UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Letícia do Nascimento

DESENVOLVIMENTO DE UMA HEURÍSTICA PARA O PROBLEMA DA MAXIMIZAÇÃO DA MODULARIDADE POR DENSIDADE COM SINAIS

Florianópolis, Santa Catarina – Brasil
2023

Letícia do Nascimento

DESENVOLVIMENTO DE UMA HEURÍSTICA PARA O PROBLEMA DA MAXIMIZAÇÃO DA MODULARIDADE POR DENSIDADE COM SINAIS

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Prof. Dr. Rafael de Santiago

Florianópolis, Santa Catarina – Brasil
2023

Letícia do Nascimento

DESENVOLVIMENTO DE UMA HEURÍSTICA PARA O PROBLEMA DA MAXIMIZAÇÃO DA MODULARIDADE POR DENSIDADE COM SINAIS

Este(a) Trabalho de Conclusão de Curso foi julgado adequado(a) para obtenção do Título de Bacharel em Ciências da Computação, e foi aprovado em sua forma final pelo Programa de Graduação em Ciências da Computação do INE – Departamento de Informática e Estatística, CTC – Centro Tecnológico da Universidade Federal de Santa Catarina.

Florianópolis, Santa Catarina – Brasil, 28 de junho de 2023.

Prof. Lúcia Helena Martins Pacheco, Dr.
Coordenador(a) do Programa de
Graduação em Ciências da Computação

Banca Examinadora:

Prof. Dr. Rafael de Santiago
Orientador
Universidade Federal de Santa
Catarina – UFSC

Alvaro Junio Pereira Franco,
Universidade Federal de Santa Catarina –
UFSC

Pedro Belin Castellucci,
Universidade Federal de Santa Catarina –
UFSC

Dedico este projeto aos meus pais, pois graças ao seu esforço e apoio que hoje posso concluir o meu curso.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que abdicaram de muitas coisas para que eu pudesse me formar e ter um futuro melhor, e nunca mediram esforços para garantir minha formação profissional e pessoal.

Agradeço ao meu namorado, que sempre esteve ao meu lado me dando forças e apoio para continuar em qualquer situação.

Agradeço à universidade e todos os profissionais da educação pela excelência da qualidade técnica de cada um, em especial ao meu orientador, Dr. Rafael de Santiago, pela sua paciência, pelo empenho e dedicação em tornar este trabalho viável.

Por fim, agradeço a todos os meus amigos, colegas de turma e familiares que direta ou indiretamente contribuíram na minha jornada.

“Our passion for learning is our tool for survival.
Carl Sagan

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MM	Maximização da Modularidade
MMD	Maximização da Modularidade por Densidade
MMDS	Maximização da Modularidade por Densidade com Sinais
PSO	Particle Swarm Optimization
GRASP	Greedy Randomized Adaptive Search Procedure
HLSMDM	Hybrid Local Search heuristic for Modularity Density Maximization

SUMÁRIO

1 INTRODUÇÃO

Encontrar grupos bem relacionados de entidades na vida real é um problema útil em diversos domínios, e o interesse nesses estudos vem crescendo nos últimos anos. O problema pode ser aplicado na medicina, para identificar a memória funcional envolvida no reconhecimento olfativo (MEUNIER; FONLUPT *et al.*, 2014), na neurociência, para segmentação de redes neurais (MEUNIER; LAMBIOTTE; BULLMORE, 2010), na astronomia, para identificação de grupos de estrelas (SCHMEJA, 2011), na biologia, para encontrar complexos de proteínas (NEPUSZ; YU; PACCANARO, 2012), na identificação de comunidades em redes de transporte público (GUIMERÀ *et al.*, 2005), entre muitas outras aplicações.

Esses grupos são denominados na literatura de “comunidades” e são geralmente formalizados em grafos, nos quais cada vértice representa uma entidade e as relações entre elas são definidas por arcos ou arestas (NEWMAN; GIRVAN, 2004; LESKOVEC; HUTTENLOCHER; KLEINBERG, 2010; FORTUNATO; CASTELLANO, 2012). A relação entre entidades pode incorporar um peso positivo ou negativo. Um exemplo de conexão com sinal seria no estudo de mídias sociais no qual, onde as ligações podem representar uma relação positiva (de amizade) ou negativa (de antagonismo) entre os usuários. A atitude de um usuário em relação a outro pode ser estimada a partir de evidências fornecidas por seus relacionamentos com outros membros da rede social (LESKOVEC; HUTTENLOCHER; KLEINBERG, 2010).

Visando encontrar grupos bem relacionados em grafos, Mark Newman e Michelle Girvan definiram um problema de otimização de maximização da modularidade, cujo objetivo é medir a diferença entre o número de ligações internas e o número esperado de ligações internas dentro de um agrupamento (NEWMAN; GIRVAN, 2004). Por tratar-se de um problema NP-Difícil (BRANDES, 2008) e por diversas vezes as aplicações serem caracterizadas por instâncias de tamanho significativo (muitos vértices e arestas), o problema da Maximização da Modularidade é frequentemente resolvido por métodos heurísticos na literatura (CLAUSET; NEWMAN; MOORE, 2004; BLONDEL *et al.*, 2008; ROTTA; NOACK, 2011).

A modularidade, contudo, apresenta um problema chamado de limite de resolução (FORTUNATO; BARTHÉLEMY, 2007). Segundo Fortunato e Barthélemy, a solução ótima do problema tem os seus agrupamentos com um número máximo de vértices limitado em relação ao total de conexões do grafo. Tal problema impossibilita o reconhecimento de comunidades em redes cuja quantidade total de vértices está abaixo de um número mínimo de conexões esperadas.

