Pontifícia Universidade Católica
do Rio de Janeiro

Maximização de Influência em Redes Sociais

João Victor Galindo Borges do Rego

Relatório de Projeto Final de Graduação

Orientador: Prof. Marco Antonio Casanova

Centro Técnico Científico - CTC
Departamento de Informática
Curso de Graduação em Ciência da Computação

Rio de Janeiro, Setembro de 2021

Sumário

1. Introdução

Grafos são ferramentas extremamente úteis que nos ajudam a compreender sistemas e relações complexas entre dados. Podemos usar grafos para, por exemplo, representar uma rede social, demonstrando as relações entre as pessoas em suas arestas. Podemos ainda usar grafos para representar espaços físicos, como no problema clássico do caixeiro viajante, onde um grafo representa as estradas entre cidades. Grafos são então úteis para estudar entidades e relacionamentos binários entre elas.

Para os fins deste projeto, iremos focar em grafos como uma rede social. O estudo de redes sociais é importante para entendermos padrões estruturais presentes em relações entre pessoas, e como esses padrões podem afetar a nossa vida, como, por exemplo, a difusão de informações falsas em redes sociais [4, 9].

Apesar de ser cada vez mais comum, hoje em dia, representar redes sociais como o Facebook ou Twitter, estudos em redes sociais existem desde antes da invenção da Internet [1]. Portanto elas não são necessariamente espaços *online*, mas sim qualquer espaço em que pessoas participem de algum tipo de troca entre elas [5], sendo a troca de objetos físicos ou de ideias e conceitos (escolas, universidades, escritórios, etc.). Podemos representar uma rede social como um grafo e aplicar técnicas de análise de grafos para achar respostas a problemas específicos à rede social [1]. Em conclusão, a análise de problemas representados por grafos é extremamente valiosa para nossa sociedade cada vez mais guiada por dados.

Este projeto aborda o problema específico de maximização de influência (*Influence Maximization* – IM) em comunidades, definido como:

- Dado um grafo não dirigido com pesos, *G*, dividido em comunidades, encontre um subconjunto de nós que maximizem a propagação de influência em cada comunidade.

Dado um grafo não dirigido com pesos *G*, intuitivamente uma *comunidade C* de *G* é um subgrafo de *G* em que os nós de *C* possuem mais arestas entre si do que entre os nós de *G* que não estão em *C*. Por sua vez, a propagação de influência em um grafo pode ser definida de várias formas. Este trabalho adota o coeficiente de centralidade de Katz[1] para medir a influência de um nó em um grafo.

Maximização de influência é um problema clássico em análise de redes sociais usado para encontrar os indivíduos mais influentes em uma rede, mas possui várias outras aplicações. Por exemplo, o problema de encontrar o pior caso de contágio com *K* alunos infectados, ou seja, o conjunto com *K* alunos que infectam o maior número possível de

[1] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.katz_centrality.html

outros alunos pode ser recontextualizado como um problema de maximização de influência. O trabalho adota uma variação do algoritmo *Community Based Influence Maximization* (CBIM) [14] para computar os nós com influência máxima em cada comunidade. De forma bem simplificada, o CBIM recebe como entrada um grafo não dirigido com pesos, *G*, o número desejado de nós mais influentes *k*, e o limite de índice de consolidação δ, divide *G* em comunidades, e computa uma lista *SN* com *k* nós mais influentes, distribuídos proporcionalmente entre as comunidades de acordo com o número de nós de cada comunidade. A seção 2 deste trabalho descreve em detalhe uma implementação do CIBM.

O trabalho relata experimentos com o CBIM em diversos grafos, explorando diferentes combinações de valores de *k* e δ. Os resultados destes experimentos e suas comparações serão demonstrados na seção 3 deste relatório.

Por fim, a seção 4 apresenta as conclusões deste trabalho.

2. Maximização de Influência

Há diversas abordagens para endereçar o problema de maximização de influência. Este trabalho adotou o algoritmo *Community Based Influence Maximization* (CBIM), uma abordagem baseada em identificar "comunidades" de nós, proposta para grafos de múltiplas camadas por Venkatakrishna et al. [14] e adaptado para grafos de camada única.

A Tabela 1 apresenta a notação que será utilizada nesta seção.

Tabela 1: Notação.

Notação	Descrição
V,E	Nós, Arestas no grafo
K	Quantidade de nós semente
δ	Limite de índice de consolidação
$nbrs(v)$	Vizinhos do nó v
γi	Condutância da comunidade i
θi	Escala da comunidade i
ψi	Índice de consolidação de i
KCC	*Katz Centrality Coefficient,* Coeficiente de centralidade de Katz
DSC	*Dice Similarity Coefficient*, Coeficiente de similaridade de Dice
CBIM	*Community Based Influence Maximization*

Para esse algoritmo ser utilizado, arestas entre o mesmo par de nós deverão ser consolidadas em uma única aresta com o peso dado por uma função cujos argumentos são os pesos das arestas originais.

O CBIM procura achar o grupo dos nós mais influentes do grafo, que chamamos de "nós sementes", e a influência desses nós é maximizada através de um processo iterativo. Em cada iteração do algoritmo, nós vizinhos à nós ativos são ativados de acordo com o modelo de propagação. Os algoritmos 1, 2 e 3 descrevem em pseudocódigo o CBIM.

Algoritmo 1 CBIM: Community Based Influence Maximization Framework

1. Grafo G = (*VG* , *EG* , *V*)
2. Identificar as comunidade iniciais
   Cinit ⟵ *FIC*(*M*)
3. Consolidar comunidades pequenas em *CSinit*
   FC ⟵ *CSC*(*CSinit* , *θ*)
4. Achar o coeficiente de centralidade de Katz (*KCC*) de cada nó
5. Calcular a cota de nós sementes por comunidade
6. Identificar os nós sementes das comunidades baseado no fator *KCC* e na cota de nós

Algoritmo 2 Detecção de comunidades

/* **Fase 1 - Detecção de comunidades inicial - FIC(M)** */

1. Inicializar as variáveis *NS* e *CSinit* , grupo de nós e estrutura inicial de comunidades, respectivamente.
   NS ⟵ *V*, *CSinit* ⟵ *φ*
2. Selecionar os nós de maior grau *Dv*
3. Selecionar aleatoriamente um nó *v* de *Dv*
4. Identificar os vizinhos mais similares (*SimNs*) de *v*, usando Dice's neighbour similarity

$$DSC(u,\ v)\ = \frac{2 * |nbrs(u) \cap nbrs(v)|}{|nbrs(u)| + |nbrs(v)|}$$

5. Selecionar aleatoriamente um vizinho *simN* de *SimNs*
6. **Se** *sn* não está em nenhuma comunidade **então**
   7. Criar uma nova comunidade e inserir *v* e *sn*
      K ⟵ |*CSinit*| ; *CK+1* ⟵ {*v, simN*}
   8. Inserir a nova comunidade na estrutura de comunidades
   9. Remover nós *v* e *sn* de *V*
      NS ⟵ *NS* - {*v, simN*}
10. **senão**
    11. Achar a comunidade em que *sn* pertence e denotar-la como *Ck*
        k ⟵ *locate*(*CSinit* , *simN*)
    12. Inserir nó *v* em *Ck* e remover o nó *v* de *NS*
        Ck ⟵ *Ck* ∪ {*v*} *and NS* ⟵ *NS* - {*v*}
13. Repetir os passos 2 ao 12, até que *NS* = *φ*

