Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise e aplicação de algoritmos de detecção de comunidades em grafos de relacionamento entre os senadores do Brasil no Twitter

Victor Ferreira Gonçalves

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Thiago de Paulo Faleiros

Brasília
2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise e aplicação de algoritmos de detecção de comunidades em grafos de relacionamento entre os senadores do Brasil no Twitter

Victor Ferreira Gonçalves

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Thiago de Paulo Faleiros (Orientador)
CIC/UnB

Prof. Dr. Luis Paulo Faina Garcia
Universidade de Brasília

Paulo Eduardo Althoff
Universidade de Brasília

Prof. Dr. Joao Jose Costa Gondim
Coordenador do Curso de Engenharia da Computação

Brasília, 8 de março de 2022

Dedicatória

Dedico esse trabalho a meus pais, amigos e professores que fizeram parte dessa minha jornada na UnB.

Agradecimentos

Este trabalho foi desenvolvido como Projeto de Final de curso de graduação em Engenharia de Computação da Universidade de Brasília. Agradeço ao meu orientador e professor Thiago Faleiros por ter me dado todo o suporte e sugestões necessárias para a conclusão deste trabalho, agradeço também ao professor Pedro Robson da ciência política por ter contribuído na parte política do trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

O objetivo deste trabalho é analisar diferentes algoritmos de detecção de comunidades em grafos feitos com base no relacionamento entre os senadores do Brasil usando como peso das arestas as semelhanças dos grupos de seguindos(usuários que os senadores seguem) desses senadores na rede social Twitter. Três métricas de análise de comunidades serão usadas para avaliar cada resultado achado pelos algoritmos para assim se ter uma avaliação de como cada algoritmo desempenhou. Do ponto de vista político espera-se obter comunidades de pessoas que possuem semelhanças ideológicas. Com um segundo intuito de analisar a consistência dos resultados dos algoritmos, também será feita uma comparação dos algoritmos aplicando eles em 4 grafos gerados aleatoriamente com diferentes números de arestas intercomunitárias e intracomunitárias. O trabalho analisa os algoritmos de detecção de comunidades: Greedy Modularity Maximization, Kernighan-Lin, Girvan-Newman, Fluid Communities e Markov Cluster. As métricas de avaliação de comunidades utilizadas são: Modularidade, Cobertura e Desempenho.

Palavras-chave: Grafo, Comunidade, Senador, Aresta, Algoritmo, Vértice, Twitter

Abstract

The objective of this work is to analyze different algorithms of community detection on graphs created based on relationship of Brazil senators using edge weights as the similarities between the senators's followings on Twitter. Three algorithms of community analysis will be used to evaluate each result found by the detection algorithms to evaluate how each of them has performed. From a political point of view is expected to find communities with people who have ideological similarities.

With a second objective of analyzing the consistency of the algorithms results, a comparison, applying them on 4 random generated graphs with different number of inter-cluster and intra-cluster edges will be done.

This work analyzes the following community detection algorithms: Greedy Modularity Maximization, Kernighan-Lin, Girvan-Newman, Fluid Communities and Markov Cluster. The communities evaluation algorithms used are: Modularity, Coverage and Performance.

Keywords: Graph, Community, Senator, Edge, Algorithm, Node, Twitter

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Os primeiros passos da teoria de redes encontram-se principalmente nos trabalhos do matemático Euler que criou os primeiros resultados da teoria dos grafos [5]. Um grafo é uma representação de um conjunto de nós conectados por arestas que, em conjunto, formam uma rede. Existem exemplos, nos diversos domínios de redes: redes de comunicação, redes sociais, redes biológicas e redes de informação. A organização de uma rede social compreende a totalidade de relações de um determinado agrupamento social. Neste sentido, pode-se dizer que a organização é composta pela interação social que constitui as relações de determinado grupo.

Uma das linhas de pesquisa na teoria de redes é a detecção de comunidades por algoritmos. O objetivo da detecção de comunidades é identificar os grupos utilizando somente a informação contida na topologia do grafo. Esse problema já é antigo e os primeiros trabalhos sobre comunidades em grafos datam de mais de 50 anos atrás. Um trabalho proposto por Girvan e Newman [6], onde eles apresentam um novo algoritmo de detecção de comunidades, despertou uma grande atividade na área, e vários novos métodos foram desenvolvidos recentemente, com várias contribuições na ciência. [7]

Pode-se perceber um grande crescimento do número de redes sociais obtidas a partir de vários modelos de interação humana e social. Redes sociais como Facebook, Linkedin, Instagram, Twitter, entre outras, mostram a atual tendência de organização e comunicação entre as pessoas.[5] O Twitter é bastante utilizado por políticos e todos os senadores possuem uma conta na rede, em decorrência disso, este trabalho usa as contas de quem os senadores seguem como forma de tentar agrupar os senadores em grupos. Com a separação de senadores em comunidades, espera-se poder identificar senadores que possuem posicionamentos parecidos e a possibilidade de identificar os diferentes grupos existentes no senado brasileiro.

1.2 Objetivo

Uma das grandes dificuldades na detecção de comunidades é identificar a melhor representação da relação entre os indivíduos a ser descrita pelas arestas no grafo, ou seja, qual aspecto dos relacionamentos interpessoais deve ser considerado para encontrar resultados significativos na detecção de comunidades e o que essas comunidades detectadas irão representar. Pensando nisso, esse trabalho visa analisar as comunidades encontradas nos grafos criados com o intuito de avaliar a qualidade dessas comunidades e dos algoritmos utilizados para se concluir qual desempenhou melhor. Os grafos serão criados usando a semelhança nos seguindos(usuários que os senadores seguem) dos senadores do Twitter, por isso é importante ter conhecimento de que muitos políticos costumam seguir grandes rivais para ter conhecimento de suas ações, e isso interfere no momento de separar comunidades com preferências políticas parecidas.

Como forma de avaliar melhor os algoritmos selecionados, cada algoritmo será testado em 4 grafos gerados aleatoriamente com diferentes números de arestas intercomunitárias e intracomunitárias, para assim ser possível avaliar a consistência dos resultados de cada algoritmo quando utilizados em diferentes grafos.

1.3 Organização da monografia

O restante da monografia está organizada da seguinte maneira: O Capítulo 2 apresenta uma introdução aos algoritmos utilizados neste trabalho e uma explicação dos principais conceitos sobre grafos. O Capítulo 3 apresenta os trabalhos com temas parecidos que foram revisados. O Capítulo 4 mostra como foram feitos os procedimentos para a construção dos grafos, aplicação dos algoritmos e comparação dos resultados. O Capítulo 5 apresenta a análise dos resultados obtidos pela aplicação de 5 algoritmos nos grafos criados. O Capítulo 6 expõe as conclusões finais acerca dos resultados obtidos.

Capítulo 2

Fundamentação Teórica

2.1 Grafos

É uma estrutura composta por um conjunto (não vazio) de pontos (vértices) e um conjunto de linhas que ligam esses pontos (arestas). Grafos são importantes modelos para uma grande variedade de problemas de engenharia, computação, matemática, economia, biologia, etc. [8]

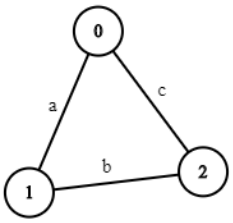


Figura 2.1: Grafo com 3 vértices 0, 1, 2 e 3 arestas a, b, c.

2.1.1 Grafo Simples

Um grafo simples é um grafo que não contém aresta cujos extremos são iguais(laços) nem arestas que possuem os mesmos extremos (arestas múltiplas). [8]

2.1.2 Grafo Direcionado

Um grafo dirigido ou digrafo ou direcionado G consiste de dois conjuntos finitos:
1. Vértices $V(G)$.

2. Arestas dirigidas $E(G)$, onde cada aresta é associada a um par ordenado de vértices chamados de nós terminais. Se a aresta e é associada ao par (u, v) de vértices, diz-se que e é a aresta dirigida de u para v. [8]

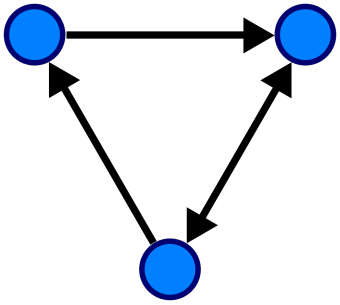


Figura 2.2: Grafo Dirigido.

2.1.3 Grafo valorado

Um grafo valorado é um grafo em que cada aresta tem um valor associado que é chamado de peso. Formalmente, um grafo valorado $G = (V, E)$ consiste de um conjunto V de vértices, um conjunto E de arestas, e uma função f de E para P, onde P representa o conjunto de valores (pesos) associados às arestas. [8]

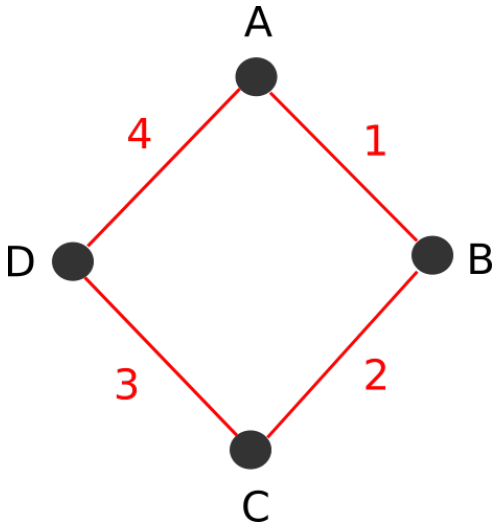


Figura 2.3: Grafo com peso nas arestas.