Com a motivação principal sendo evitar a resolução limite, uma reformulação para o problema da Maximização da Modularidade foi desenvolvida (LI; ZHANG *et al.*, 2008), chamada de Maximização da Modularidade por Densidade. A reformulação utiliza a

diferença entre a densidade interna e externa de ligações para cada agrupamento, avaliando assim as soluções do problema. Sua variação para lidar com grafos com sinais foi proposta em Li, Liu e Liu (2014). Alguns métodos exatos e heurísticos foram propostos na literatura (LI; LIU; LIU, 2014; DE SANTIAGO; LAMB, 2020).

Este trabalho propõe uma heurística[1] para o problema da Maximização da Modularidade por Densidade com Sinais a partir do estado atual da arte, levantando os métodos computacionais presentes na literatura e analisando os resultados, comparando a heurística desenvolvida com os métodos existentes. A estratégia proposta compreende na aplicação de uma busca local monótona à solução inspirada por (CHEN *et al.*, 2017), e utiliza conceitos de busca gulosa, aleatória e adaptativa (GRASP, do inglês ***greedy randomized adaptive search procedure***. Algumas configurações de teste foram definidas, cujos resultados foram analisados e comparados com o estudo (DE SANTIAGO; LAMB, 2020).

1.1 OBJETIVOS

Esta seção apresenta o objetivo geral e objetivos específicos deste trabalho.

1.1.1 Objetivo geral

Propor uma heurística para o problema de maximização da modularidade por densidade com sinais.

1.1.2 Objetivos específicos

Para atingir o objetivo geral, deve-se cumprir os seguintes objetivos específicos:

- Especificar nova heurística para o problema de maximização da modularidade por densidade com sinais;
- Desenvolver nova heurística para o problema de maximização da modularidade por densidade com sinais;
- Analisar os resultados.

1.2 METODOLOGIA

A pesquisa da presente proposta é quantitativa e exploratória, pois visa propor uma heurística, analisando seu comportamento quanto ao valor de avaliação e da eficiência em tempo computacional do método proposto.

Para desenvolver a presente pesquisa, executaram-se as seguintes etapas:

[1] Vale ressaltar que na literatura científica consultada, não se conhece um algoritmo exato de tempo polinomial determinístico para resolver o problema.

Primeiramente, foi efetuado um levantamento teórico com intuito de fornecer maior credibilidade à pesquisa. Esta etapa foi realizada com a busca de artigos e livros no portal da SCOPUS, utilizando algumas palavras-chave como ***"community detection"***, ***"clustering"***, ***"graph clustering"*** e ***"community structure"***, tendo em vista que grande parte do conteúdo da área é produzido em inglês. A partir dos estudos encontrados na busca, foram escolhidos três trabalhos cujo critério de seleção foi a relevância de acordo com o SCOPUS e passando por uma filtragem da autora, buscando garantir que os trabalhos sejam relacionados com a pesquisa aqui desenvolvida.

Concluída a primeira etapa, o próximo passo foi a especificação da heurística. Com a especificação concluída, foi realizado o desenvolvimento da heurística na linguagem Python. A escolha da ferramenta foi com base na afinidade pela mesma.

A última etapa consiste na análise dos resultados, na qual obtiveram-se informações sobre o desempenho do método proposto, comparando os resultados obtidos com outros da literatura. Mais especificamente, os resultados foram comparados com um dos trabalhos relacionados que, a partir do desenvolvimento de um algoritmo exato, obteve resultados ótimos em seus experimentos.

Para a consecução dos objetivos supracitados, é apresentada no Capítulo 2 a fundamentação teórica do trabalho, dando inicialmente uma visão geral sobre grafos e detecção de comunidades, em seguida são explorados os conceitos que circundam a Maximização da Modularidade e Maximização da Modularidade por Densidade com Sinais. Após isso, ainda no mesmo capítulo, é explicado sobre Heurísticas e Metaheurísticas e como elas são utilizadas nos problemas de otimização. Posteriormente, são descritos no Capítulo 3 os pontos principais de três trabalhos relacionados, onde dois dos mesmos foram utilizados como base para o desenvolvimento da proposta da heurística desenvolvida. A seguir, no Capítulo 4, apresenta-se a proposta do trabalho e o seu desenvolvimento, abordando cada etapa em detalhes descritos por algoritmos além de todos os experimentos e análises levantadas a partir dos mesmos. Por fim, no Capítulo 5, conclui-se o trabalho desenvolvido, retomando brevemente o assunto abordado, discutindo resultados obtidos e finalizando com trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

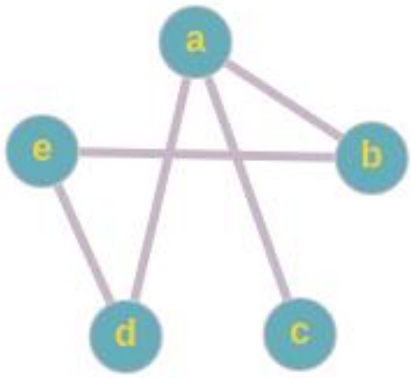
O capítulo de fundamentação teórica consiste na contextualização dos fundamentos da Maximização da Modularidade por Densidade com Sinais, bem como um breve histórico de evolução da detecção de comunidades, aplicações e importância. Para isso, apresenta-se na Subseção 2.1, inicialmente, os conceitos fundamentais para detecção de comunidades. Em seguida, na Subseção 2.1.1 definimos a Modularidade e apresentamos uma das suas fragilidades. Na Subseção 2.1.2, definimos a Modularidade por Densidade e Modularidade por Densidade com Sinais. Por fim, na Subseção 2.1.3 definimos a importância das heurísticas e metaheurísticas no desenvolvimento de soluções para problemas de Maximização da Modularidade.