/* **Fase 2 - Consolidação de comunidades - CSC(CSinit , θ)***/

14. Inicializar comunidades finais (FC)

    $FC \longleftarrow CSinit$

15. Calcular condutância (γi) e escala (θi) para cada comunidade *Ci* em *CSinit*
16. Calcular o índice de consolidação (ψi) para cada comunidade em *CSinit*

    $Ci(\psi) = Ci(\theta) * Ci(\gamma)$

17. Selecionar a comunidade com o menor índice de consolidação (*Cx*)
18. Achar a comunidade mais similar (*Cy*) a (*Cx*) e consolidar as duas comunidades para formar uma nova comunidade (*Cn*)

    $s \longleftarrow argmaxi\ \{Sim(Cx, Cy) | i = 1, 2, ...CS, x = y\}\ and\ Cn = Cx \cup Cy$

19. Calcular o índice de consolidação (ψi) para a nova comunidade (*Cn*)
20. Trocar as duas comunidades *Cx* e *Cy* pela nova comunidade *Cn* na estrutura final de comunidades (*FC*)
21. Repetir os passos 17 ao 20, até que $\psi i > \delta$
22. Retornar *FC*

Na Fase 1 do Algoritmo 2, é identificado comunidades de nós em comum no grafo *G*, a partir do coeficiente de similaridade de Dice (DSC), definido como:

$$DSC(u,\ v)\ = \frac{2 * |nbrs(u) \cap nbrs(v)|}{|nbrs(u)| + |nbrs(v)|}$$

Esse método gera um número grande de comunidades pequenas. Isso não é ideal para os nossos fins, e não cumpre as características fundamentais de uma comunidade. Portanto, é preciso consolidá-las na fase 2.

É importante notar que nos passo 2 é montado uma lista *Dv* de nós com o maior grau do grafo, ou seja, todos os nós de *Dv* possuem graus idênticos. Portanto, no passo seguinte 3, é escolhido aleatoriamente o nó *v* para evitar qualquer tipo de viés no algoritmo. A mesma lógica se aplica para o passo 4, onde todos os vizinhos em *SimNs* possuem o mesmo DSC, e no passo 5 é escolhido um aleatoriamente da lista. Por causa desses aspectos aleatórios, o algoritmo CBIM pode e irá render resultados não idênticos para mesmos valores de parâmetros *k* e δ no mesmo grafo *G*. Se evitar essa aleatoriedade é de interesse, o algoritmo pode ser facilmente alterado para selecionar sempre o primeiro nó de maior grau no passo 2 e sempre o primeiro vizinho de maior DSC no passo 4. Os passos 3 e 5 podem então serem omitidos.

Nas comunidades geradas na fase 1, a soma do peso das arestas dentro da comunidade é menor que a soma do peso das arestas que saem da comunidade. Isso é uma violação das características fundamentais de uma comunidade. Portanto, na fase 2 do algoritmo 2, é consolidado as comunidades identificadas na fase 1.

Essas novas comunidades são a união de duas comunidades menores. É determinado se duas comunidades devem ser mescladas a partir dos seguintes fatores:

- Condutância de comunidade (*γ*): é o fator que representa a "qualidade" de uma comunidade. Uma comunidade de nós densamente conectados e ligados ao resto da rede por poucas arestas é considerada uma comunidade de alta qualidade. Definido como:

$$\gamma(C) = \frac{|Eout|}{2|Ein| + |Eout|}$$

  Onde *Eout* é o grupo de arestas conectando a comunidade C com outras comunidades, e *Ein* é o grupo de arestas presentes dentro da comunidade C.

- Escala de uma comunidade (*θ*): define o quão grande a comunidade é em relação a rede como um todo. Definido como:

$$\theta(C) = \frac{|Vc|}{|V|}$$

  Onde *Vc* é o grupo de nós da comunidade *C*, e *V* é o grupo de nós do grafo *G* inteiro.

- Índice de consolidação de uma comunidade (*ψ*): É o produto da condutância e escala de uma comunidade. Se esse índice for maior que o limite de índice de consolidação (*δ*), a comunidade em questão precisa ser mesclada com outra comunidade. Definido como:

$$\psi(C) = \theta(C) * \gamma(C)$$

Para encontrar o par da comunidade *C*, calculamos a similaridade (*SIM*) entre *C* e as outras comunidades:

$$Sim(Ci,\ Cj) = \frac{\sum_{u \in Ci,\, v \in Cj} DSC(u, v)}{|Cj|}$$

Onde *DSC(u, v)* é o coeficiente de similaridade de Dice entre $u \in Ci$ e $v \in Cj$. Portanto, *SIM(Ci, Cj)* é a soma das similaridades de Dice entre nó *u* em comunidade *Ci* e nó *v* em comunidade *Cj* dividido pela quantidade de nós em *Cj*.

Algoritmo 3 Achando o coeficiente de centralidade de Katz e selecionando os nós semente

---

23. Inicializa o grupo de nós sementes (*SN*) e grupo de comunidades (*CS*)
    $SN \longleftarrow \varphi$, $CS \longleftarrow FC$
24. Calcular o coeficiente de centralidade de Katz (*KCC*) para cada nó do grafo

$$KCC(v) = \alpha \sum_{u} Avu \cdot KCC(u) + \beta$$

25. Ordenar todos os nós do grafo em ordem decrescente de *KCC*
26. Calcular os nós sementes necessários para cada comunidade com uma abordagem baseada em cota

    $Quota(Ci) = k * \frac{ni}{|V|}$

27. Selecionar a cota de nós sementes com maior *KCC* de cada comunidade (*Ci*)

    $SN = SN \cup Quota(Ci)$

28. Retornar *SN*

Depois de identificarmos as comunidades finais, calculamos o coeficiente de centralidade de Katz (*KCC*) para cada nó. Baseado no *KCC*, selecionamos os nós sementes (*SN*) para cada comunidade usando uma cota.