2.1.4 Grafo Ciclo

Em teoria dos grafos um grafo ciclo ou grafo circular é um grafo que consiste de um único ciclo, ou em outras palavras, um número de vértices conectados em uma rede fechada. O grafo ciclo com n vértices é chamado Cn. O número de vértices em um Cn se iguala ao número de arestas, e cada vértice tem grau 2; isto é, cada vértice tem exatamente duas arestas incidentes a ele. [8]

2.1.5 Subgrafo

Um subgrafo de um grafo é uma "parte"do grafo. Um grafo $H = (V\text{t}, E\text{t})$ é dito ser um subgrafo de um grafo $G = (V, E)$ se:
– cada vértice de H é também um vértice de G, ou seja, $V\text{t} \subseteq V$;
– cada aresta de H é também uma aresta de G, ou seja, $E\text{t} \subseteq E$; e
– cada aresta de H tem os mesmos nós terminais em G, ou seja, se $(u, v) \in E\text{t}$ então $(u, v) \in E$. [8]

2.1.6 Grau de um vértice

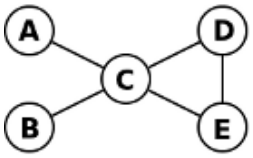
Seja G um grafo e um vértice v de G. O grau de v, denominado $grau(v)$ ou $deg(v)$, é igual ao número de arestas que são incidentes a v, com uma aresta que seja um laço contada duas vezes. O grau total de G é a soma dos graus de todos os vértices de G. [8]

2.2 Representando grafos

Existem muitas maneiras de se representar grafos, cada uma com suas vantagens e desvantagens. Algumas situações ou algoritmos nos quais queremos usar grafos como entradas pedem por uma representação, e outros pedem outra diferente. As representações mais usadas são Matriz de adjacência e Lista de adjacência. [8]

2.2.1 Matriz de adjacência

A matriz de adjacência de um grafo com $|V|$ vértices é uma matriz $|V| \times |V|$ de zeros e uns, na qual a entrada na linha i e coluna j é 1 se e somente se a aresta (i, j) estiver no grafo. Para indicar o peso da aresta, coloca-se o peso na entrada da linha i, coluna j, e um valor especial (vazio) para indicar uma aresta ausente. [8]



	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

Figura 2.4: exemplo de grafo e sua matriz de adjacência.

2.2.2 Matriz de transição

A matriz de transição de um grafo com $|V|$ vértices é uma matriz $|V| \times |V|$, na qual a entrada na linha i e coluna j é igual a probabilidade do vértice i escolher a aresta que chegará ao vértice j, no caso de um vértice com 4 arestas por exemplo, a probabilidade seria 1/4 para todas as arestas(sem considerar o peso). [8]

2.2.3 Lista de adjacência

A representação de um grafo com listas de adjacência combina as matrizes de adjacência com as listas de arestas. Armazena-se um arranjo dos vértices adjacentes a cada vértice i. Tipicamente, temos um arranjo de $|V|$ listas de adjacência, uma lista por vértice. [8]

2.3 Conectividade de um grafo

Informalmente um grafo é conexo (conectado) se for possível caminhar de qualquer vértice para qualquer outro vértice através de uma sequência de arestas adjacentes. [8]

Figura 2.5: Grafo conexo e desconexo.

2.3.1 Componente conexo

Um grafo H é um componente conexo de um grafo G se:
1. H é um subgrafo de G;
2. H é conexo;

3. Nenhum subgrafo conexo I de G tem H como um subgrafo e I contém vértices ou arestas que não estão em H.
Um grafo pode ser visto como a união de seus componentes conexos. [8]

2.3.2 Componente fortemente conexo

Um grafo dirigido $G = (V, E)$ é fortemente conexo se cada dois vértices quaisquer são alcançáveis a partir um do outro.
Os componentes fortemente conexos de um grafo dirigido são conjuntos de vértices sob a relação "são mutuamente alcançáveis".
Um grafo dirigido fortemente conexo tem apenas um componente fortemente conexo. [8]

2.3.3 Centralidade de intermediação

A centralidade de intermediação de uma aresta é a soma da fração de todos os menores caminhos de pares de vértices que passam por essa aresta. A centralidade de um aresta "e" é definida em (1) como:

$$Centralidade = \sum_{s,t \in V} \frac{\sigma(s,t|e)}{\sigma(s,t)} \qquad (1)$$

Onde V é o par de vértices, $\sigma(s,t)$ é o número de caminhos mais curtos de (s,t), e $\sigma(s,t|e)$ é o número desses caminhos que passam pela aresta "e". [9]

2.4 Árvore

Uma árvore (também chamada de árvore livre) é um grafo não dirigido acíclico e conexo.

2.4.1 Árvore geradora

Uma árvore geradora de um grafo G é um grafo que contém cada vértice de G e é uma árvore.

- Proposição:

– Cada grafo conexo tem uma árvore geradora.
– Duas árvores geradores quaisquer de um grafo têm a mesma quantidade de arestas. [8]

2.4.2 Árvore geradora mínima

Um grafo com peso é um grafo onde cada aresta possui um peso representado por um número real. A soma de todos os pesos de todas as arestas é o peso total do grafo. Uma

árvore geradora mínima para um grafo com peso é uma árvore geradora que tem o menor peso total possível dentre todas as possíveis árvores geradoras do grafo. Se G é um grafo com peso e e uma aresta de G então:
– $w(e)$ indica o peso da aresta "e".
– $w(G)$ indica o peso total do grafo G. [8]

2.5 Redes sociais e comunidades

Uma rede social pode ser entendida como uma rede na qual o conjunto de objetos são as pessoas e as conexões entre elas são relações sociais. Comunidades, também chamadas de clusters ou agrupamentos, são grupos de vértices que tem grande probabilidade de compartilhar propriedades comuns ou tem papéis semelhantes no grafo. A formulação matemática para determinar comunidades em uma rede é denominada particionamento de grafos(graph clustering). Arestas que ligam vértices dentro de uma mesma comunidade são chamadas de arestas intracomunitárias e arestas que ligam vértices em diferentes comunidades são chamadas de arestas intercomunitárias. [5]

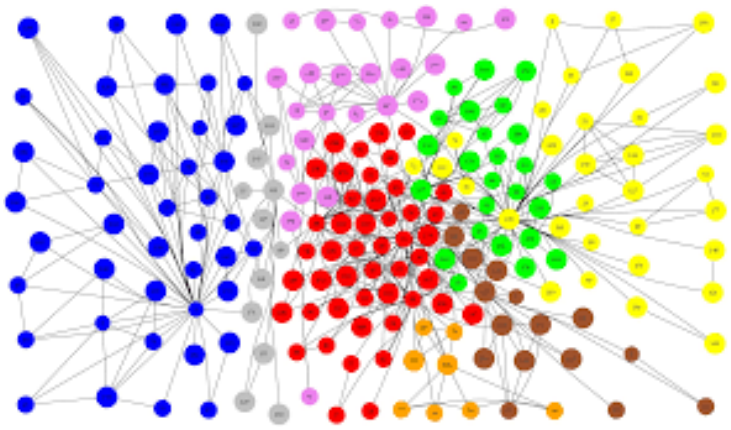


Figura 2.6: Exemplo de grafo com comunidades representadas por diferentes cores [1].

2.6 Métricas de avaliação de comunidades

2.6.1 Modularidade

O conceito de modularidade (Q) representa a qualidade de um particionamento de um grafo em comunidades. A modularidade é definida em (2):

$$Q = \sum_{c=1}^{n} [\frac{Lc}{m} - \gamma(\frac{Kc}{2m})^2] \qquad (2)$$

Onde a soma itera sobre todas as comunidades "c", "m" é o número de arestas, Lc é o número de conexões intracomunitárias para as comunidades "c", Kc é a soma dos graus dos vértices da comunidade e γ é o parâmetro de resolução.

O parâmetro de resolução define uma compensação arbitrária entre as arestas intracomunitárias e as arestas intercomunitárias. Padrões de agrupamento mais complexos podem ser descobertos analisando a mesma rede com vários valores de gama e, em seguida, combinando os resultados, no entantado o comum é usar simplesmente gamma = 1. De um modo geral valores de resolução maiores favorecem comunidades menores. [10]

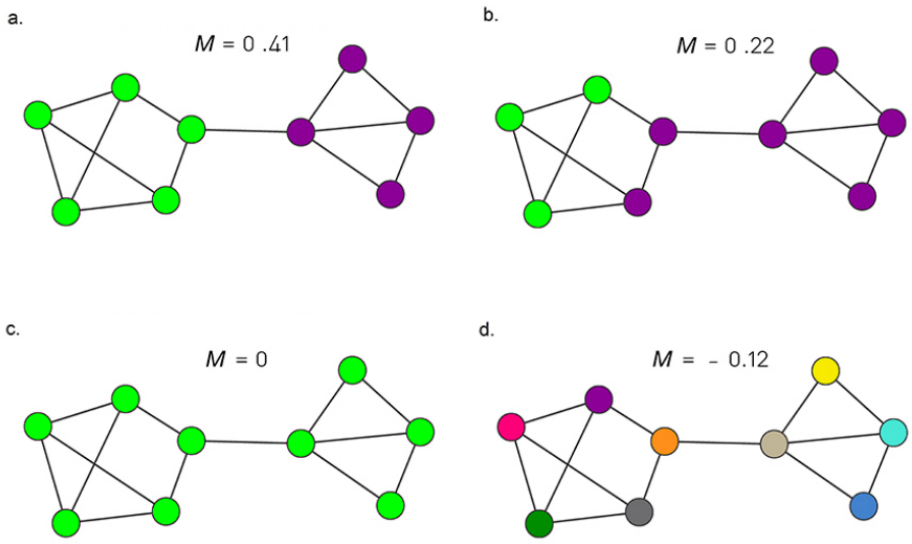


Figura 2.7: Exemplo de valores de modularidade M para diferentes resultados de comunidades achadas no mesmo grafo sendo cada comunidade representada por uma cor diferente [2].