2.1 DETECÇÃO DE COMUNIDADES

Para definir comunidades, precisamos primeiro definir grafos. Grafos são estruturas formadas por um grupo de vértices (também chamados de nós) e um grupo de arestas, que são conexões entre pares de vértices (FORTUNATO; CASTELLANO, 2012). Em outras palavras, podemos definir um grafo $G = (V, E)$, onde V é o conjunto finito de objetos chamados vértices e E são pares não ordenados de vértices, chamados de arestas (DIESTEL, 2017). Grafos correspondem a uma rede, e sua representação visual é feita por linhas (arestas) conectando pontos do plano (vértices), como na Figura 1.

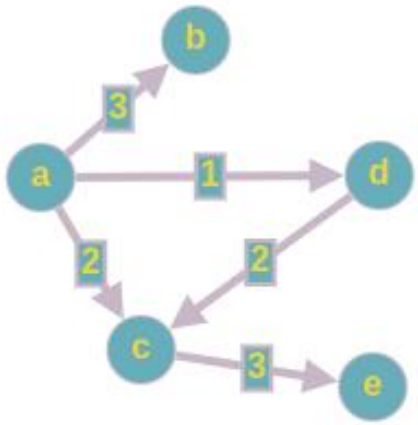
Outro conceito importante é o grau de um vértice. O grau de um vértice é definido como o número de arestas incidentes a esse vértice. Em outras palavras, o grau de um vértice é a contagem de quantas arestas estão conectadas a ele. Para grafos

Figura 1 – Exemplo de grafo.



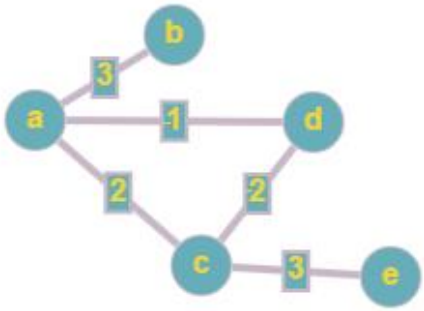
Fonte: Autoria própria.

Figura 2 – Exemplo de digrafo com sinal.



Fonte: Autoria própria.

Figura 3 – Exemplo de grafo com sinal.



Fonte: Autoria própria.

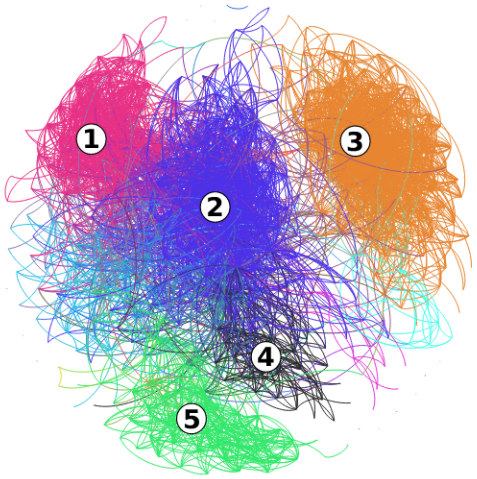
direcionados, seu grau de entrada é o número de arestas que apontam para esse vértice, e o grau de saída é o número de arestas que partem desse vértice. Já para grafos não direcionados, o grau de um vértice é igual à soma do grau de entrada e do grau de saída, pois todas as arestas são bidirecionais (DIESTEL, 2017).

Inúmeros problemas nas mais diversas áreas podem ser representados por grafos. Um grafo pode ser associado a qualquer conjunto no qual há uma definição de uma relação binária, como a relação de uma árvore genealógica (a é filho de b).

Comunidades são subconjuntos de vértices que estão densamente conectados entre si, diferente da sua relação com os demais vértices do mesmo grafo (LI; ZHANG *et al.*, 2008). Comunidades podem ser encontradas, por exemplo, em redes telefônicas, onde é possível realizar uma análise para identificar grupos criminosos com base nos metadados de contatos telefônicos (FERRARA *et al.*, 2014).

Um tópico de grande interesse no estudo de redes complexas é a identificação

Figura 4 – Exemplo de comunidades.



Fonte: Santiago e Luís C Lamb (2017)

de comunidades. Esse é um tema de grande importância para análise de dados e que tem tido uma atenção relevante nos últimos anos (FERRARA *et al.*, 2014; MEUNIER; FONLUPT *et al.*, 2014; ZHOU; WANG, 2016; CHEN *et al.*, 2017; SANTIAGO, R. de; LAMB, 2020), uma vez que permite uma visão da relação funcional e estrutural de uma rede, e possui abrangentes áreas de aplicação.

Resolver problemas de detecção de comunidades em redes complexas pode ser um trabalho árduo por ser um problema NP-Difícil (BRANDES, 2008). Na seções seguintes, são apresentados problemas de otimização na literatura que lidam com a detecção de comunidades.

2.1.1 Maximização da Modularidade

O problema da Maximização da Modularidade (MM) foi formalizado por Newman e Girvan, onde o objetivo do seu estudo era desenvolver uma função para avaliar a qualidade de uma partição de uma rede em comunidades (NEWMAN; GIRVAN, 2004). A modularidade é uma medida que quantifica a estrutura de comunidades em uma rede, e o conceito básico por trás da MM é encontrar uma divisão da rede em comunidades que maximize a modularidade total. Esse estudo propôs duas heurísticas para resolução do problema da MM, chamados de "***shortest-path betweenness***" e "***random walks***".

