = V^c - CS$;
enquanto $UL \neq \emptyset$ **faça**
    $\mathscr{TC} = TC_1, TC_2, \cdots TC_k$, onde $TC_i = \emptyset$;
    Compute os vizinhos das comunidades $CV = \bigcup CN(C_i)$ para $i = 1, 2, \ldots, k$;
    para cada *vértice* $u \in CV$ **faça**
        se *u é não rotulado* **então**
            se $N(u) \subseteq C_i$ *para* $i = 1, 2, \ldots, k$ **então**
                $TC_i = TC_i \bigcup u$;
            fim
            senão
                $h = argmaxsim(u, C_i)_{i=1,2,\ldots,k}$, $TC_h = TC_h \bigcup u$;
            fim
            $UL = UL - u$;
        fim
    fim
    Atualize $C_i - C_i \bigcup TC_i$ para $i = 1, 2, \ldots, k$;
fim
retorna *Comunidade* $\mathscr{C}^c$.

8.1.4. Propagação da estrutura de comunidade

Nessa parte, nós ”descomprimimos”os super-nós e atribuímos cada nó do super-nó à comunidade em que este estava presente. Obtemos então um grafo igual ao inicial (não comprimido) e com todos os nós atribuídos a exatamente uma comunidade.

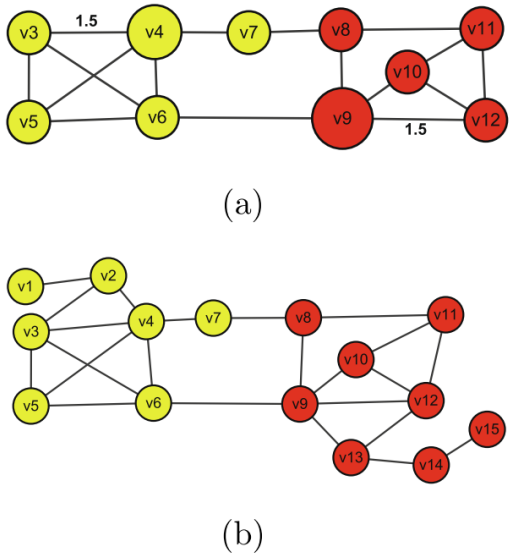

Figura 8: Comunidades do grafo original e do grafo comprimido: (a) grafo original; (b) grafo comprimido.

Algoritmo 4: Algoritmo de propagação de comunidades

Entrada: Grafo $G = (V, E, W)$, grafo comprimido $G^c = (V^c, E^c, W^c)$ e as comunidades de G^c: $^c = C_1, C_2, \ldots, C_k$.

Saída: Comunidades de G: $\mathscr{C}$.

```
1 Inicializa comunidades de G: 𝒞 = 𝒞^c;
2 para cada comunidade C_i ∈ 𝒞^c faça
3     para cada vértice u ∈ C_i faça
4         se IV(u) > 1 então
5             Atualize C_i = C_i ⋃ IV(U) fim
6         fim
7     fim
8     retorna Comunidade 𝒞.
9
```

8.2. Análise de Complexidade

O algoritmo é dividido em 4 partes, portanto analisaremos primeiramente a complexidade de cada parte. Tome as seguintes informações:

- m é a quantidade de arestas da entrada;
- n é a quantidade de vértices (nós) da entrada;
- t é a quantidade de iterações feitas na compressão;
- n_c é a quantidade de vértices do grafo comprimido;
- $\overline{C}$ é o tamanho médio das comunidades durante a execução;
- k é a quantidade de comunidades.

Analisamos cada parte do algoritmo:

- Compressão do grafo:

  Comprimimos todos os nós de grau 1 e todos os nós de grau 2 que não sejam pontes - no escopo desse algoritmo tome a seguinte definição de ponte: nó de

grau 2 tal que seus adjacentes não são adjacentes entre si - até que não sobre nenhum nó dos tipos descritos. Para tal, se visita no máximo 2 vezes cada aresta - uma vez por cada vértice terminal - e no máximo 1 vez cada vértice (além dos acessos feitos a partir da aresta) em cada iteração. Temos que a complexidade de cada iteração é $O(m+n)$ e de todo o algoritmo de compressão $O((m+n)t)$.

- Determinação das sementes de comunidade:

  Neste algoritmo, o gargalo é dado pela operação de ordenação dos nós por um parâmetro chamado índice de centralidade, logo é $O(n_c \log n_c)$.