2.6.2 Cobertura

A cobertura(coverage) de uma partição é a razão entre o número de arestas intracomunitárias e o número total de arestas no grafo. Quanto maior for a cobertura melhor será a qualidade do particionamento, considerando um mínimo de três comunidades. A cobertura é definida em (3):

$$Coverage = \frac{Ei}{Et} \qquad (3)$$

Onde Ei é o número de arestas intracomunitárias e Et o número total de arestas no grafo. [11]

2.6.3 Desempenho

O desempenho(performance) de uma partição é o número de arestas intracomunitárias mais as "não arestas"intercomunitárias dividido pelo número total de arestas potenciais. O desempenho de uma partição é definida em (4):

$$Desempenho = \frac{[(i,j) \in E, Ci = Cj] + [(i,j) \notin E, Ci \neq Cj]}{n(n-1)/2} \qquad (4)$$

Em outras palavras o desempenho é a soma entre pares de vértices ligado por uma aresta na mesma comunidade e pares de vértices que não estão ligados por uma aresta e não estão na mesma comunidade divididos por todas as possíveis arestas do grafo. [11]

2.7 Algoritmos de detecção de comunidades

2.7.1 Greedy Modularity Maximization

Criado por Clauset, Newman e Moore e por isso também chamado de CNM, o Greedy modularity maximization inicia com cada vértice tendo sua própria comunidade e é tentado unir pares de comunidades de forma que a modularidade seja a maior possível, dessa forma esse algoritmo tenta maximizar a modularidade das comunidades encontradas. O algoritmo cria um dendograma com o histórico de todas as junções feitas do início até haver apenas uma comunidade e assim é selecionado do dendograma a melhor divisão de comunidades com base no valor de modularidade. A figura 2.7 mostra comparações de valores de modularidade achadas no mesmo grafo com a variação das comunidades encontradas.

O algoritmo funciona da seguinte maneira:

Para um grafo com n vertices e m arestas:

1. É definido n comunidades singulares.
2. Enquanto não houver uma comunidade com todos os vertices:

   a. Para cada aresta e, se adicionar e irá conectar componentes desconectados, é calculado onde irá ocorrer o maior aumento da modularidade com a adição de e.

   b. Para o aumento de modularidade mais alto calculado, é feito a junção das duas comunidades.

3. Do dendrograma criado, é selecionado onde ocorreu a maior modularidade geral do grafo.

O algoritmo retorna o dendograma como uma lista de listas de vértices, cada lista de vértices representando uma comunidade. [10]

2.7.2 Girvan–Newman

O algoritmo Girvan–Newman detecta comunidades removendo progressivamente as arestas do grafo original. O algoritmo remove a aresta "mais valiosa", tradicionalmente a aresta com a maior centralidade de intermediação, em cada etapa. Um dendograma de resultados é criado conforme o grafo vai perdendo suas arestas. [9]

O algoritmo funciona da seguinte maneira:

1. A centralidade de intermediação de todas as arestas existentes no grafo é calculada.
2. Enquanto houver arestas no grafo:
    a. A aresta com maior centralidade é removida.
    b. A centralidade de todas as arestas afetadas pela remoção é recalculada.

2.7.3 Kernighan–Lin

Este algoritmo particiona um grafo em dois conjuntos de comunidades, trocando iterativamente pares de vértices para no fim achar onde foi obtido o maior valor de modularidade. [12]

O algoritmo funciona da seguinte maneira:

1. O grafo é dividido em dois grupos A e B de forma aleatória.
2. Até que todos os vértices tenham sido trocados pelo menos uma vez:
    a. É calculado a diferença de modularidade para todas as possíveis possibilidades de trocas de pares de vértices entre A e B.
    b. Pares de vértices que mais fazem a modularidade aumentar são trocados de comunidade.
3. É escolhido em que momento entre as trocas de vértices o grafo possuia a maior modularidade.

2.7.4 Fluid Communities

O algoritmo Fluid Communities é baseado na ideia simples de fluidos interagindo em um ambiente, expandindo e empurrando uns aos outros. Este processo é executado várias vezes até a convergência. Em todos os momentos, cada comunidade tem uma densidade total de 1, que é igualmente distribuída entre os vértices que contém. Sua inicialização é aleatória, então as comunidades encontradas podem variar um pouco em diferentes execuções. Um parâmetro "k" é preciso ser dado para o algoritmo definir o número de comunidades que irá retornar como resultado [3].

O algoritmo funciona da seguinte maneira:

1. Cada uma das "k" comunidades iniciais é inicializada em um vértice aleatório no grafo.

2. Até que uma iteração completa sobre todos os vértices é feita , de forma que nenhum vértice mude a comunidade a que pertence, o algoritmo:

   a. Itera sobre todos os vértices em uma ordem aleatória, atualizando a comunidade de cada vértice com base em sua própria comunidade e nas comunidades de seus vizinhos.

   b. Se um vértice muda de comunidade, as densidades de vértice das comunidades afetadas são ajustadas imediatamente.

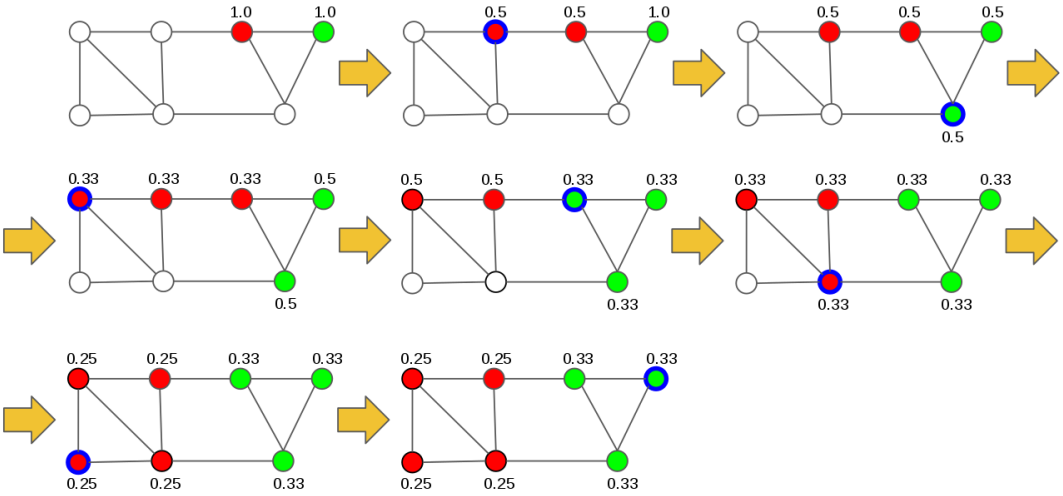


Figura 2.8: Etapas do algoritmo Fluid Communities usando $k = 2$ sendo a primeira comunidade representada pela cor vermelha e a segunda pela cor verde. Em cada etapa do algoritmo um dos vértices checa os valores de densidade dos vizinhos e decide em qual comunidade irá ficar [3].

2.7.5 Markov Cluster

O Markov Cluster é um algoritmo bastante utilizado para grafos fortemente conexos e que possuem pesos. Por meio de uma série de expansões e inflações na matriz de transição do grafo, a matriz vai aumentando os valores altos e diminuindo os menores cada vez mais para assim ser possível cortar as arestas que ficam com valores proximos de 0. É usado um parâmetro de expansão(normalmente 2) e um parâmetro de inflação.[4]

O algoritmo funciona da seguinte maneira com o parâmetro de expansão padrão ($expansão = 2$):

1. É Criado uma matriz adjancência a partir do grafo dado.
2. É feita uma matriz de transição a partir da matriz de adjacência.
3. Até que um estado estacionário seja alcançado (convergência):
    a. Expande a matriz fazendo uma multiplicação matricial dela por ela mesmo.
    b. É feita a inflação multiplicando cada termo da matriz por ele mesmo elevado ao número do parâmetro de inflação escolhido(normalmente 2).
    c. As arestas com valores próximos de 0 são cortadas.

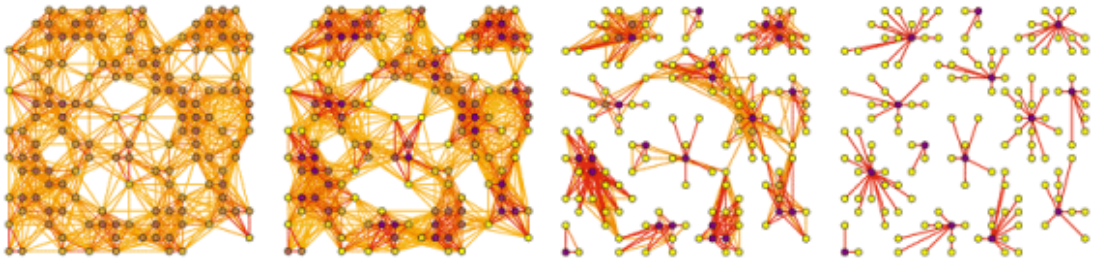


Figura 2.9: Algoritmo Markov Cluster cortando as arestas mais fracas até encontrar comunidades [4].

Capítulo 3

Trabalhos Relacionados

Nesta revisão da literatura, foram pesquisados artigos que descrevem o uso de algoritmos de detecção de comunidades e o processo de construção do grafo. Inicialmente, destaca-se o artigo que inspirou este trabalho, que foi a pesquisa de Pablo Barberá [13] onde foi extraídos os seguindos de usuários do Twitter para estimar sua posição política usando como base o posicionamento político de usuários famosos e a estimativa de ponto ideal bayesiana. Diferente do que foi feito no trabalho de Barberá [13], neste trabalho utilizam-se grafos para representar o relacionamento entre os senadores e detecção de comunidades para avaliar as semelhanças entre eles com base em seus seguindos.

Uma pesquisa bastante completa sobre detecção de comunidades em grafos foi feita por Santo Fortunato [11]. É feita uma exposição bastante aprofundada do tema, a partir da definição dos principais elementos do problema, até a apresentação da maioria dos métodos desenvolvidos, com especial enfoque nas técnicas projetadas por físicos e estatísticos, a partir da discussão de questões cruciais como o significado de clustering e como os métodos devem ser testados e comparados entre si. Esse trabalho também usa algoritmos de detecção de comunidades, mas com o objetivo de apenas explicar o funcionamento dos algoritmos e suas características.