Na primeira abordagem de desenvolvimento do problema da MM, a função foi implementada apenas para grafos não-dirigidos, e nenhuma identificação de sinal ou qualquer ponderação foram considerados. No entanto, existem algumas variações dela na literatura. A função de modularidade para uma dada partição C é dada pela Equação 1. O grafo $G = (V, E)$ é a entrada do problema, onde V é o conjunto de vértices, e E o conjunto de arestas. O grau do vértice $i \in V$ é dado por d_i. A partição

C é um conjunto de comunidades separadas. Um maior valor para Q indica uma boa estrutura de comunidade.

$$Q(C) = \frac{1}{2|E|} \sum_{c \in C} \sum_{i,j \in c} \left(a_{ij} - \frac{d_i d_j}{2|E|} \right) \tag{1}$$

Na qual:

$Q(C)$: É a medida de modularidade para a configuração de comunidades C. A modularidade é uma medida que avalia a qualidade da divisão de uma rede em comunidades. Quanto maior o valor de Q, melhor é a divisão em comunidades.

$|E|$: É o número total de arestas na rede, representando o tamanho do conjunto de todas as arestas na rede.

$c \in C$: Representa uma comunidade específica dentro da configuração de comunidades C.

$i, j \in c$: Representam os vértices pertencentes à comunidade c.

a_{ij}: É o número de arestas entre os vértices i e j, representando o grau de conexão entre esses vértices.

d_i: É o grau do vértice i, ou seja, o número de arestas que incidem no i.

d_j: É o grau do vértice j.

A fórmula calcula a modularidade (Q) somando, para cada comunidade c na configuração C, a diferença entre a proporção real de arestas (a_{ij}) dentro da comunidade c e a proporção esperada de arestas entre os vértices i e j com base nos graus dos vértices (d_i, d_j). Essa diferença é ponderada pela metade do número total de arestas na rede ($|E|$).

A modularidade mede a densidade[1] de conexões dentro das comunidades em relação às expectativas aleatórias. Valores positivos de Q indicam que a divisão das comunidades é melhor do que esperado aleatoriamente, enquanto valores negativos indicam que a divisão é pior do que esperado. O objetivo é encontrar uma configuração de comunidades que maximize a modularidade.

No entanto, já foram encontradas algumas fragilidades na fórmula desenvolvida. Uma delas trata-se da “resolução limite” (FORTUNATO; BARTHÉLEMY, 2007). Fortunato e Barthélemy mostram que há uma escala relativa ao número de conexões de uma rede a ser respeitada, onde caso o número de conexões de um grafo for menor que essa escala, o mesmo pode não ser resolvido pelo problema da MM - ainda que seja um grafo cujas comunidades se conectam por uma única aresta. Portanto, o problema da resolução limite depende do grau de interconectividade entre pares de comunidades, impossibilitando afirmar quando uma comunidade, obtida através da MM, é

[1] A densidade considerada neste trabalho é a proporção de arestas em um grafo em relação a quantidade total possível. Considerando n o número de vértices, sabe-se que o total possível de arestas em um grafo (não-multigrafo) é $\frac{n^2-n}{2}$.

de fato uma única comunidade ou um grupo de comunidades menores. Estudos foram realizados a fim de evitar os problemas encontrados, tais como (KARRER; NEWMAN, 2011), (LANCICHINETTI; FORTUNATO; RADICCHI, 2011) e (TRAAG; VAN DOOREN, 2011).

2.1.2 Maximização da Modularidade por Densidade com Sinais

A Maximização da Modularidade por Densidade (MMD) (LI; ZHANG *et al.*, 2008) surgiu com a motivação de evitar o problema da resolução limite da MM (FORTUNATO; BARTHÉLEMY, 2007), utilizando o conceito de grau médio de modularidade, chamada de medida quantitativa da densidade de modularidade. A MMD é uma medida alternativa de modularidade que considera a densidade das comunidades em relação à densidade global da rede. Em contrapartida à fórmula da modularidade original, que utiliza a diferença entre o número real e o número esperado de arestas, a modularidade por densidade considera a razão entre essas duas quantidades, permitindo que a medida seja menos influenciada pela densidade global da rede e mais sensível a comunidades menos densas em redes esparsas. Uma variação da modularidade por densidade para lidar com grafos com sinais foi proposta em (LI; LIU; LIU, 2014), chamada de Maximização da Modularidade por Densidade com Sinais (MMDS). Sua função é dada na Equação 2. Nela, considera-se que o grafo de entrada para o problema possui arestas com valores positivos (representando atração) e negativas (representando repulsa).

Dado um grafo com sinal G e uma comunidade C, considere $G = (V, E^+, E^-)$, onde V é o conjunto de vértices, E^+ é o conjunto de arestas positivas, e E^- é o conjunto de arestas negativas. Considere que $w^+_{uv} = 1$ se $\{u, v\} \in E^+$, caso contrário $w^+_{uv} = 0$, e $w^-_{uv} = 1$ se $\{u, v\} \in E^-$, caso contrário $w^-_{uv} = 0$. Os conjuntos E^+_c e E^-_c são compostos pelos vértices positivos e negativos, respectivamente, de uma comunidade $c \in C$. Portanto, $|E^+_c|$ representa o número de arestas positivas (ou arestas dentro da comunidade) em uma determinada comunidade c e $|E^-_c|$ é o número de arestas negativas (ou arestas fora da comunidade) em uma determinada comunidade c. Dividimos o grau de cada vértice entre positivos e negativos. Logo, o grau positivo de v é dado por $d^+_v = \sum_{u \in V} w^+_{uv}$ (soma dos graus positivos), e o grau negativo é dado por $d^-_v = \sum_{u \in V} w^-_{uv}$ (soma dos graus negativos). A somatória $\sum_{c \in C}$ indica que a fórmula deve ser avaliada para cada comunidade no conjunto. O número de vértices na comunidade c é dado por $|c|$.