- Expansão das sementes de comunidade:

  Neste algoritmo o gargalo é dado pelos cálculos das similaridades entre cada comunidade e cada nó. Dado que a complexidade do cálculo da similaridade entre uma comunidade e um nó é linear em relação ao tamanho da comunidade, temos que a complexidade do calculo das similaridades de um nó com cada comunidade é $O(\overline{C}k)$. Uma vez que existem n_c nós no grafo comprimido, a complexidade dessa parte é $O(n_c\overline{C}k)$.

- Propagação das comunidades

  Nesse algoritmo percorremos cada nó de cada supernó e os rotulamos, logo é $O(n)$ já que passa uma vez por cada nó.

Temos então que a complexidade do algoritmo todos é a soma das complexidades das partes, ou seja $O((m+n)t + n_c\overline{C}k)$.

Note que a taxa de crescimento da implementação do algoritmo pode diferir da prevista pela complexidade devido a limitações das estruturas de dados empregadas.

8.3. Descrição dos experimentos computacionais

8.3.1. Base de dados

Foram utilizados casos de teste da literatura e os arquivos foram obtidos a partir de alguns repositórios no Github, como:

- `https://github.com/xmweijh/CommunityDetection/tree/main/data`

- `https://github.com/RapidsAtHKUST/CommunityDetectionCodes/tree/master/Datasets`
- `https://github.com/zzz24512653/CommunityDetection/tree/master/network`

Os casos de testes utilizados foram: dolphin, football, karate, exemplo do artigo de Louvain e o exemplo do artigo de Zhao.

Foram adicionados outros arquivos de teste ao projeto, no entanto, por limitação de memória não foi possível utilizar como casos de teste. O limite máximo de elementos de cada std::set foi de 1000 elementos, nos exemplos que passaram disso foi dado erro de *segmentation fault*.

8.3.2. Configuração de hardware e software

Para a realização da análise empírica (geração dos tempos de execução de cada algoritmo) foi utilizado um computador com as seguintes especificações:

- Processador: AMD Ryzen 3 3100 4-Core Processor 3.60 GHz
- Memória Ram: 2 memórias de 8GB DDR4 (16GB)
- Placa de Vídeo: Radeon RX 5500 XT 8GB OC

A geração dos tempos foi realizado através do sistema operacional Windows com utilização do Subsistema Windows para Linux para a compilação e execução:

- Sistema operacional: Windows 11 Home 64bits
- Windows Subsystem for Linux 2 (WSL 2), sistema operacional Ubuntu 20.04 LTS
- g++ 9.4.0

8.3.3. Detalhes adicionais

Para ter acesso a todo o código implementado do algoritmo basta acessar o projeto em `https://github.com/joaoguilac/community_detection/tree/main`.

Para executar o algoritmo é necessário ter instalado o g++ e compilar com a versão 17 ou superior do C++. No arquivo README.md é apresentado todo o passo a passo de como realizar a compilação e execução.

8.3.4. Resultados obtidos

Datasets	# Vertices	# Arestas	Comunidades	Tempo médio de execução
Louvain exemplo [3]	16	28	4	0.887209ms
Zhao exemplo [11]	15	23	2	0.761544ms
Karate [2]	34	78	2	1.91392ms
Dolphin [12]	62	159	2	4.89031ms
Football [2]	115	613	12	17.4061ms

Tabela 2: Resultados da execução da implementação do algoritmo de Zhao

A qualidade de um algoritmo de detecção de comunidades é difícil de ser quantificada, visto que cada aplicação necessita que uma forma diferente de identificar as comunidades. Desse modo o algoritmo apresentado pode ser mais ou menos eficiente a depender da aplicação.

No geral, pelo observado, não teremos um grande número de comunidades mesmo para grandes redes, normalmente vários vértices estão contidos numa mesma comunidade. Ademais, por mais que o algoritmo tenha sido proposto para ser para redes sociais de larga escala, muitos casos de teste não foram possíveis de utilizar dada a quantidade de memória pela implementação feita. Mesmo que seja problema da implementação feita, no artigo original, o tempo de execução da maioria dos algoritmos passa de 10h.

9. Conclusão

Em suma, a detecção de comunidades pode ser bastante útil para redes sociais, sendo uma ferramenta importante para algoritmos de recomendação de conteúdo. Ademais existe uma literatura ampla sobre o tema, com diversas abordagens e soluções diferentes, a depender das necessidades e características da rede. O problema muda de

figura com o tamanho da entrada, tendo, ao longo dos anos se utilizado novas abordagens no tema, com algoritmos novos.