No artigo feito por Leandro Farias [1], foram criados dois grafos, um "Grafo Questionário", onde foi criado com base em um formulário em que uma das perguntas tinha o objetivo de identificar as relações de amizade existentes entre os alunos, então através dessa pergunta foram criadas as arestas que representam relações de amizade entre alunos. O segundo grafo, que foi chamado de "Grafo Seção", foi criado a partir de dados da ficha de escolha de especialidade de engenharia preenchida pelos alunos no final do segundo ano do curso. Nessa ficha, cada aluno ordena as especialidades de engenharia de acordo com a sua preferência, considerando as 10 especialidades oferecidas no IME, as arestas são ligações entre alunos e especialidades, em que cada aluno está conectado a cada uma das três primeiras especialidades. O intuito dos grafos é fazer uma previsão de modo a com-

parar as comunidades obtidas para os grafos com as turmas reais formadas no início do terceiro ano letivo. O artigo não utiliza nenhuma métrica de avaliação das comunidades encontradas e apenas utiliza um algoritmo de detecção, o Greedy Modularity. O Greedy Modularity busca otimizar a modularidade do grafo, por isso teria sido útil a métrica de avaliação de modularidade nos dois grafos criados no citado trabalho.

O segundo artigo analisado foi o de Isabelle Alvesa [14]. O presente trabalho traz um estudo do problema de detecção de comunidades associadas às redes complexas e de forma a detectar as comunidades nessas redes também foi usado o algoritmo Greedy Modularity e também é utilizado um algoritmo de agrupamento (K-means) em cada um dos grafos criados, considerando o número de grupos igual ao número de comunidades. A comunidade definida e os agrupamentos construídos foram avaliados após o cálculo da modularidade e do coeficiente de silhueta. As diferenças entre este trabalho e o de Isabelle se da pelo fato de Isabelle ter utilizado um algoritmo de agrupamento depois de utilizado o Greedy Modularity, no entanto, não foram calculados os coeficientes de desempenho e cobertura.

O artigo de Diego Andrés de Barros Lima Barbosa [15] apresenta uma análise de uma rede de co-autoria construída a partir de dados reais da Plataforma Lattes do CNPq. A análise consiste em determinação dos vértices centrais, considerando as medidas de centralidade mais usadas. Um estudo sobre comunidades nesta rede também é apresentado utilizando os algoritmos de Girvan e Newman, e Greedy Modularity. O autor visa com os resultados obtidos utiliza-los para determinar os principais autores, grupos e, como consequência, os principais assuntos pesquisados no Brasil na área de grafos. Os vértices mais centrais dentro de uma mesma comunidade podem indicar características dos trabalhos dos pesquisadores. Assim como os vértices na fronteira das comunidades podem destacar interlocutores entre os grupos de pesquisa. Diego utiliza em seu trabalho os mesmos algoritmos utilizados neste (Girvan Newman e Greedy Modularity), no entanto, o foco dele é nas medidas de centralidade como a centralidade de intermediação e o grau de um vértice, por isso ele não utiliza nenhuma das três métricas de avaliação de comunidades aqui apresentadas.

O trabalho de Daiana Medeiros [16] tem por objetivo comparar o desempenho de dois algoritmos de agrupamento não-hierárquicos, quando aplicados a um conjunto de redes, frente a dois algoritmos de Detecção de Comunidades (Greedy Modularity e Walktrap). As comparações entre os resultados foram feitas a partir da aplicação do Índice de Silhueta. Esses algoritmos foram aplicados em 30 bases de dados artificiais, obtidas através do pacote "clustergeneration"da linguagem "R". O trabalho de Daiana Medeiros também compara diferentes algoritmos, mas com a diferença no uso do parâmetro de avaliação, pois ela usa o Índice de Silhueta e o trabalho aqui proposto utiliza modularidade, desempenho

e cobertura, e a comparação é feita com 5 algoritmos diferentes.

Também foram revisados artigos para serem usados como base teórica para entendimento de todos os algoritmos utilizados neste trabalho: Greedy Modularity [10], Kernighan-Lin [12], Fluid Communities [3], Girvan-Newman [9], Markov Cluster [4], modularidade [10], desempenho e cobertura [4].

Capítulo 4

Desenvolvimento

4.1 Criação dos grafos de senadores e seguindos

Para a criação dos grafos, inicialmente, foram extraídos os usuários dos seguindos de todas as contas do Twitter dos senadores. Para isso, foi necessário usar uma API de extração de dados do Twitter que usa o framework Selenium para automatizar tarefas no navegador.

Na construção dos grafos foi utilizada a biblioteca igraph [1]. Com os usuários dos seguindos de cada senador, foi construído um primeiro grafo com os senadores como vértices e como arestas os seguindos em comum entre dois senadores. A cada aresta é associado um peso, que corresponde ao número de seguindos em comum. Depois, foi construído um segundo grafo com os seguindos como vértices e as arestas os senadores em comum entre dois seguindos. O peso das arestas é o número de senadores em comum que os seguiam. Cada vértice recebeu o nome do usuário como atributo para identificação.

Para uma avaliação ideal das comunidades dos senadores, seria feita uma análise de cada um dos senadores individualmente em uma determinada comunidade, mas para não fugir do escopo do trabalho e para simplificar a análise, os partidos foram usados como atributo de cada vértice de senador para posteriormente avaliar as comunidades usando os partidos desses senadores.

Nos dois grafos construídos, foram removidas as arestas que possuíam peso menor que a média dos pesos de todas as arestas para eliminar arestas pouco significativas. Isso foi feito para aumentar a esparsidade do grafo e remover os pesos das arestas (construir grafos sem pesos nas arestas). Essas alterações possibilitaram a ampliação do número de algoritmos de detecção a serem utilizados para assim ser possível aplicar algoritmos de detecção de comunidades que não usam peso nas arestas e aplicar os grafos com peso nos algoritmos que necessitam de peso nas arestas.

[1] https://igraph.org/

Com isso, foram construídos quatro grafos, sendo dois com os senadores totalizando 80 vértices e dois com os seguindos dos senadores totalizando 1020 vértices. Sendo os dois grafos valorados para serem utilizados em algoritmos que precisem do peso das arestas. Algumas vezes um algoritmo pode não achar nenhuma comunidade para dado grafo, por isso foi importante ter criado tanto um grafo de senadores como um grafo de seguindos desses senadores para que nenhum algoritmo fique de fora das análises.

Foi desenvolvido também um código que plota com a biblioteca Matplotlib os vértices do grafo para assim identificar de forma visual as comunidades criadas pelos algoritmos. Para cada comunidade é designada uma cor única gerada aleatoriamente.

Outra biblioteca utilizada neste trabalho é a biblioteca de construção de grafos NetworkX[2]. Ela possui três importantes algoritmos na avaliação de comunidades (modularidade, cobertura e desempenho), além de possuir algoritmos de detecção de comunidades. Em decorrência disso, após a criação dos grafos, foi feita uma migração do formato de grafo do Igraph para o formato do NetworkX.

4.2 Aplicação dos algoritmos

Primeiramente, foi testado o algoritmo de detecção Greedy Modularity Maximization nos dois grafos valorados usando como parâmetro o peso e todos os valores de "resolução"que conseguiram achar comunidades no grafo(valores de 1 até 20). Para cada resultado de comunidade achado para um dado parâmetro de resolução foi usado as três métricas de avaliação(Modularidade, Desempenho, e Cobertura), então para isso foi feito um código que automaticamente aplicou todos os parâmetros e métricas de avaliação e logo depois anotou os resultados. O algoritmo Greedy Modularity possui a vantagem de ter como objetivo principal tentar otimizar os parâmetros de modularidade, parâmetro mais popular na avaliação de comunidades.

O segundo algoritmo de detecção de comunidades foi o Kernighan-Lin que também visa otimizar a modularidade, mas de uma forma diferente do Greedy Modularity. É esperado que o algoritmo consiga polarizar politicamente os senadores, já que o algoritmo retorna obrigatoriamente duas comunidades. Foram usados como parâmetro os dois grafos valorados e anotados os resultados da utilização das três métricas de avaliação.

O terceiro algoritmo foi o Girvan-Newman, que diferente dos anteriores, não se preocupa com a modularidade e sim com a centralidade de intermediação das arestas. O algoritmo elimina todas as arestas uma por uma começando da aresta com maior centralidade até a menor, assim foi calculada as métricas de avaliação para cada etapa da remoção das arestas para ser possível identificar em que momento da remoção das arestas

[2]https://networkx.org/

houve melhores resultados. Foi usado como parâmetro o grafo dos senadores sem peso nas arestas e foram anotados todos os resultados das métricas de avaliação a cada corte. (As arestas do grafo foram cortadas de três em três).

O quarto algoritmo foi o Fluid Communities, a vantagem desse algoritmo é a possibilidade de poder escolher o número exato de comunidades "k" a serem encontradas no grafo. O Fluid Communities não usa modularidade nem centralidade para construir as comunidades, em resumo, o algoritmo tenta fazer as "k" comunidades puxarem para si os vértices mais próximos e com mais conexões. Foi usado como parâmetro o grafo dos seguindos sem peso nas arestas e "k" variando de 3 a 19 para assim avaliar com as métricas em qual número de comunidades houve os melhores resultados.

O quinto e último algoritmo escolhido foi o Markov Cluster, que é bastante utilizado em dados de bioinformática [17]. O Markov usa a matriz de transição com as operações de inflação e expansão para cortar o máximo de arestas possíveis e achar as comunidades. Foi usado como parâmetro o grafo dos senadores com peso nas arestas e os valores de inflação que acharam comunidades(4 até 25) e anotado os resultados das três métricas utilizadas para cada valor de inflação.