$$D_\lambda(C) = \sum_{c \in C} \left(\frac{2\lambda|E^+_c| - 2(1-\lambda)(\sum_{v \in c} d^+_v - |E^+_c|)}{|c|} \right. \\ \left. \frac{-2(1-\lambda)|E^-_c| + 2\lambda(\sum_{v \in c} d^-_v - |E^-_c|)}{|c|} \right). \tag{2}$$

Nessa função, $D_\lambda(C)$ é a medida de modularidade de identidade para o conjunto de comunidades C com um parâmetro de ajuste λ. O parâmetro λ é usado para obter a "associação de proporção" para encontrar comunidades pequenas quando $\lambda > 0.5$, e o "recorte de proporção" para encontrar comunidades grandes quando $\lambda < 0,5$. O estudo (LI; ZHANG *et al.*, 2008) sugeriu que esta função pode ser usada para encontrar o nível apropriado da estrutura topológica de grafos para encontrar partições apropriadas. Em outras palavras, λ é um parâmetro de ajuste que controla o equilíbrio entre as contribuições das arestas positivas e negativas para a modularidade.

Em resumo, a fórmula em si calcula a modularidade de identidade para cada comunidade no conjunto C e, em seguida, soma as contribuições de todas as comunidades para obter o valor total da modularidade de identidade $D_\lambda(C)$. A fórmula considera tanto as arestas positivas quanto as arestas negativas, e o parâmetro λ permite ajustar a importância relativa dessas duas contribuições.

2.1.3 Heurísticas e Metaheurísticas

Não se conhece um algoritmo exato de tempo polinomial para resolver o problema da Maximização da Modularidade na literatura, por ser um problema NP-Difícil (BRANDES, 2008). Assim, o problema da Maximização da Modularidade é frequentemente resolvido por métodos heurísticos.

O uso de heurísticas e metaheurísticas para resolver problemas do mundo real é amplamente aceito na comunidade de pesquisa operacional. Sabemos que a grande maioria dos problemas complexos de decisão do mundo real, quando modelados como problemas de otimização, pertencem à classe de problemas NP-difíceis. Isto implica que abordagens exatas estão fadadas ao fracasso ao lidar com grandes instâncias de escala real, sejam elas provenientes de negócios, engenharia, economia ou ciência. Hoje, os processos de tomada de decisão são cada vez mais complexos e mais abrangentes, no sentido de que mais variáveis de decisão são usadas para modelar situações complexas e mais dados de entrada e parâmetros estão disponíveis para capturar a complexidade dos próprios problemas (MANIEZZO; STÜTZLE; VOSS, 2009).

Heurísticas são métodos computacionais que não garantem a solução ótima, mas buscam soluções com características presentes nas melhores soluções. As heurísticas podem parar depois de encontrar soluções chamadas de ótimas locais. As ótimas locais são soluções de referência em uma região no espaço de busca. Por se tratarem da melhor solução da região, pode dificultar que soluções melhores sejam encontradas. Diz-se então que a heurística "ficou presa" em um ótimo local (ROTHLAUF, 2011).

Para evitar que ótimos locais impeçam buscas por melhores soluções, utiliza-se de metaheurísticas. Elas utilizam-se de abordagens conhecidas como intensificação e diversificação. A intensificação explora o espaço de busca, de maneira limitada,

procurando por soluções melhores, ou seja, “intensificadas”. A diversificação acontece para impedir que um espaço muito limitado de busca seja explorado. Desse modo, uma metaheurística pode se utilizar de mais de uma estratégia heurística (GENDREAU; POTVIN, 2010).

Existem dois tipos de heurísticas clássicas: construção e melhoria. Uma heurística de construção, parte de uma solução vazia e utiliza-se de um procedimento iterativo que seleciona partes da solução para compô-la. O procedimento termina quando a solução estiver completa. Uma heurística de melhoria parte de uma solução inicial completa e tenta iterativamente melhorar a solução. Geralmente, heurísticas de melhoria terminam quando um ótimo local é atingido. A heurística aplicada no presente trabalho utiliza os conceitos de construção e melhoria em seu desenvolvimento.

O teorema “não há almoço grátis” (WOLPERT; MACREADY, 1997) demonstra que não é possível definir uma heurística de propósito geral que funciona bem para todos os problemas de otimização. O estudo conclui que para qualquer algoritmo, qualquer desempenho elevado em uma classe de problemas é compensado pelo desempenho em outra classe.

Nesse contexto, o seguinte capítulo traz os principais conceitos de alguns trabalhos relacionados que utilizam abordagens heurísticas no desenvolvimento dos seus estudos para tratar o problema da Maximização da Modularidade por Densidade com Sinais.

3 TRABALHOS RELACIONADOS

Utilizando a ferramenta do SCOPUS para pesquisar sobre Maximização da Modularidade por Densidade, foi possível encontrar alguns estudos recentes sobre o tema. Nessa seção serão apresentados algoritmos desenvolvidos para identificar comunidades em redes com sinal utilizando algoritmo guloso (CHEN *et al.*, 2017), partículas (PSO) (ZHOU; WANG, 2016) e algoritmo ***branch-and-price*** (SANTIAGO, R. de; LAMB, 2020).