Referências

[1] M. Zhang, Y. Liu, A commentary of tiktok recommendation algorithms in mit technology review 2021, Fundamental Research 1 (6) (2021) 846–847. doi:https://doi.org/10.1016/j.fmre.2021.11.015.
URL https://www.sciencedirect.com/science/article/pii/S2667325821002235

[2] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (12) (2002) 7821–7826. arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.122653799, doi:10.1073/pnas.122653799.
URL https://www.pnas.org/doi/abs/10.1073/pnas.122653799

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) P10008. doi:10.1088/1742-5468/2008/10/P10008.
URL https://dx.doi.org/10.1088/1742-5468/2008/10/P10008

[4] A. Clauset, M. E. J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111. doi:10.1103/PhysRevE.70.066111.
URL https://link.aps.org/doi/10.1103/PhysRevE.70.066111

[5] J. Xie, S. Kelley, B. K. Szymanski, Overlapping community detection in networks: The state-of-the-art and comparative study, ACM Comput. Surv. 45 (4) (aug 2013). doi:10.1145/2501654.2501657.
URL https://doi.org/10.1145/2501654.2501657

[6] H. G. Elmongui, R. Mansour, H. Morsy, S. Khater, A. El-Sharkasy, R. Ibrahim, Trupi: Twitter recommendation based on users' personal interests, in: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing, Springer International Publishing, Cham, 2015, pp. 272–284.

[7] P. Gupta, V. Satuluri, A. Grewal, S. Gurumurthy, V. Zhabiuk, Q. Li, J. Lin, Real-time twitter recommendation: Online motif detection in large dynamic graphs, Proc. VLDB Endow. 7 (13) (2014) 1379–1380. doi:10.14778/2733004.2733010.
URL https://doi.org/10.14778/2733004.2733010

[8] C. A. Gomez-Uribe, N. Hunt, The netflix recommender system: Algorithms, business value, and innovation, ACM Trans. Manage. Inf. Syst. 6 (4) (dec 2016). doi:10.1145/2843948.
URL https://doi.org/10.1145/2843948

[9] X. Amatriain, J. Basilico, Recommender Systems in Industry: A Netflix Case Study, Springer US, Boston, MA, 2015, pp. 385–419. doi:10.1007/978-1-4899-7637-6_11.
URL https://doi.org/10.1007/978-1-4899-7637-6_11

[10] H. Steck, L. Baltrunas, E. Elahi, D. Liang, Y. Raimond, J. Basilico, Deep learning for recommender systems: A netflix case study, AI Magazine 42 (3) (2021) 7–18. doi:10.1609/aimag.v42i3.18140.
URL https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/18140

[11] X. Zhao, J. Liang, J. Wang, A community detection algorithm based on graph compression for large-scale social networks, Information Sciences 551 (2021) 358–372. doi:https://doi.org/10.1016/j.ins.2020.10.057.
URL https://www.sciencedirect.com/science/article/pii/S0020025520310574

[12] W. Luo, N. Lu, L. Ni, W. Zhu, W. Ding, Local community detection by the nearest nodes with greater centrality, Information Sciences 517 (2020) 377–392.

doi:https://doi.org/10.1016/j.ins.2020.01.001.
URL https://www.sciencedirect.com/science/article/pii/S0020025520300013

[13] V. Krebs, Books about us politics, http://www.orgnet.com/ (2004).

[14] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, A. Arenas, Self-similar community structure in a network of human interactions, Phys. Rev. E 68 (2003) 065103. doi:10.1103/PhysRevE.68.065103.
URL https://link.aps.org/doi/10.1103/PhysRevE.68.065103

[15] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (2004) 056122. doi:10.1103/PhysRevE.70.056122.
URL https://link.aps.org/doi/10.1103/PhysRevE.70.056122

[16] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: Densification and shrinking diameters, ACM Trans. Knowl. Discov. Data 1 (1) (2007) 2–es. doi:10.1145/1217299.1217301.
URL https://doi.org/10.1145/1217299.1217301

[17] M. Richardson, R. Agrawal, P. Domingos, Trust management for the semantic web, in: D. Fensel, K. Sycara, J. Mylopoulos (Eds.), The Semantic Web - ISWC 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 351–368.

[18] J. Yang, J. Leskovec, Defining and evaluating network communities based on ground-truth, in: Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics, MDS '12, Association for Computing Machinery, New York, NY, USA, 2012. doi:10.1145/2350190.2350193.
URL https://doi.org/10.1145/2350190.2350193