A medida de centralidade de Katz foi escolhida por ser uma medida frequentemente usada para definir a centralidade de um nó em uma rede social, por isso, ela é uma abordagem extensamente provada, eficaz e consistente. Na implementação do algoritmo, foi usado a função **katz_centrality** da biblioteca **NetworkX** para Python. Ela é definida pela seguinte equação matemática:

$$KCC(v) = \alpha \sum_{u} Avu \cdot KCC(u) + \beta$$

Onde *v* e *u* são nós do grafo *G*, α é o fator de atenuação, β é o peso atribuído à vizinhança de *v*, e *A* é a matriz adjacente do grafo *G* com autovalores λ. O parâmetro β controla a centralidade inicial, peso adicional pode ser provido através de β. Conexões feitas entre vizinhos distantes são penalizados pelo fator de atenuação α. Este fator de atenuação deve ser estritamente menor que o inverso do maior autovalor da matriz *A*.

Como o coeficiente *KCC* não é afetado pelos parâmetros *k* e δ do algoritmo CBIM, os valores de *KCC(v)* serão sempre os mesmos para o mesmo nó *v* do grafo *G*.

Os nós sementes *SN* são escolhidos a partir de uma cota para cada comunidade. São escolhidos os nós com os *Quota(C)* maiores *KCC*. A definição para *Quota(C)* é:

$$Quota(C) = k * \frac{n}{|V|}$$

Onde *C* é a comunidade, *K* é a quantidade de nós sementes, *n* é a quantidade de nós na comunidade *C* e *V* é o grupo de nós no grafo *G*, portanto, $|V|$ é o número de nós totais no grafo *G*.

Uma observação importante a ser feita, o valor final de *Quota(C)* precisa ser um número inteiro positivo, para efetivamente representar a quantidade de nós a serem escolhidos para cada comunidade *C*. Por isso, na implementação desse algoritmo foi utilizado a função **round** da biblioteca base do Python. Essa função arredonda o valor decimal para cima ou para baixo, dependendo de qual número inteiro está mais perto do valor real.

Desta forma utilizamos o número mais próximo do valor real de *Quota(C)*, mas a consequência desta abordagem é que ocasionalmente a quantidade de nós sementes que o CBIM define nem sempre será igual ao parâmetro *k* (consultar Experimento 4 e 5 na seção 3.2).

Se for de interesse, o algoritmo pode ser modificado para sempre retornar uma lista de nós sementes de tamanho igual a *k*. Basta omitir o arredondamento de *Quota(C)* e utilizar um contador para limitar a quantidade de loops no passo 4 do Algoritmo 3, para pegar estritamente os nós com os *k* maiores *KCC*.

3. Experimentos

3.1. Implementação do Algoritmo CBIM

O algoritmo CBIM foi implementado em Python 3.10.9 em formato Jupyter Notebook. Utilizando as bibliotecas **numpy** para manipulação de estruturas de dados, **random** para implementação dos aspectos aleatórios do algoritmo, **pandas** para utilização de data frames, **matplotlib.pyplot** para desenhar a estrutura do grafo, e **NetworkX** para manipulação do grafo e cálculo da centralidade de Katz.

A implementação pode ser livremente acessada no repositório do GitHub abaixo:

3.2. Preparação dos Dados

Os dados usados para teste do algoritmo foram gerados aleatoriamente. O grafo foi gerado a partir de uma matriz triangular superior de dimensões 15 x 15, preenchida com

valores de 0 a 3, onde 80% dos valores são 0, 10% dos valores são 1, 5% dos valores são 2 e 5% dos valores são 3. Os valores representam o peso de uma aresta entre as coordenadas da matriz, sendo 0 dois nós sem arestas conectando-os diretamente.

No processo de geração da matriz foi utilizado a função **random.choice** da biblioteca **numpy** para gerar os valores com suas respectivas chances, e a função **triu** da biblioteca **numpy** para formatar a matriz em uma matriz triangular superior.

A matriz então foi convertida para o formato **DataFrame** da biblioteca **Pandas**.

	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0
0.0	0	0	2	0	0	2	0	0	0	1	0	0	0	0	0
1.0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	0
2.0	0	0	0	1	0	0	0	0	0	2	0	1	0	3	0
3.0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
4.0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
5.0	0	0	0	0	0	0	3	0	2	0	0	1	0	0	0
6.0	0	0	0	0	0	0	2	0	0	0	2	0	1	0	1
7.0	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0
8.0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
9.0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
10.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
11.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14.0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Imagem 1: Matriz triangular superior gerada aleatoriamente, utilizada em todos os testes do CBIM.

Então é gerado um grafo a partir do DataFrame, utilizando a biblioteca **NetworkX**. E desse grafo é identificado todas as arestas de auto loop a partir do atributo **selfloop_edges** e removidas do grafo utilizando o método **remove_edges_from** (refira-se a imagem 2).

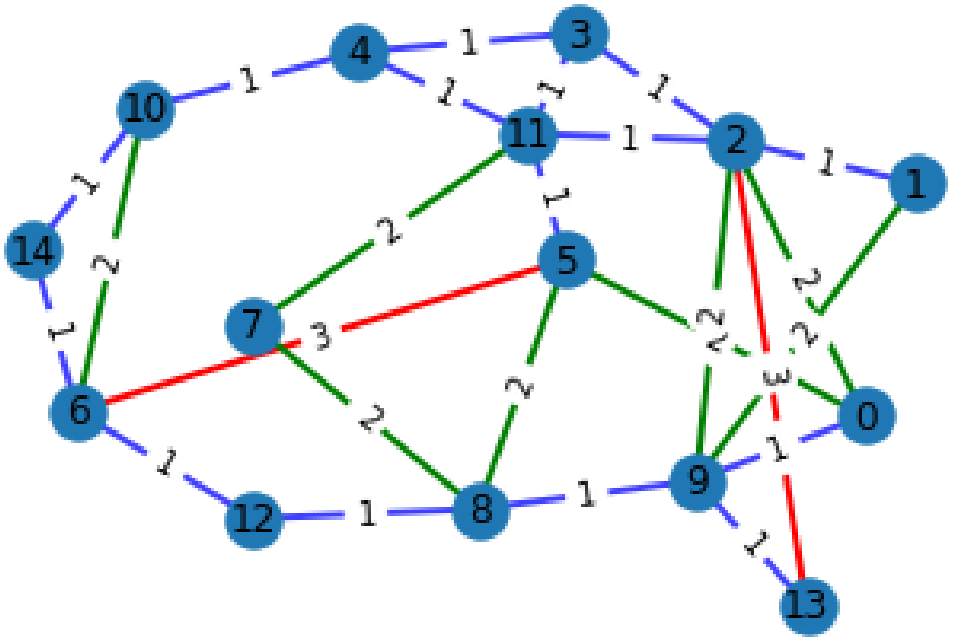

Imagem 2: Representação do grafo utilizado nos testes, depois de ter suas arestas de auto loop removidas. Arestas de peso 1, 2 e 3 são coloridas azul, verde e vermelho, respectivamente.

Tratamento dos Dados

Antes de montar o grafo, foi preciso fazer alguns tratamentos aos dados.

O primeiro passo foi extrair todos os dados não relevantes, nesse caso, entradas de períodos anteriores. A estrutura de Dataframe do Pandas permite o uso de um filtro, que foi usado para facilmente remover essas entradas.