Foi feito uma plotagem dos grafos de senadores sendo cada vértice colorido de uma cor que representa uma comunidade. Não foi feito plotagem para o grafo de seguindos pois, o número muito grande de arestas atrapalha a visualização do grafo. Para cada senador foi vinculado seu partido e feito um algoritmo que calcula quantos senadores de um determinado partido a comunidade possui, com isso foi possível criar uma tabela que mostra o número de partidos para um dado resultado de comunidades.

Para analisar a mudança do desempenho dos algoritmos com os diferentes parâmetros que os algoritmos possibilitam escolher, foi feito um gráfico que mostra a variação dos resultados das métricas para cada parâmetro utilizado na entrada dos algoritmos.

4.3 Comparação com grafos gerados aleatoriamente

Com o intuito de fazer uma comparação entre os algoritmos em diferentes situações de grafos, eles foram aplicados em grafos com diferentes características. Foi criado 4 grafos gerados aleatoriamente usando a função "gaussian random partition graph"da biblioteca do NetworkX. A função pede como parâmetro o número de vértices, a probabilidade de gerar arestas intracomunitárias e a probabilidade para gerar arestas intercomunitárias.

Os quatro grafos foram criados com 100 vértices e foram divididos em 4 dificuldades diferentes: "fácil", "médio", "difícil"e "muito difícil", sendo o grafo mais fácil, o grafo com mais arestas intracomunitárias e menos arestas intercomunitárias e o mais difícil o oposto.

As probabilidades de geração de arestas usadas para cada um dos 4 grafos são exibidas na tabela abaixo:

	Probabilidade de arestas intracomunitárias	Probabilidade de arestas intercomunitárias
Grafo fácil	0.5	0.05
Grafo médio	0.25	0.1
Grafo difícil	0.1	0.2
Grafo muito difícil	0.05	0.3

Tabela 4.1: Tabela com a probabilidade usada para geração de arestas intracomunitárias e intercomunitária para cada um dos 4 grafos.

Assim como foi feito com o grafo de senadores, foram anotados todos os resultados das métricas variando os parâmetros do algoritmo, mas os gráficos gerados foram feitos usando a dificuldade do grafo como eixo X. Assim, foi possível avaliar o quanto os resultados dos algoritmos variam quando utilizados nas 4 diferentes dificuldades de grafos. O melhor resultado de cada algoritmo foi anotado em uma tabela para cada um dos quatro grafos.

Capítulo 5

Resultados

5.1 Grafos aleatórios de diferentes dificuldades

Além da aplicação dos algoritmos nos grafos de senadores, é importante analisar a consistência dos resultados desses mesmos algoritmos quando aplicados em diferentes tipos de grafos para evitar tirar conclusões precipitadas sobre os algoritmos quando avaliamos apenas considerando os grafos criados a partir dos senadores.

Foi feito um gráfico para cada métrica que mostra a variação do resultado do algoritmo para os 4 diferentes níveis de dificuldade. Também foi criada uma tabela de comparação para cada um dos 4 grafos gerados que mostram o melhor resultado que os algoritmos conseguiram atingir nas 3 métricas de avaliação. Com exceção do Kernighan-Lin que é feito para achar apenas 2 comunidades, foi considerado na comparação apenas resultados que acharam um mínimo de 3 comunidades. Para o Markov Cluster, foi usado apenas o parâmetro padrão de $inflação = 2$ pois, foi o único parâmetro que conseguiu achar pelo menos 3 comunidades nos grafos.

5.1.1 Análise dos gráficos dos resultados

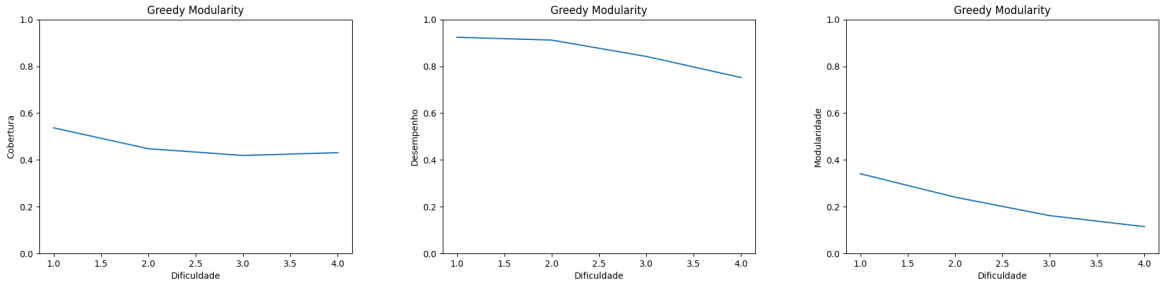


Figura 5.1: Gráficos da variação das 3 métricas de avaliação do Greedy Modularity para cada nível de dificuldade.

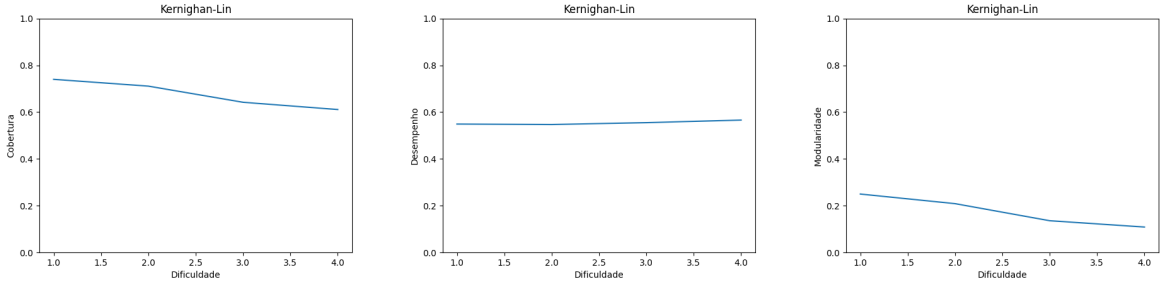


Figura 5.2: Gráficos da variação das 3 métricas de avaliação do Kernighan-Lin para cada nível de dificuldade.

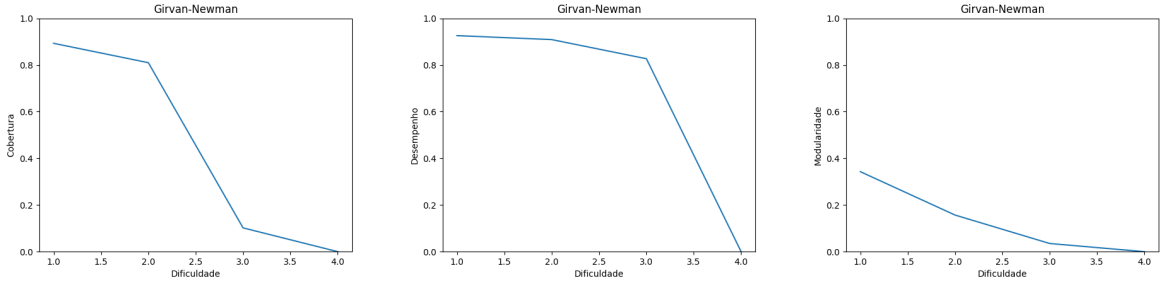


Figura 5.3: Gráficos da variação das 3 métricas de avaliação do Girvan-Newman para cada nível de dificuldade.

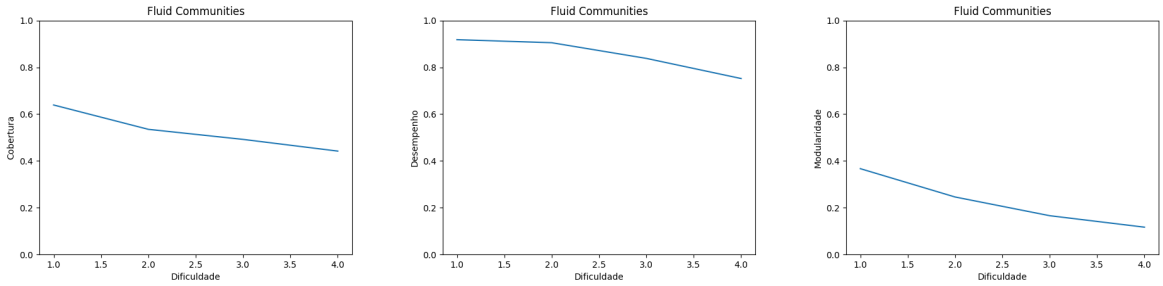


Figura 5.4: Gráficos da variação das 3 métricas de avaliação do Fluid Communities para cada nível de dificuldade.

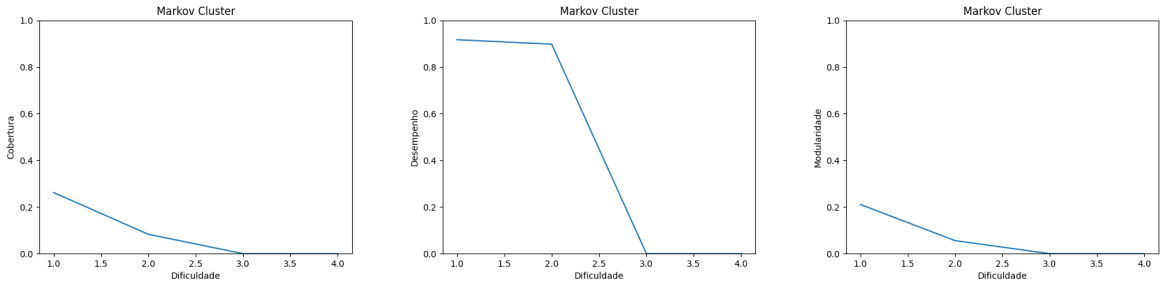


Figura 5.5: Gráficos da variação das 3 métricas de avaliação do Markov Cluster para cada nível de dificuldade.