Para identificar comunidades em redes com sinal, um novo algoritmo guloso foi explorado (CHEN *et al.*, 2017) levando em consideração os sinais e a densidade dessas ligações. A ideia principal do algoritmo é o procedimento inicial de modularidade por densidade com sinais e as regras de atualização correspondentes. Especialmente, empregaram os procedimentos "***Asymmetric and Constrained Belief Evolution***" para avaliar o número ótimo de comunidades. De acordo com resultados experimentais, o algoritmo provou ser capaz de funcionar bem. Mais especificamente, o algoritmo proposto é muito eficiente para redes de tamanho médio, tanto densas quanto esparsas.

Outro estudo aborda a otimização de enxame de partículas (PSO) (ZHOU; WANG, 2016) no problema de detecção de comunidades e propõe um algoritmo baseado em estratégia de rótulo. O PSO é um algoritmo que pode ser utilizado para encontrar comunidades em redes complexas, e é um método metaheurístico que se baseia em simular o comportamento de um enxame de partículas em busca de soluções ótimas. O método foi inspirado no comportamento de um enxame de pássaros ou insetos, e cada partícula representa uma possível divisão da rede em comunidades, onde a posição de uma partícula corresponde às atribuições de comunidade para cada vértice da rede (KENNEDY; EBERHART, 1995). Em contraste com outras estratégias de propagação de rótulos, a principal contribuição deste estudo é a introdução do conceito de "impacto da vizinhança" na função de avaliação do PSO, e colocá-lo em uso. Esse impacto da vizinhança mede a influência dos vértices vizinhos na determinação das comunidades, quanto maior o impacto da vizinhança, mais forte é a influência dos vizinhos na atribuição de comunidades. Experimentos em redes sintéticas e reais mostram a eficácia da estratégia proposta. Além disso, esta estratégia é estendida para redes com sinal, e a função objetivo correspondente é definida na Subseção 2.1.2.

Ainda, mais um algoritmo foi proposto (SANTIAGO, R. de; LAMB, 2020) para encontrar soluções para o problema da Maximização da Modularidade por Densidade com Sinais. Devido ao número exponencial de variáveis do problema exato, o método ***branch-and-price*** usa um procedimento de geração de colunas para descobrir variáveis importantes, e então os valores inteiros são encontrados para eles. O resultado dessa pesquisa é importante, pois permite a identificação de comunidades em redes

complexas de uma maneira precisa e eficiente.

3.1 DESCOBRINDO ESTRUTURAS DE COMUNIDADES EM REDES SOCIAIS BASEADAS EM OTIMIZAÇÃO GANANCIOSA

O estudo (CHEN *et al.*, 2017) tem como objetivo focar na análise de redes sociais com sinal e na identificação de suas estruturas de comunidades utilizando um algoritmo de otimização guloso.

Nas redes sociais, indivíduos ou entidades são conectados por arestas que representam relacionamentos ou interações entre eles. Esses relacionamentos podem ser positivos (indicando amizade, colaboração, etc.) ou negativos (indicando inimizade, discordância, etc.), formando uma rede social com sinal. Compreender a estrutura comunitária dentro dessas redes é essencial para várias aplicações, como identificar indivíduos influentes, prever comportamentos de usuários ou estudar a difusão de informações.

Os autores propõem um método baseado em otimização gananciosa para revelar a estrutura de comunidades em redes sociais, representadas por redes com sinal. A otimização gananciosa é uma estratégia que faz escolhas localmente ótimas de forma iterativa para resolver um problema.

O algoritmo guloso explorado para identificar comunidades (CHEN *et al.*, 2017) é muito eficiente para buscar as mudanças na modularidade ΔQ_{ij} que estão relacionados com a fusão entre cada par de comunidades. A função objetivo e restrição do problema de otimização do algoritmo é dada pela Equação 3:

$$max\ Q\ s.t.\ \exists \lambda Q_{ij} > 0 (\forall i, j). \tag{3}$$

Na qual:

$max\ Q$: indica que o objetivo é maximizar o valor de Q, que representa a medida de modularidade usada para avaliar a qualidade das estruturas da comunidade.

$s.t.$: significa "sujeito a", indicando que a seguinte condição é uma restrição no problema de otimização.

$\exists \lambda Q_{ij} > 0$: especifica a restrição de que deve existir pelo menos um par de vértices i e j para o qual a mudança na modularidade (ΔQ_{ij}) é maior que zero. Em outras palavras, garante que haja potenciais fusões de comunidades que possam levar a uma melhoria na modularidade.

$\forall i, j$: denota a quantificação universal sobre todos os pares possíveis de vértices i e j, o que significa que a restrição se aplica a todos os pares na rede.

Ou seja, a fórmula representa o objetivo de maximizar a modularidade (Q) enquanto garante que existam pares de vértices com mudanças positivas na modularidade (ΔQ_{ij}) disponíveis para fusões de comunidades. Este problema de otimização visa

encontrar a melhor estrutura de comunidade que maximize a modularidade por meio da fusão iterativa de comunidades com valores ΔQ_{ij} positivos.

1. Inicialmente, cada vértice é considerado como o único membro de uma comunidade. Então é calculado ΔQ_{ij} dependendo da Equação 2 caso i e j possuam arestas, e o valor é zero caso i e j não possuam arestas. Então, calcula-se ΔQ_{ij} como se segue. Considerando que w^+ é o peso total das arestas positivas da rede e w^- é o peso total das arestas negativas da rede. Sendo v um vértice da rede, w_v^+ é o peso total das arestas positivas conectadas no vértice v, e w_v^- o peso total das arestas positivas conectadas no vértice v.