O segundo passo foi extrair quantos alunos únicos estão presentes nos dados restantes do primeiro passo. Para isso, percorremos o Dataframe uma única vez e adicionamos cada aluno novo em um array. Esta operação tem complexidade O(n), sendo *n* o número de linhas no Dataframe.

O terceiro passo foi limpar os dados privados dos alunos. Como estes dados são reais, eles possuem as matrículas reais dos alunos, por isso, as matrículas foram trocadas por números incrementais começando por 0. Para isso, foi usado a propriedade *loc* do Dataframe, novamente com um filtro, para substituir todas as entradas do aluno x_0 por 0, x_1 por 1, e assim por diante. Esta operação tem complexidade O(n), sendo *n* o número de alunos *únicos*.

Depois de limpar os dados, podemos começar a montar o grafo.

Um novo Dataframe é criado. Este é o Dataframe que será usado de fato para montar o grafo. Ele consiste de uma matriz *n* por *n*, onde *n* é a quantidade de alunos únicos, extraída do segundo passo no tratamento dos dados. Cada coordenada (x,y) da matriz representa quantas turmas os alunos *x* e *y* possuem em comum.

O processo de montar essa matriz é simples, porém custoso. Percorremos o dataframe, e para cada entrada, percorremos todas as entradas com a mesma disciplina e turma. Este processo é mais eficiente que força bruta em casos que não o pior caso, mas a complexidade ainda é $O(n^2)$.

3.3. Resultados

É importante levar em conta que o algoritmo CBIM possui alguns aspectos aleatórios. Portanto, os resultados não serão sempre idênticos para os mesmos valores de *k* e δ no mesmo grafo *G*.

O coeficiente de centralidade de Katz (*KCC*) não depende dos parâmetros *k* e δ, apenas da estrutura do grafo *G*. Por isso, os valores serão os mesmos para todos os experimentos abaixo:

- Coeficientes de centralidade de Katz de cada nó em ordem decrescente (Nó, *KCC*):

```
(2, 0.378888594901617),
(5, 0.3391924104931806),
(9, 0.313881768000951),
(6, 0.2893644006798845),
(0, 0.28828297191196767),
(8, 0.2737529325667632),
(11, 0.2669748049920425),
(13, 0.2583341614238882),
(7, 0.22142852473865923),
(1, 0.2139469795791596),
(10, 0.20544541207179415),
(3, 0.19588143366680033),
(4, 0.18011507730433252),
(12, 0.16959657065829048),
(14, 0.16276640416117097)
```

Experimento 1: Grafo *G* com *k* = 4 e δ = 0.1

Para o Experimento 1 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 11, 3, 4], [6, 14, 10], [8, 7, 12]`

- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1, 5, 11, 3, 4], [6, 14, 10], [8, 7, 12]`

- Cotas para cada comunidade:
  `[2, 1, 1]`

- Nós Semente (*SN*):
  `[2, 5, 6, 8]`

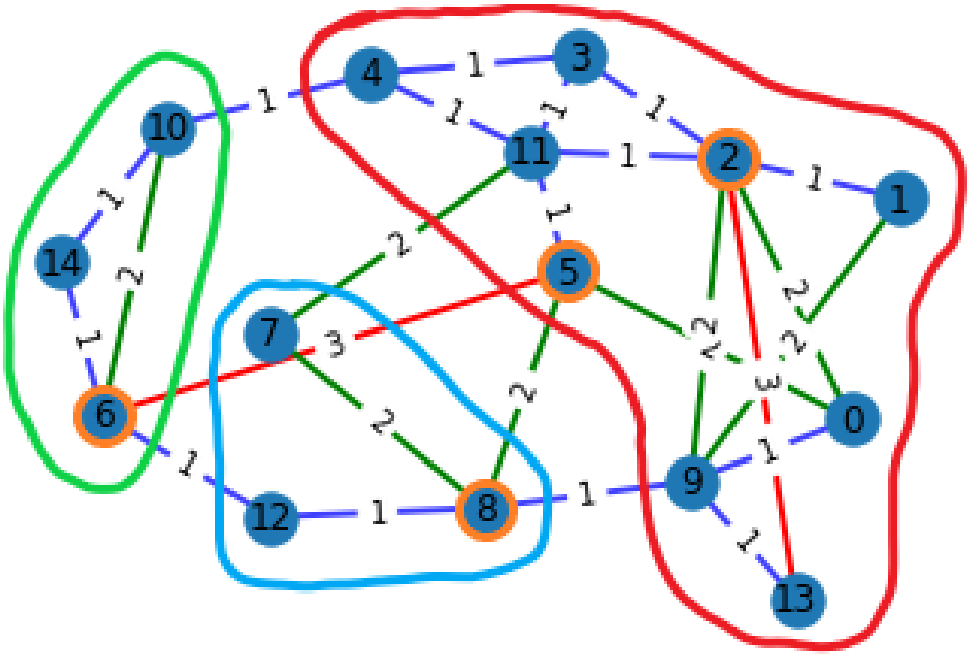

Imagem 3: Representação do resultado do Experimento 1. Os grupos de nós circulados em vermelho, verde e azul representam as comunidades 0, 1 e 2, respectivamente. Os nós circulados em laranja são os nós sementes.

As comunidades 0, 1 e 2 estão circuladas em vermelho, azul e verde (observar a Imagem 3 acima). Esta notação persiste para os seguintes experimentos, quando houver mais de 3 comunidades, elas serão representadas pelas cores amarelo e lilás.

Pode-se observar que a comunidade 0 é a maior comunidade entre elas, com uma soma de peso de arestas internas igual a 20 e peso das arestas externas igual a 6. Uma característica fundamental de uma comunidade é que a soma do peso de suas arestas internas é maior ou igual a soma do peso de suas arestas externas. As comunidades 1 e 2 não cumprem essa característica.

Conforme o comparamos com os seguintes experimentos, nós observamos que o parâmetro δ diretamente influencia a quantidade e o tamanho das comunidades finais, que

por sua vez, influencia quais nós são escolhidos para serem nós sementes. O parâmetro *k* dita quantos nós sementes serão escolhidos, mas não é estritamente a quantidade final de nós sementes, devido a fatores aleatórios e arredondamentos no algoritmo.