Analisando o gráfico é possível concluir que o aumento da dificuldade do grafo diminui todos os resultados das métricas das comunidades. Os dois algoritmos que mais sofreram alterações nos resultados foram o Girvan-Newman e o Markov Cluster. O Girvan-Newman conseguiu ótimos resultados no grafo fácil, mas teve uma queda brusca com o aumento da dificuldade. Os algoritmos Kernighan-Lin, Greedy Modularity e Fluid Communities obtiveram quedas mais leves nos resultados. Por forçar sempre o retorno de exatamente duas comunidades como resultado, o Kernighan-Lin apesar de possuir valores de métricas menores que os outros algoritmos, foi o que apresentou maior estabilidade, tendo o desempenho praticamente estável para todos os 4 diferentes grafos.

5.1.2 Comparação dos resultados

Grafo Fácil

Com uma boa diferença dos outros algoritmos, o Girvan-Newman após 4 cortes obteve o melhor valor de cobertura para o grafo fácil ($Cobertura = 0.893$), e também obteve o melhor valor de desempenho após 38 cortes ($Desempenho = 0.926$), perdendo apenas para o Fluid Communities em modularidade que obteve o maior valor com $modularidade = 0.367$. Apesar de um bom valor de desempenho, o Markov Cluster obteve os piores valores de cobertura e modularidade.

Grafo Fácil			
	Cobertura	**Desempenho**	**Modularidade**
Girvan-Newman	0.893 (Cortes = 4)	0.926 (Cortes = 38)	0.343 (Cortes = 15)
Markov Cluster	0.262	0.917	0.211
Kernighan-Lin	0.740	0.549	0.250
Greedy Modularity	0.537 (Resolution = 1)	0.924 (Resolution = 5)	0.341 (Resolution = 1)
Fluid Communities	0.639 (k = 3)	0.918 (k = 16)	0.367 (k = 11)

Tabela 5.1: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo fácil.

Grafo Médio

Assim como no grafo fácil, o Girvan-Newman continuou obtendo o maior valor de cobertura ($cobertura = 0.81$) e precisou de mais cortes de arestas para conseguir bons valores de desempenho e modularidade. O Greedy Modularity obteve o melhor valor de desempenho ($desempenho = 0.912$) e o segundo melhor valor de modularidade ($modularidade = 0.241$)

perdendo apenas para o Fluid Communities que obteve uma modularidade um pouco maior ($modularidade = 0.246$). Assim como no grafo fácil, o Markov Cluster obteve um bom valor de desempenho, mas também teve os piores valores de cobertura e modularidade.

Grafo Médio			
	Cobertura	**Desempenho**	**Modularidade**
Girvan-Newman	0.810 (Cortes = 8)	0.909 (Cortes = 57)	0.157 (Cortes = 35)
Markov Cluster	0.083	0.898	0.056
Kernighan-Lin	0.711	0.547	0.209
Greedy Modularity	0.448 (Resolution = 1)	0.912 (Resolution = 4)	0.241 (Resolution = 1)
Fluid Communities	0.535 (k = 3)	0.905 (k = 19)	0.246 (k = 7)

Tabela 5.2: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo médio.

Grafo Difícil

Para o grafo difícil o Girvan-Newman precisou de muitos cortes para achar comunidades e não conseguiu manter um bom valor de cobertura. Assim como no grafo médio, o Greedy Modularity obteve o melhor valor de desempenho ($desempenho = 0.827$) e na modularidade obteve um valor um pouco menor que o Fluid Communities que obteve o melhor valor de modularidade ($modularidade = 0.166$). O Kernighan-Lin conseguiu o melhor valor de cobertura ($cobertura = 0.642$) e o Markov Cluster não conseguiu achar comunidades nessa dificuldade de grafo.

Grafo Difícil			
	Cobertura	**Desempenho**	**Modularidade**
Girvan-Newman	0.102 (Cortes = 73)	0.827 (Cortes = 79)	0.035 (Cortes = 73)
Markov Cluster	0	0	0
Kernighan-Lin	0.642	0.555	0.136
Greedy Modularity	0.419 (Resolution = 1)	0.842 (Resolution = 3)	0.162 (Resolution = 1)
Fluid Communities	0.492 (k = 3)	0.838 (k = 19)	0.166 (k = 6)

Tabela 5.3: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo difícil.

Grafo Muito Difícil

Para o grafo mais difícil, tanto o Girvan-Newman como o Markov Cluster não conseguiram achar comunidades. O Fluid Communities e o Greedy Modularity obtiveram resultados extremamente próximos nas três métricas, os dois algoritmos empataram e obtiveram o melhor desempenho ($desempenho = 0.752$), mas o Fluid Communities conseguiu a melhor modularidade ($modularidade = 0.117$) e o Kernighan-Lin obteve o melhor valor de cobertura ($cobertura = 0.611$).

Grafo Muito Difícil			
	Cobertura	**Desempenho**	**Modularidade**
Girvan-Newman	0	0	0
Markov Cluster	0	0	0
Kernighan-Lin	0.611	0.566	0.109
Greedy Modularity	0.431 (Resolution = 1)	0.752 (Resolution = 2)	0.115 (Resolution = 1)
Fluid Communities	0.442 (k = 3)	0.752 (k = 15)	0.117 (k = 3)

Tabela 5.4: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo muito difícil.

5.2 Grafos de senadores e seguindos

Os resultados adquiridos usando os cinco algoritmos foram organizados em uma tabela, cada coluna da tabela mostrando os resultados de cada uma das três métricas utilizadas para um dado parâmetro do algoritmo e para cada métrica foi feito um gráfico. Os gráficos para cada grafo(seguindos e senadores) foram colocados a baixo lado a lado para cada uma das três métricas para facilitar a comparação. Para o grafo dos senadores foi feita também uma tabela que mostra o número de senadores de um partido na comunidade encontrada. Por fim foi feita uma tabela de comparação para cada grafo com o melhor resultado de cada algoritmo considerando apenas os resultados que conseguiram achar pelos menos duas comunidades.

5.2.1 Análise dos gráficos dos resultados

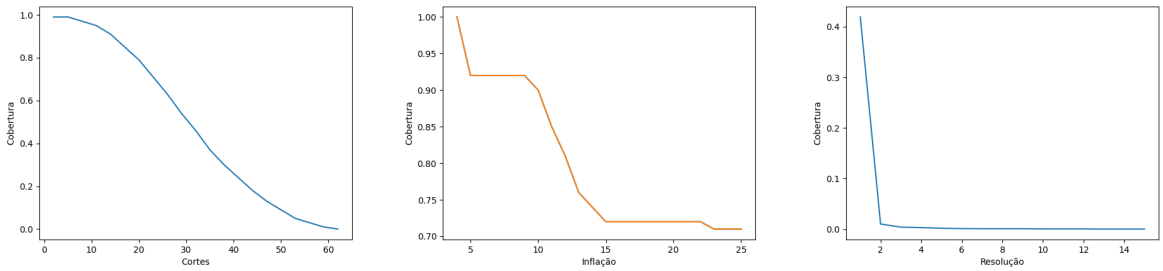


Figura 5.6: Gráfico da cobertura nos grafos de senadores para cada valor dos parâmetros utilizados nos algoritmos(Girvan-Newman, Markov Cluster e Greedy Modularity).

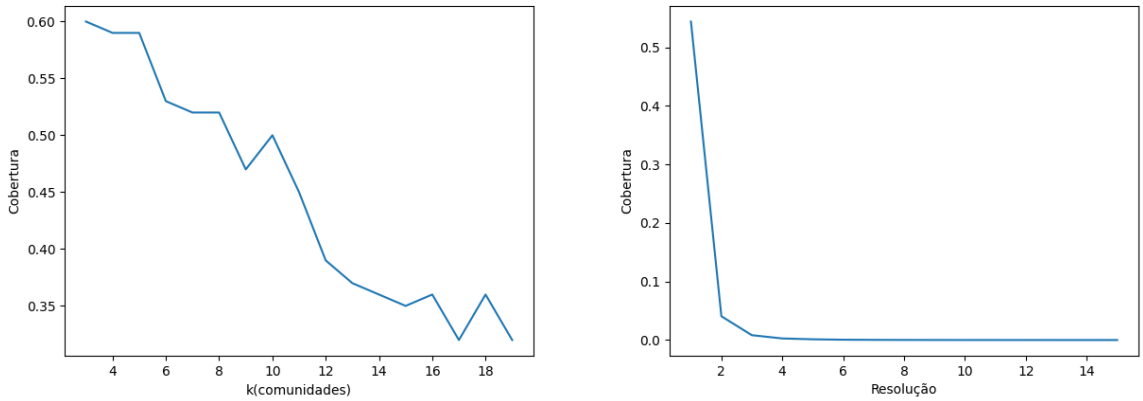


Figura 5.7: Gráfico da cobertura nos grafos de seguindos para cada valor dos parâmetros utilizados nos algoritmos(Fluid Communities e Greedy Modularity).

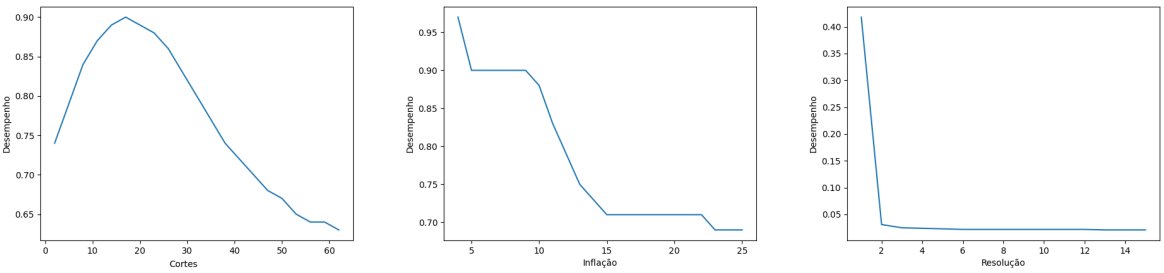


Figura 5.8: Gráfico do desempenho nos grafos de senadores para cada valor dos parâmetros utilizados nos algoritmos(Girvan-Newman, Markov Cluster e Greedy Modularity).