   Caso as arestas dos vértices i e j sejam positivas,

$$\Delta Q_{ij} = \frac{w^+}{w^+ + w^-}\left(\frac{1}{w^+} - \frac{w_i^+ w_j^+}{(w^+)^2}\right) + \frac{w^-}{w^+ + w^-}\frac{w_i^- w_j^-}{(w^-)^2}. \tag{4}$$

   Caso as arestas dos vértices i e j sejam negativas,

$$\Delta Q_{ij} = -\frac{w^+}{w^+ + w^-}\frac{w_i^+ w_j^+}{(w^+)^2} - \frac{w^-}{w^+ + w^-}\left(\frac{1}{w^-} - \frac{w_i^- w_j^-}{(w^-)^2}\right). \tag{5}$$

   Do contrário,

$$\Delta Q_{ij} = 0. \tag{6}$$

   Podemos obter a matriz ΔQ pelas Equações 4 e 5, e calcular para cada vértice i por $\frac{w_i^+}{w^+}$ e $\frac{w_i^-}{w^-}$. A partir disso, dois $arrays$ (ou grupos) de vetores podem ser expressados.

2. Para cada i, j na matriz ΔQ, escolher o maior elemento entre $\Delta Q_{i,j}$. No caso do resultado for maior que 0, converter para o passo 3, do contrário, parar.

3. Combinar as comunidades correspondentes e atualizar a matriz ΔQ e $\frac{w_j^+}{w^+}$, $\frac{w_j^-}{w^-}$ como segue. No fim, voltar para o passo 2.

   Quando o vértice k se combina com os vértices i e j, ΔQ_{jk} é atualizado como:

$$\Delta Q_{jk} = Q_{ik} + Q_{jk} \tag{7}$$

Se a comunidade k estiver conectada apenas ao vértice i, mas não a j, então $\Delta Q'_{jk}$ é atualizado como:

$$\Delta Q'_{jk} = Q_{ik} - 2\left(\frac{w^+}{w^+ + w^-}\frac{w_j^+}{w^+}\frac{w_k^+}{w^+} - \frac{w^-}{w^+ + w^-}\frac{w_j^-}{w^-}\frac{w^-}{w^-}\right). \tag{8}$$

Se a comunidade k se combina apenas com o vértice j, mas não com i, então $\Delta Q'_{jk}$ é atualizado como:

$$Q'_{jk} = Q_{jk} - 2\left(\frac{w^+}{w^+ + w^-}\frac{w_i^+}{w^+}\frac{w_k^+}{w^+} - \frac{w^-}{w^+ + w^-}\frac{w_i^-}{w^-}\frac{w^-}{w^-}\right). \tag{9}$$

Os pesos $\frac{w_j^+}{w^+}$ e $\frac{w_j^-}{w^-}$ são atualizados como:

$$\left(\frac{w_j^+}{w^+}\right) = \frac{w_j^+}{w^+} + \frac{w_j^+}{w^+} \tag{10}$$

e

$$\left(\frac{w_j^-}{w^-}\right) = \frac{w_j^-}{w^-} + \frac{w_j^-}{w^-} \tag{11}$$

Ou seja, o algoritmo repete os passos 2 e 3 até que nenhuma melhoria adicional na modularidade seja observada. Seguindo essas etapas, o algoritmo ganancioso mescla iterativamente as comunidades, selecionando os pares que resultam no maior aumento na modularidade. Ele atualiza os elementos relevantes na matriz ΔQ e os pesos associados às comunidades para refletir as mudanças. Dessa forma, o algoritmo busca maximizar a medida de modularidade, otimizando gananciosamente a estrutura de comunidades na rede.

3.2 ALGORITMO DE DETECÇÃO DE COMUNIDADE BASEADO EM IMPACTO DE VIZINHANÇA VIA PSO DISCRETO

Na abortagem do artigo (ZHOU; WANG, 2016), a modularidade por densidade com sinais é utilizada como função objetivo (definida na Subseção 2.1.2) para identificar comunidades em redes complexas. Para resolver o problema de detecção de comunidades, o estudo utiliza uma técnica chamada PSO discreto (do inglês ***Particle Swarm Optimization***), que é uma técnica inspirada no comportamento de enxames de partículas. O PSO é usado para otimização e busca de soluções em espaços de alta dimensionalidade. No estudo, a detecção de comunidades é realizada considerando o impacto das vizinhanças de cada vértice na comunidade. A motivação é que os vértices em uma comunidade tenham conexões mais fortes com seus vizinhos dentro da comunidade do que com vértices externos.

No estudo, a propagação de rótulos é empregada (GONG *et al.*, 2014). O rótulo do vértice é usado para atribuir o vértice para diferentes comunidades, e os vértices que têm o mesmo rótulo são considerados na mesma comunidade. Esta estratégia de propagação de rótulos considera o número de vértices com o mesmo rótulo na vizinhança para atualizar o atual rótulo do vértice. Por exemplo, considerando um vértice i cujos vizinhos são $i^1, i^2, ...i^n$, ao usar esta propagação de rótulo para atualizar o rótulo do vértice i, verifica-se todos os rótulos de $i^1, i^2, ...i^n$ para encontrar o rótulo que aparece mais, e então o vértice i atribuído a este rótulo. Na verdade, a contribuição de cada vértice para a vizinhança não é a mesma que o vértice que é escolhido. O artigo propõe uma definição do "impacto do vértice" e uma nova propagação de rótulos baseada na definição criada.