Experimento 2: Grafo *G* com *k* = 4 e *δ* = 0.4

Para o experimento 2 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 8, 7], [6, 14, 10, 12], [11, 3, 4]`
- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1, 11, 3, 4, 5, 8, 7], [6, 14, 10, 12]`
- Cotas para cada comunidade:
  `[3, 1]`
- Nós Semente (*SN*):
  `[2, 5, 9, 6]`

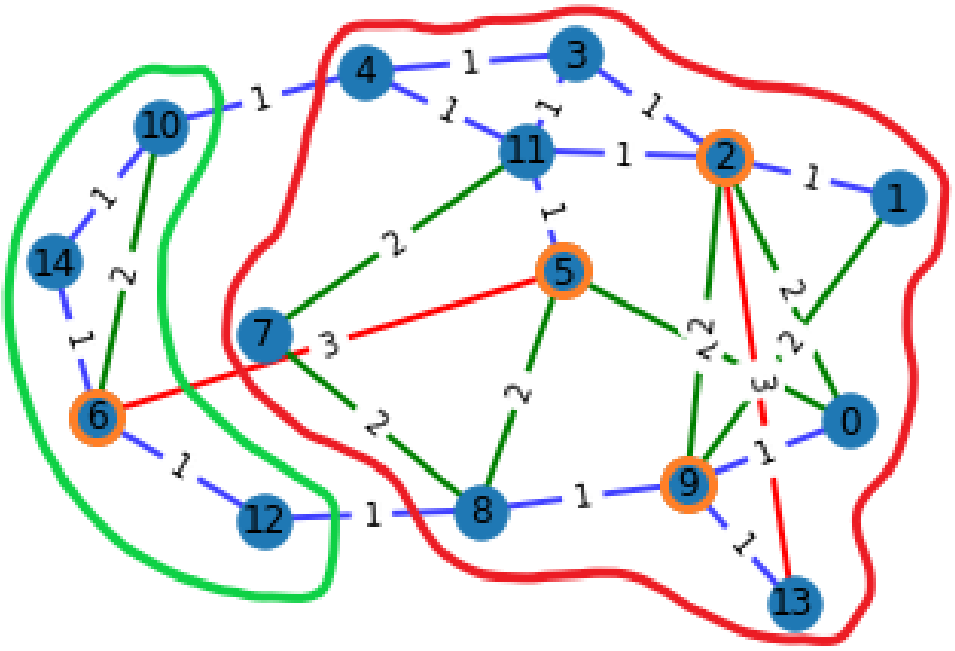

Imagem 4: Representação do resultado do Experimento 2. Os grupos de nós circulados em vermelho e verde representam as comunidades 0 e 1, respectivamente. Os nós circulados em laranja são os nós sementes.

Neste experimento o parâmetro *k* foi mantido o mesmo, e o parâmetro *δ* foi aumentado para 0,4. O aumento do parâmetro *δ* foi feito com o objetivo de formar comunidades que cumprem a característica fundamental de uma comunidade.

Podemos observar que desta vez foram geradas apenas 2 comunidades grandes, mas de maior qualidade. Ambas cumprem o requisito de uma quantidade maior ou igual de arestas internas a arestas externas. A comunidade 0 possui um peso interno total de 25 e externo de 5, enquanto a comunidade 1 possui peso interno e externo de 5. Um resultado mais satisfatório.

Os nós sementes escolhidos também mudaram. Os nós 2, 9 e 6 se mantiveram na lista, mas o nó 8 foi substituído pelo nó 9, de maior *KCC*.

Experimento 3: Grafo *G* com *k* = 4 e *δ* = 0.07

Para o experimento 2 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 8, 7], [6, 14, 10, 12], [11, 3, 4]`
- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1, 11, 3, 4], [5, 8, 7], [6, 14, 10, 12]`
- Cotas para cada comunidade:
  `[2, 1, 1]`
- Nós Semente (*SN*):
  `[2, 9, 5, 6]`

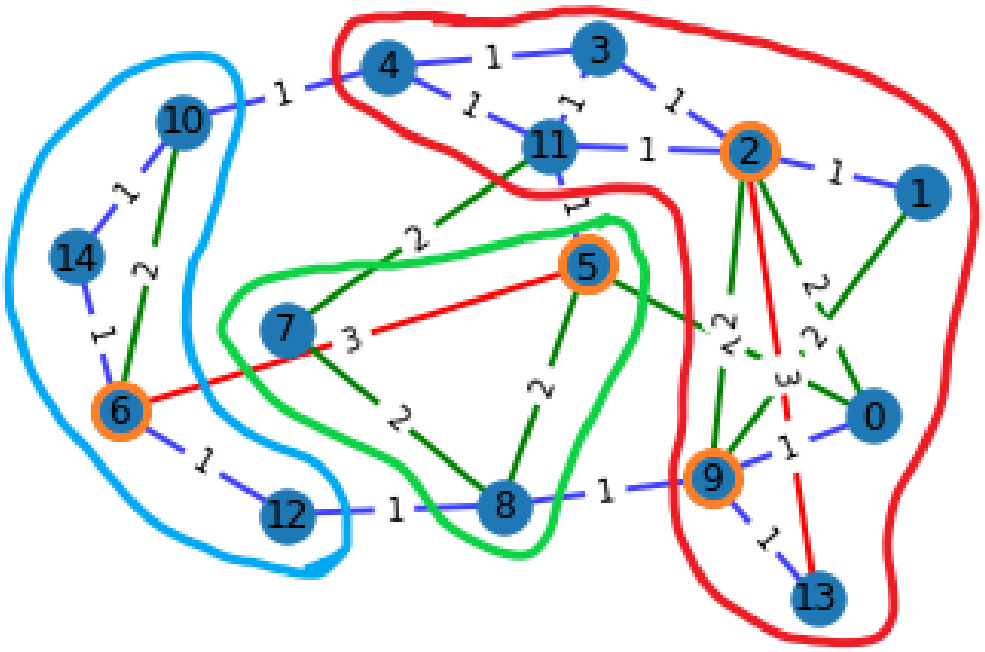

Imagem 5: Representação do resultado do Experimento 3. Os grupos de nós circulados em vermelho, verde e azul representam as comunidades 0, 1 e 2, respectivamente. Os nós circulados em laranja são os nós sementes.

Neste experimento, mantivemos o valor de 4 de *k*, e diminuímos o valor de δ para 0,07, um valor menor ainda do que o usado no Experimento 1.

O resultado é esperado, podemos observar que quanto menor o valor de δ, mais comunidades recebemos no resultado final. Isso é devido à fase de consolidação de comunidade do algoritmo CBIM, o parâmetro δ dita quantas iterações dessa fase o algoritmo irá executar, menos iterações resulta em menos consolidação de comunidades e, portanto, mais comunidades de menor tamanho e qualidade.

O interessante a ser observado nesse experimento, é que a lista de nós sementes se manteve idêntica ao Experimento 2. Talvez seja necessário um grafo maior para observarmos o impacto nos nós sementes com mais consistência.