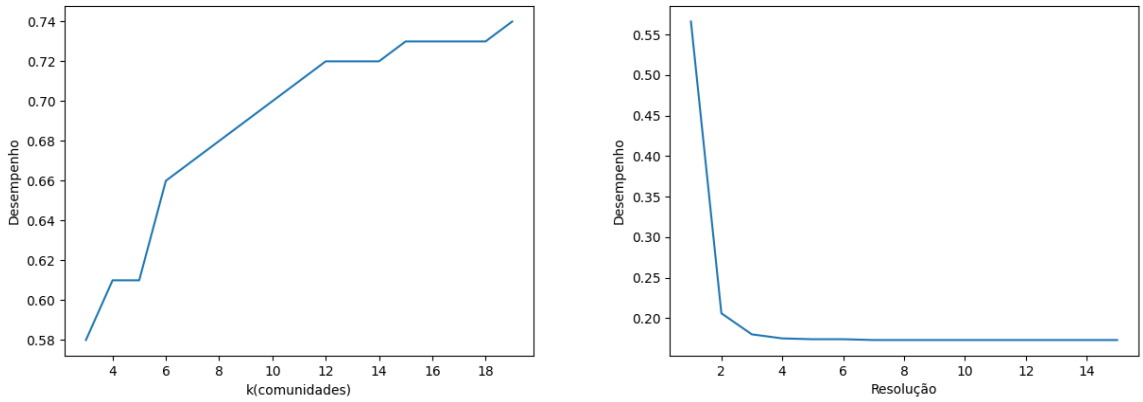


Figura 5.9: Gráfico do desempenho nos grafos de seguindos para cada valor dos parâmetros utilizados nos algoritmos(Fluid Communities e Greedy Modularity).

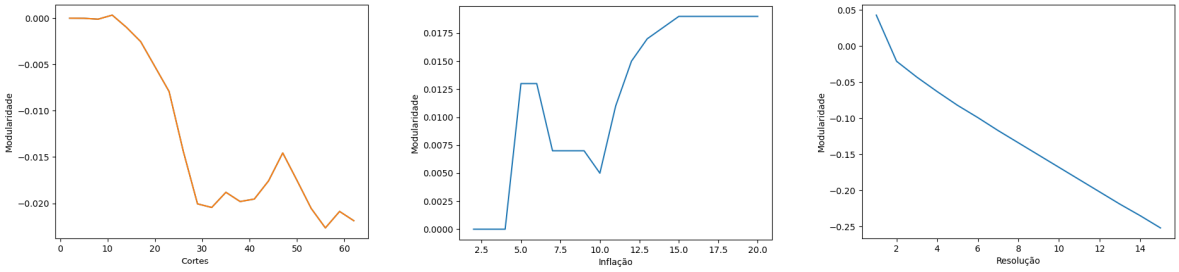


Figura 5.10: Gráfico da modularidade nos grafos de senadores para cada valor dos parâmetros utilizados nos algoritmos(Girvan-Newman, Markov Cluster e Greedy Modularity).

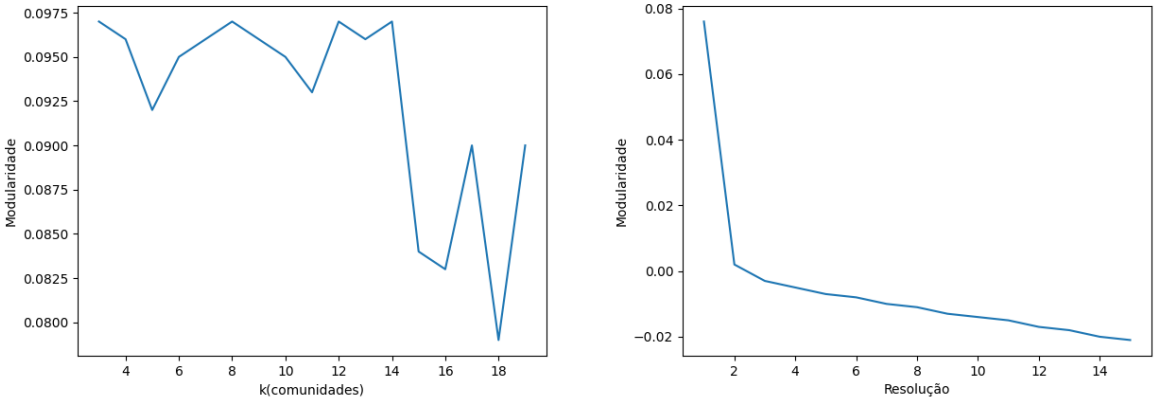


Figura 5.11: Gráfico da modularidade nos grafos de seguindos para cada valor dos parâmetros utilizados nos algoritmos(Fluid Communities e Greedy Modularity).

Para o Fluid Communities foi usado o grafo de seguindos sem peso. Não foi possível utilizar o grafo de senadores sem peso, pois, o grafo possui dois componentes que não estão ligados, por isso não se trata de um grafo conexo e o algoritmo exige um para funcionar. Tanto a modularidade como a cobertura diminuíram com o número de comunidades achadas no grafo, isso se da pelo fato de arestas intercomunitárias irem aumentando com o aumento de comunidades no grafo. O desempenho, no entanto, obteve um crescimento contínuo com o aumento do número de comunidades.

Para o Girvan-Newman foi usado o grafo de senadores sem peso. O algoritmo não foi capaz de achar comunidades com o grafo de seguindos. Tanto a modularidade como a cobertura diminuíram com o número de cortes no grafo. Como o algoritmo corta arestas aos poucos para achar comunidades, é esperado que a cada corte e consequentemente aumento de comunidades, o número de arestas intercomunitárias aumente. O desempenho apresentou um aumento até chegar no ápice em 17 cortes, e após 17 cortes foi diminuindo, o que significa que após esse número de cortes o algoritmo começou a cortar arestas intracomunitárias.

Para o Markov Cluster foi usado o grafo de senadores com peso nas arestas. Assim como o algoritmo de Girvan-Newman, o Markov Cluster corta arestas aos poucos para achar comunidades. O algoritmo não conseguiu achar comunidades com o grafo de seguindos. Tanto o desempenho como a cobertura diminuíram com o aumento da inflação e apresentaram uma diminuição síncrona. Por outro lado, a modularidade aumentou e estabilizou a partir de $inflação = 15$ o que mostra que um número alto de inflação pode diminuir a cobertura e o desempenho, mas aumentar a modularidade.

Para o Greedy Modularity foi possível achar comunidades tanto no grafo de senadores com peso nas arestas como no grafo de seguindos com peso nas arestas. O aumento do parâmetro de resolução não apresentou nenhuma melhora nas métricas, pois, as três métricas apresentaram uma diminuição muito grande com o aumento da resolução nos dois grafos. O melhor valor de resolução acabou sendo o valor 1 que é o padrão de resolução do algoritmo.

O algoritmo de Kernighan-Lin não possui parâmetro, mas também foi possível aplicar ele tanto no grafo de senadores com peso nas arestas como no grafo de seguindos com peso nas arestas. Os valores de cobertura e desempenho foram bastante parecidos nos dois grafos, e foram bem próximos de 0.5 o que mostra que o algoritmo equilibrou o número de arestas intercomunitárias e intracomunitárias das duas comunidades nos dois grafos.

Senadores Kernighan-Lin		
Cobertura	Desempenho	Modularidade
0.49	0.48	0.0048
Seguindos Kernighan-Lin		
Cobertura	Desempenho	Modularidade
0.47	0.46	0.0193

Tabela 5.5: Tabela dos resultados das métricas para o grafo de seguindo e de senadores para o algoritmo de Kernighan-Lin.

5.2.2 Análise das comunidades de senadores

Girvan-Newman

O algoritmo não apresentou bons resultados para o dado grafo porque conseguiu achar apenas uma comunidade para todos os cortes, isso significa que para cada corte da aresta com maior centralidade, o algoritmo foi isolando senadores da comunidade inicial fazendo com que haja apenas uma comunidade grande e vários senadores isolados.

Comunidade 0
CIDADANIA: 1
DEM: 2
MDB: 2
PATRIOTA: 1
PL: 1
PODEMOS: 3
PP: 3
PROS: 1
PSC: 1
PSDB: 3
PSD: 2
PSL: 1
PT: 1
REDE: 1

Tabela 5.6: Tabela do número de senadores de um partido na única comunidade achada pelo Girvan-Newman após 35 cortes.

Markov Cluster

Usando 20 como inflação, é possível notar nas 3 comunidades encontradas que 5/6 dos senadores do PT se encontram na comunidade 0 enquanto os senadores mais voltados para a direita estão na comunidade 1. Fazendo uma análise mais específica dos dois senadores isolados do MDB na comunidade 2 e checando os seus nomes, foi visto que os

dois se tratam da senadora Nilda e o senador Veneziano, senadores que são mãe e filho respectivamente o que explica a grande similaridade a ponto de o algoritmo botar os dois na mesma comunidade e separados de todos os outros senadores.

Comunidade 0	Comunidade 1	Comunidade 2
MDB: 2	CIDADANIA: 3	MDB: 2
PODEMOS: 1	DEM: 4	
PSD: 2	MDB: 12	
PT: 5	PATRIOTA: 1	
	PDT: 3	
	PL: 4	
	PODEMOS: 8	
	PP: 6	
	PROS: 3	
	PSC: 1	
	PSDB: 7	
	PSD: 9	
	PSL: 1	
	PT: 1	
	REDE: 2	
	REPUBLICANOS: 1	

Tabela 5.7: Tabela do número de senadores de um partido nas 3 comunidades encontradas.

Kernighan-Lin

A divisão de partidos foi bem igualitária com exceção do partido PODEMOS E PP que tiveram grande parte de seus senadores na comunidade 0 e do PT que todos os senadores ficaram na comunidade 1.

Comunidade 0	Comunidade 1
CIDADANIA: 1	CIDADANIA: 2
DEM: 2	DEM: 2
MDB: 9	MDB: 7
PDT: 2	PATRIOTA: 1
PL: 2	PDT: 1
PODEMOS: 7	PL: 2
PP: 5	PODEMOS: 2
PROS: 1	PP: 2
PSDB: 3	PROS: 2
PSD: 5	PSC: 1
REDE: 1	PSDB: 4
REPUBLICANOS: 1	PSD: 6
	PSL: 1
	PT: 6
	REDE: 1

Tabela 5.8: Tabela do número de senadores de um partido nas 2 comunidades encontradas.

Greedy Modularity

Usando 1 como resolução, foram achadas 4 comunidades de senadores, da para se observar uma concentração de senadores de partidos de direita na comunidade 0 enquanto a comunidade 1 apresentou maior concentração de senadores de partidos de esquerda como todos os 6 do PT. A comunidade 2 apresentou uma mistura e a comunidade 3 isolou os mesmos 2 senadores do MDB(mãe e filho) assim como o Markov Cluster, mas com a adição de um senador do PP.

Comunidade 0	Comunidade 1	Comunidade 2	Comunidade 3
CIDADANIA: 2	DEM: 1	CIDADANIA: 1	MDB: 2
DEM: 3	MDB: 3	MDB: 2	PP: 1
MDB: 9	PODEMOS: 1	PDT: 2	
PATRIOTA: 1	PROS: 2	PL: 2	
PDT: 1	PSD: 3	PP: 1	
PL: 2	PT: 6	PSDB: 1	
PODEMOS: 8	REDE: 2	PSD: 2	
PP: 5			
PROS: 1			
PSC: 1			
PSDB: 6			
PSD: 6			
PSL: 1			
REPUBLICANOS: 1			

Tabela 5.9: Tabela do número de senadores de um partido nas 4 comunidades encontradas.

5.2.3 Comparação dos resultados

Grafo de seguindos

No grafo de seguindos o algoritmo que obteve melhores resultados em todas as métricas de avaliação foi o Fluid Communities aplicado ao grafo com as arestas cortadas usando $k = 3$ comunidades, o algoritmo obteve *cobertura* $= 0.6$ e *modularidade* $= 0.097$, e o melhor valor de desempenho foi 0.74 para $k = 19$. Em segundo lugar fica o Greedy Modularity com *resolução* $= 1$ que apresentou melhores resultados que o Kernighan-Lin em todas as métricas.

Grafo de seguindos			
	Cobertura	**Desempenho**	**Modularidade**
Kernighan-Lin	0.47	0.46	0.019
Greedy Modularity	0.544	0.566	0.076
Fluid Communities	0.6 (k = 3)	0.74 (k = 19)	0.097 (k = 3)

Tabela 5.10: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo de seguindos.

Grafo de senadores

Nos grafos de senadores o Girvan-Newman como mencionado anteriormente conseguiu achar apenas uma comunidade então podemos desconsiderar ele da comparação, pois, é necessário que o algoritmo ache um mínimo de 2 comunidades para ser possível comparar.

O Markov Cluster com $inflação = 7$ obteve com uma grande diferença os melhores resultados para cobertura e desempenho, 0.92 e 0.9 respectivamente, e para $inflação = 20$ obteve-se 0.019 de modularidade, ficando atrás apenas do Greedy Modularity que obteve 0.043 com $resolução = 1$ que foi o melhor parâmetro para todas as métricas deste algoritmo. O Kernighan-Lin teve um resultado melhor que o Greedy Modularity em desempenho e cobertura, e obteve o pior resultado em modularidade.

Grafo de senadores			
	Cobertura	**Desempenho**	**Modularidade**
Kernighan-Lin	0.49	0.48	0.005
Greedy Modularity	0.419	0.418	0.043
Markov Cluster	0.90 (inflação = 2)	0.92 (inflação = 2)	0.019 (inflação = 20)

Tabela 5.11: Tabela com o melhor resultado de cada algoritmo para cada uma das três métricas para o grafo de senadores.

Capítulo 6

Conclusão

Com esse trabalho foi possível testar e entender o funcionamento dos principais algoritmos de detecção de comunidades. Analisando todos os resultados envolvendo os senadores, se conclui que o algoritmo Fluid Communities obteve o melhor resultado nas três métricas de avaliação para os seguindos e o Markov Cluster foi o melhor para os senadores considerando que das três métricas ele obteve os melhores valores em desempenho e cobertura. Esses resulados são interessantes de se notar dado que a maioria dos artigos que usam algum algoritmo de detecção de comunidades costumam usar o Greedy Modularity.

Como comentado anteriormente, o que impediu os resultados das comunidades de senadores serem mais consistentes é a possibilidade dos senadores seguirem políticos com visões políticas muito diferentes. A criação de grafos usando seguidores ao invés de seguindos poderia minimizar esse erro, mas mesmo assim é possível observar uma polarização política nos resultados principalmente em relação aos senadores do partido PT e PODEMOS que quase não tem seus senadores divididos e estão sempre em comunidades opostas. Apesar de fugir do escopo deste trabalho, o ideal seria analisar cada senador um por um para observar melhor os resultados das comunidades ao invés de dividir apenas por partidos, pois, muitos senadores de um mesmo partido podem possuir opiniões políticas extremamente diferentes principalmente dentro dos partidos considerados de "centro".

Avaliando todos os grafos incluindo os aleatórios, o Fluid Communities foi o melhor algoritmo na métrica de modularidade e conseguiu bons resultados de cobertura e desempenho nos grafos mais difíceis, mas não conseguiu resultados tão bons de cobertura e desempenho nos grafos mais fáceis. O Greedy Modularity conseguiu bons resultados de modularidade para todos os grafos em que foi utilizado, mesmo ficando um pouco atrás do Fluid Communities, mas ainda assim conseguiu obter um valor de desempenho maior que o Fluid Communities na maioria dos casos, em contrapartida, o valor de cobertura costuma ser baixo principalmente nos grafos mais fáceis.

O Kernighan-Lin não obteve resultados altos nas métricas de avaliação, mas foi o

algoritmo que se mostrou mais estável com o aumento da dificuldade do grafo, conseguindo um bom valor de cobertura até para os grafos difíceis. O Girvan-Newman conseguiu excelentes resultados nos grafos mais fáceis conseguindo o melhor valor de desempenho e cobertura no grafo fácil, mas os resultados decaíram bastante com a dificuldade do grafo. Assim como o Girvan-Newman o Markov Cluster também consegue obter bons valores em grafos fáceis, principalmente em desempenho, mas os resultados decaem drasticamente em grafos mais difíceis.

Referências

[1] Oliveira, Leandro Farias Maia; Heytor Bruno Nobre; Claudia M. Justel; Camila C.F.: *Community detection problem for an academic network.* Revista Eletrônica Sistemas e Gestão, 9(4), 2014. ix, 8, 14

[2] *https://www.cos.ufrj.br/ daniel/rc-2018/slides/aula*$_1$*6.pdf.ix*, 9

[3] *Parés f. et al. (2018) fluid communities: A competitive, scalable and diverse community detection algorithm. in: Cherifi c., cherifi h., karsai m., musolesi m. (eds) complex networks their applications vi. complex networks 2017. studies in computational intelligence, vol 689. springer, cham. https://doi.org/10.1007/978-3-319-72150-7*$_1$*9.ix*, 11, 12, 16

[4] Paulo Faleiros, Thiago de: *Algoritmos para o problema de particionamento.* 2010. ix, 12, 13, 16

[5] Oliveira, Leandro Farias Maia ; Heytor Bruno Nobre ; Claudia M. Justel ; Camila C.F.: *Detecção de comunidades numa rede de relacionamento de alunos.* System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES, 679(05), 2005. 1, 8

[6] *Girvan, m. newman, m. e. j. (2002). community structure in social and biological networks. proceedings of the national academy of sciences, 99(12):7821–7826. issn 0027-8424.* 1

[7] Machado, Felipe Menezes: *Detecção de comunidades em grafos multicamada muito grandes.* 2016. 1

[8] *https://homepages.dcc.ufmg.br/ loureiro/md/md_9grafos.pdf.* 3, 4, 5, 6, 7, 8

[9] Despalatović, Ljiljana, Tanja Vojković e Damir Vukicević: *Community structure in networks: Girvan-newman algorithm improvement.* Em *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, páginas 997–1002, 2014. 7, 11, 16

[10] Clauset, A., Newman M. E. Moore C.: *Finding community structure in very large networks.* Physical Review E, 70(4), 2004. 9, 10, 16

[11] Fortunato, Santo: *Community detection in graphs.* Physics Reports, 486(3-5):75–174, Feb 2010, ISSN 0370-1573. `http://dx.doi.org/10.1016/j.physrep.2009.11.002`. 9, 10, 14

[12] Kernighan, B. W.; Lin, Shen: *An efficient heuristic procedure for partitioning graphs.* Bell Systems Technical Journal, 49, 1970. 11, 16

[13] Barberá, Pablo: *Birds of the same feather tweet together: Bayesian ideal point estimation using twitter data.* Political Analysis, 23(1):76–91, 2015. 14

[14] Moura Brito, Isabelle Alves ; Carla Silva Oliveira ; José André de: *An analysis of the problem of community detection in networks.* Revista Eletrônica Sistemas e Gestão, 9(4), 2014. 15

[15] Silva, José André de Moura Brito, Carla Silva Oliveira Daiana Medeiros da: *Medidas de centralidade e detecção de comunidades em rede de co-autoria.* XIX Simpósio de pesquisa operacional e logística da marinha, 2011. 15

[16] Oliveira Claudia Justel, Diego Andrés de Barros Lima Barbosa Leonardo Borges Avelino Rarylson Freitas de Souza Camila Cristina Gomes Ferreira de: *Um estudo computacional comparativo entre algoritmos de agrupamento e de detecção de comunidades.* XLIII Simpósio Brasileiro de PESQUISA OPERACIONAL, 2018. 15

[17] *https://micans.org/mcl/.* 19