Experimento 4: Grafo *G* com *k* = 4 e δ = 0.02

Para o experimento 3 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 6, 12], [8, 7], [11, 3, 4], [10, 14]`

- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1], [5, 6, 12], [8, 7], [11, 3, 4], [10, 14]`

- Cotas para cada comunidade:
  `[1, 1, 1, 1, 1]`

- Nós Semente (*SN*):
  `[2, 5, 8, 11, 10]`

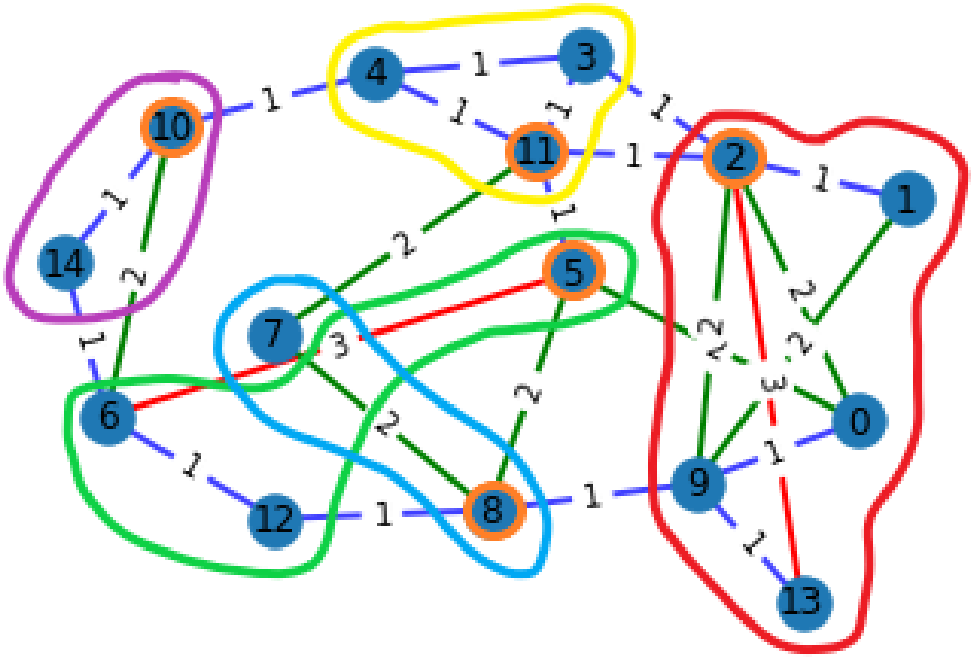

Imagem 6: Representação do resultado do Experimento 4. Os grupos de nós circulados em vermelho, verde, azul, amarelo e lilás representam as comunidades 0, 1, 2, 3 e 4, respectivamente. Os nós circulados em laranja são os nós sementes.

Neste experimento abaixamos o parâmetro δ ao extremo para o valor de 0,02. O valor de *k* foi mantido o mesmo.

Pode-se observar que o resultado de abaixar o valor de δ é obtermos muitas comunidades de pequeno tamanho. Um pulo de 3 comunidades para 5, do Experimento 4. Estas comunidades são de qualidade baixa, mas a comunidade 0 se mantém similar às suas reencarnações passadas, certamente devido à alta conectividade entre os nós 0, 1, 2, 9 e 13.

Os nós sementes também mudaram do último experimento, com os nós 2 e 5 se mantendo na lista, provavelmente devido a seu alto *KCC.*

Experimento 5: Grafo *G* com *k* = 2 e δ = 0.1

Para o experimento 3 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 8, 12], [6, 14, 10], [11, 3, 7, 4]`

- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1, 11, 3, 7, 4], [5, 8, 12], [6, 14, 10]`

- Cotas para cada comunidade:
  `[1, 0, 0]`

- Nós Semente (*SN*):
  `[2]`

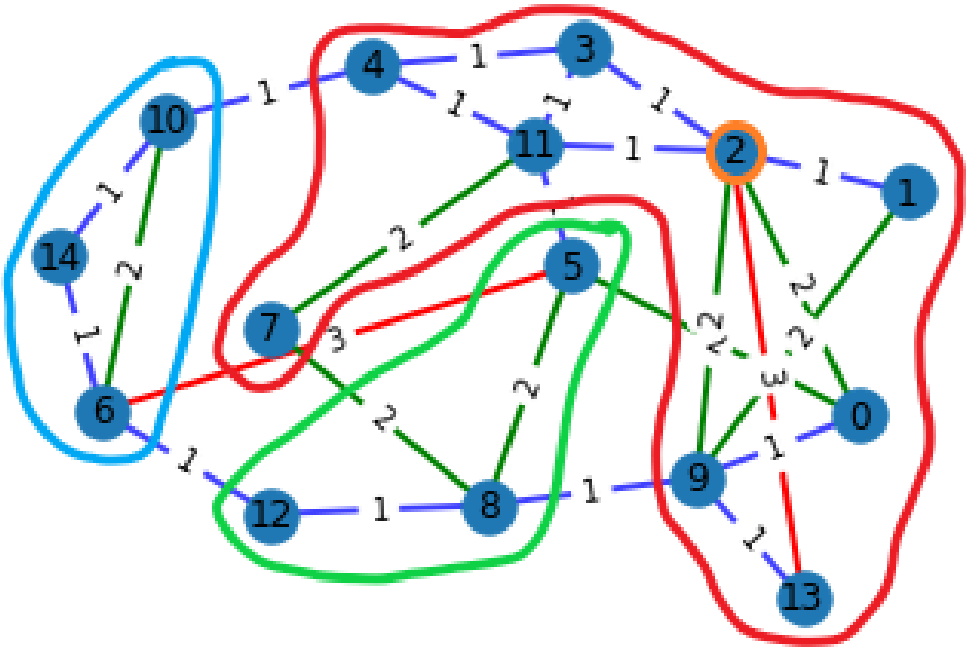

Imagem 7: Representação do resultado do Experimento 5. Os grupos de nós circulados em vermelho, verde e azul representam as comunidades 0, 1 e 2, respectivamente. Os nós circulados em laranja são os nós sementes.

Foi utilizado o valor de 0,1 novamente para o parâmetro δ, desta vez com o valor novo de 2 para *k*.

Encontramos um grupo de comunidades parecido com o Experimento 1, como esperado, devido ao valor de δ. Mas desta vez temos um único nó semente. Este resultado é interessante pois utilizamos *k* = 2, mas recebemos apenas um nó. Isto demonstra as consequências de se utilizar arredondamento para a cota de nós no algoritmo.

Experimento 6: Grafo *G* com *k* = 2 e δ = 0.4

Para o experimento 3 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 13, 1], [5, 0, 8, 7], [6, 14, 10, 12], [11, 3, 4]`

- Comunidades finais (*FC*):
  `[2, 9, 13, 1, 5, 0, 8, 7, 11, 3, 4], [6, 14, 10, 12]`

- Cotas para cada comunidade:
  `[1, 1]`

- Nós Semente (*SN*):
  `[2, 6]`

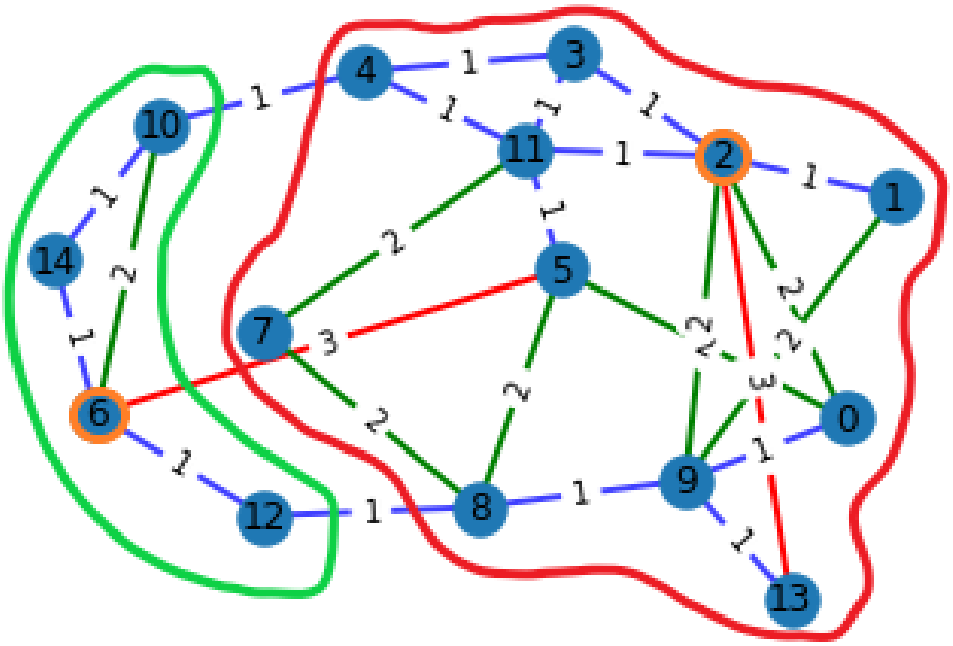

Imagem 8: Representação do resultado do Experimento 6. Os grupos de nós circulados em vermelho e verde representam as comunidades 0 e 1, respectivamente. Os nós circulados em laranja são os nós sementes.

Neste experimento, tentamos replicar os resultados do Experimento 2. Foi utilizado o valor de 2 para *k* e 0,4 para δ.

Observamos que foram geradas duas comunidades grandes, como esperado. E os dois nós com maior *KCC* para cada comunidade foram escolhidos para nós semente.

Experimento 7: Grafo *G* com *k* = 2 e δ = 0.07

Para o experimento 3 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 13, 1], [5, 0], [6, 14, 10, 12], [8, 7], [11, 3, 4]`

- Comunidades finais (*FC*):
  `[2, 9, 13, 1, 5, 0], [6, 14, 10, 12], [8, 7], [11, 3, 4]`

- Cotas para cada comunidade:
  `[1, 1, 0, 0]`

- Nós Semente (*SN*):
  `[2, 6]`

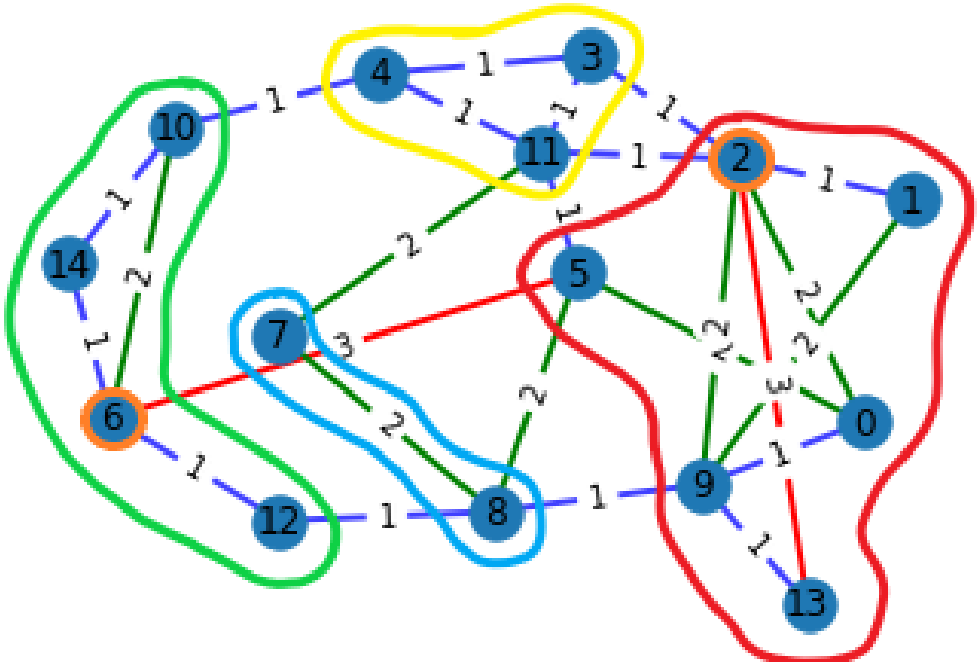

Imagem 9: Representação do resultado do Experimento 7. Os grupos de nós circulados em vermelho, verde, azul e amarelo representam as comunidades 0, 1, 2 e 3, respectivamente. Os nós circulados em laranja são os nós sementes.

Neste experimento voltamos para valores baixos de 0,07 para δ e 2 para k.

Desta vez o algoritmo gerou uma comunidade a mais que na última vez que utilizamos δ = 0,07, no Experimento 3. Como esperado, estas comunidades são pequenas e de baixa qualidade, especialmente as comunidades 2 e 3 (azul e amarelo, respectivamente), que não tiveram nenhum nó escolhido para nó semente.

Experimento 8: Grafo *G* com k = 2 e δ = 0.02

Para o experimento 3 foram obtidos os seguintes resultados:

- Comunidades iniciais (*CSinit*):
  `[2, 9, 0, 13, 1], [5, 8, 12], [6, 14, 10], [11, 3, 7, 4]`

- Comunidades finais (*FC*):
  `[2, 9, 0, 13, 1], [5, 8, 12], [6, 14, 10], [11, 3, 7, 4]`

- Cotas para cada comunidade:
  `[1, 0, 0, 1]`

- Nós Semente (*SN*):
  `[2, 11]`

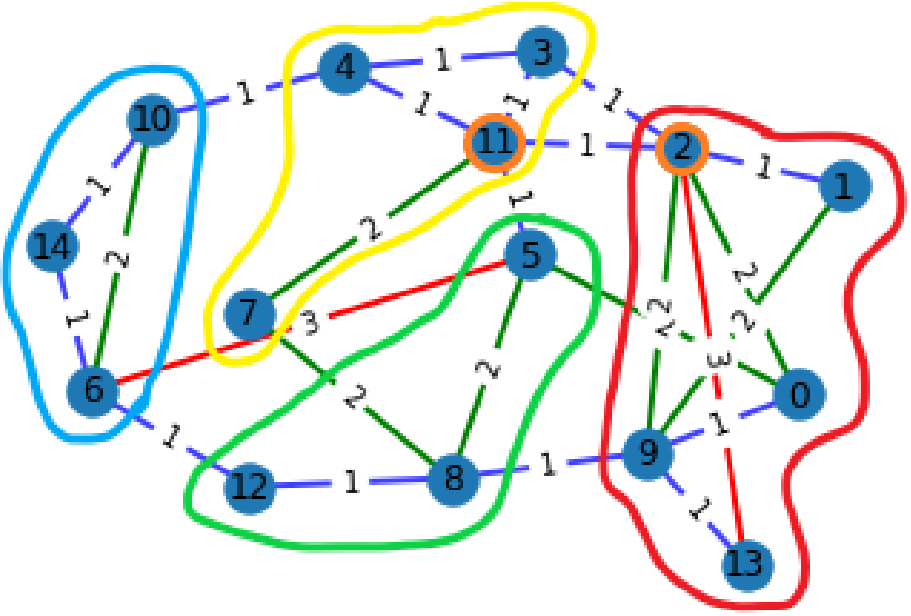

Imagem 10: Representação do resultado do Experimento 8. Os grupos de nós circulados em vermelho, verde, azul e amarelo representam as comunidades 0, 1, 2 e 3, respectivamente. Os nós circulados em laranja são os nós sementes.

No último experimento repetimos o valor baixo de 0,02 para δ e 4 para *k*.

Dessa vez temos quatro comunidades, sendo as comunidades 1, 2 e 3 de baixa qualidade. Curiosamente, o nó 11 foi escolhido como nó semente neste experimento, provavelmente devido à qualidade maior da comunidade 3 em comparação às comunidades 1 e 2.

Como esperado, o nó 2 foi escolhido como nó semente em todos os experimentos feitos, e também estava sempre presente na comunidade de maior qualidade, certamente por ser o nó com o maior *KCC* do grafo *G* inteiro.

Outra observação interessante, é que o nó 2 também é o nó de maior *grau* do grafo *G*, podemos inferir então que o *KCC* está diretamente ligado ao grau de um nó? É preciso mais testes com um grafo mais robusto.

4. Conclusões

Neste projeto abordamos o problema de maximização de influência (*Influence Maximization* – IM) utilizando uma variação do algoritmo *Community Based Influence Maximization* (CBIM).

Foram relatados experimentos com o CBIM em diversos grafos, explorando diferentes combinações de valores de k e δ. Os resultados foram interessantes, e a implementação do algoritmo está a livre acesso para outros acadêmicos utilizarem.

Foi utilizado um grafo de pequena escala, não direcionado, com peso nas arestas, para melhor visualizar o passo-a-passo do algoritmo, com imagens acompanhando.

Porém, é importante observar que os testes feitos neste trabalho são limitados e que há muitos outros cenários a serem ainda replicados com o CBIM, especialmente grafos com maior quantidade de nós. Mas também, grafos de múltiplas camadas, direcionados, cíclicos, com e sem peso nas arestas.

CBIM é um algoritmo de alta eficiência. CBIM seleciona nós sementes mais influentes que outros algoritmos, e a densidade de arestas das comunidades geradas pelo CBIM também é maior que a de outros algoritmos, resultando em comunidades de maior qualidade. No futuro, este algoritmo pode ser utilizado para análise de influência em diversas redes sociais complexas, como Twitter ou Facebook, e pode servir como base para algoritmos de recomendação a usuários.

O próximo passo deste trabalho seria escalar os testes para grande volume de dados reais, aplicando também modelos de difusão, como *Independent Cascade* (IC) e *Linear Threshold* (LT). E fazer uma análise de performance mais profunda, em quesito de velocidade e qualidade das comunidades e nós sementes.

5. Referências Bibliográficas

[1] Aggarwal, C. (ed.) (2011) *Social Network Data Analytics*. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-8462-3_1 (Cited by 915) https://link.springer.com/content/pdf/10.1007%2F978-1-4419-8462-3.pdf overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery

[2] Brilhante, I., Macedo, J.A., Nardini, F.M., Perego, R., Renso, C. (2014) TripBuilder: A Tool for Recommending Sightseeing Tours. In: de Rijke M. et al. (eds) Advances in Information Retrieval. ECIR 2014. Lecture Notes in Computer Science, vol 8416. Springer, Cham. https://doi.org/10.1007/978-3-319-06028-6_93

[3] Cai, F., Qiu, L., Kuai, X., Zhao, H., "CBIM-RSRW: An Community-Based Method for Influence Maximization in Social Network," in IEEE Access, vol. 7, pp. 152115-152125, 2019, doi: 10.1109/ACCESS.2019.2944350.

[4] Cinelli, M., Quattrociocchi, W., Galeazzi, A. et al. The COVID-19 social media infodemic. Sci Rep 10, 16598 (2020). https://doi.org/10.1038/s41598-020-73510-5

[5] Dijkstra, J. Social exchange: relations and networks. Soc. Netw. Anal. Min. 5, 60 (2015). https://doi.org/10.1007/s13278-015-0301-1

[6] Guidotti, R., Monreale, A., Rinzivillo, S. et al. Unveiling mobility complexity through complex network analysis. Soc. Netw. Anal. Min. 6, 59 (2016). https://doi.org/10.1007/s13278-016-0369-2

[7] Guilbeault, D. et al., 'Complex Contagions: A Decade in Review', in Lehmann S. and Ahn Y. (eds.), Spreading Dynamics in Social Systems (Springer Nature, 2018).

[8] Kucharski, A. The Rules of Contagion (Wellcome Collection) (p. 277). Profile. Kindle Edition.

[9] Marcoux, T., Galeano, K., Galeano, R. et al. A public online resource to track COVID-19 misinfodemic. Soc. Netw. Anal. Min. 11, 45 (2021). https://doi.org/10.1007/s13278-021-00748-w

[10] Martino, F., Spoto, A. Social Network Analysis: A brief theoretical review and further perspectives in the study of Information Technology. PsychNology Journal 4(1):53-86, January 2006.

[11] Neo4J: Graph Database Platform. https://neo4j.com/. Acesso em Setembro/2021

[12] Smith, M.A., Shneiderman, B., Milic-Frayling, N., Rodrigues, E.M., Barash, V., Dunne, C., Capone, T., Perer, A., Gleave, E. 2009. Analyzing (social media) networks with NodeXL. In Proceedings of the fourth international conference on Communities

and technologies (C&T '09). Association for Computing Machinery, New York, NY, USA, 255–264. DOI:https://doi.org/10.1145/1556460.1556497

[13] Tabassum, S., Pereira, F.S.F., Fernandes, S., Gama, J. Social network analysis: An 8(5):e1256 (April 2018). DOI: 10.1002/widm.1256Springer, Cham. https://doi.org/10.1007/978-3-319-06028-6_93

[14] Venkatakrishna, R.K, Chowdary, C.R. CBIM: Community-based Influence Maximization in Multilayer Networks (2021). Submitted for publication.