Para uma visão geral do estudo, considere um vértice i cujos vizinhos são $i^1, i^2, ...i^k$; então o impacto do vértice i^j no vértice i pode ser definido da seguinte forma:

$$imp(i^j) = grau(i) * num(l(i^j)), \tag{12}$$

onde imp é o impacto do vértice, $grau(i^j)$ é i grau do vértice i^j, $l(i^j)$ referencia o rótulo do vértice e $num(l(i^j))$ referencia o número de vizinhos de i que possuem o mesmo rótulo do vértice i. O impacto do vértice geralmente é diferente quando o vértice escolhido está na vizinhança de vértices diferentes. Para redes com sinal, consideram-se apenas as conexões positivas uma vez que o número de conexões negativas em uma comunidade é geralmente menor do que as ligações negativas entre comunidades, e dois vértices conectados com conxexões negativas geralmente estão localizados em diferentes comunidades.

O imp descreve a eficácia do vértice na rede, e o valor do imp pode mostrar o nível da conexão na rede local correspondente. Quanto maior o imp é, mais forte será a conexão. Assim, o imp pode ser usado como medida para detectar a qual comunidade o vértice pertence.

No algoritmo PSO (***Particle Swarm Optimization***), um bom esquema de inicialização pode gerar um conjunto de alta qualidade de soluções e reduzir significativamente o tempo de busca. O método tradicional de inicialização gera soluções aleatoriamente. Isto não leva a matriz de adjacência da comunidade em consideração, enquanto esta matriz geralmente fornece informações para a otimização.

Em resumo, o estudo propõe um algoritmo baseado em PSO discreto para detectar comunidades em redes complexas, considerando o impacto das vizinhanças dos vértices. O algoritmo busca maximizar o impacto das conexões internas da comunidade e pode ser aplicado em várias áreas para análise de redes e identificação de estruturas de comunidades. O estudo apresenta resultados experimentais demonstran-

do a eficácia do algoritmo proposto em detectar comunidades em redes complexas, comparando-o com outros métodos existentes.

3.3 SOLUÇÃO PARA A MODULARIDADE POR DENSIDADE COM SINAIS E ANÁLISE DAS SOLUÇÕES ÓTIMAS

Este estudo aborda a maximização exata da densidade de modularidade com sinais em redes complexas e explora o significado real das soluções encontradas. Seu objetivo é encontrar soluções ótimas que maximizem a densidade de modularidade, levando em consideração os sinais das conexões entre os vértices da rede.

O algoritmo ***branch-and-price*** desenvolvido (DE SANTIAGO; LAMB, 2020) utiliza um método de geração de colunas que deriva de um modelo linear inteiro que foi inspirado em modelos anteriores similiares para problemas envolvendo redes sem sinais (ALOISE *et al.*, 2010; SANTIAGO; LAMB, 2017; DE SANTIAGO; LAMB, 2020).

A geração de colunas encontra a solução ótima em espaço de solução contínua. Por essa razão, foi utilizado um algoritmo para encontrar a solução ótima em um espaço de soluções inteiras. O método ***branch-and-price*** é relatado no Algoritmo 1.

O algoritmo começa definindo o problema mestre reduzido. As variáveis iniciais são definidas por um método de busca local de duas etapas de (SANTIAGO; LAMB, Luís C, 2017) (***Hybrid Local Search heuristic for Modularity Density Maximization (HLSMDM-λ)***) adaptado para redes com sinal. Em seguida, a pilha, ou a *heap* que guarda o maior valor D no método ***branch-and-price*** é definida.

A variável ***lowerbound*** (limite inferior) armazena o valor objetivo da melhor solução encontrada. A cada iteração, o vértice ***branch-and-price*** com o valor D mais alto é obtido pela ***heap***. Se a solução deste vértice tiver um valor D pior do que o valor *lowerbound*, seus vértices irmãos não encontrarão uma solução melhor e o ***branch-and-price*** para. Se a procura não parar, o vértice selecionado é submetido ao procedimento de geração de colunas.

O procedimento de geração de colunas começa adicionando as restrições do vertice ***branch-and-price*** ao *RM*. A cada iteração da geração da coluna, uma nova coluna é adicionada ao modelo usando o modelo *Auxiliar*. Durante a seleção de uma nova coluna, o modelo *RM* é resolvido, e os valores para as variáveis primárias *z*, as variáveis duais *y* e o valor objetivo *Dens* são armazenados. Então o modelo *Auxiliar* é resolvido, e se uma nova variável for encontrada, ela é inserida no modelo *RM*. Após o loop, se a solução encontrada for inteira, a variável ***lowerbound*** é atualizada ou não. Caso não seja, são criados dois vértices ***branch-and-price***. A variável z que possui o maior valor de fração tem seu valor fixado em 0 para um vértice e 1 para o outro vértice.

Algoritmo 1 Branch-and-price

```
1: Criar o modelo reduced master RM com variáveis iniciais dadas por HLSMD_λ±
2: heap = new FibonacciHeap()
3: // construindo o vértice inicial do branch-and-price
4: heap.push((D = −∞, cons = {}))
5: lowerBound = −∞
6: stop = false
7: while (not stop) do
8:     data = heap.pop()
9:     if ( data.D < lowerBound ) then
10:        stop = true
11:    end if
12:    if (not stop) then
13:        columnGeneration (RM, data, lowerbound)
14:    end if
15:    if heap.empty() then
16:        stop = true
17:    end if
18: end while
```

Por fim, este estudo apresenta uma abordagem exata para maximizar a densidade de modularidade com sinais em redes complexas. Nele, um algoritmo de busca exata é desenvolvido e são realizadas análises para compreender o significado real das soluções encontradas. Os resultados mostram que a abordagem proposta é eficaz na maximização da densidade de modularidade e na interpretação das soluções em termos das características da rede.

4 MÉTODO PROPOSTO

Este trabalho consiste na implementação de uma heurística inspirada no artigo apresentado na Seção 3.1. Neste artigo, utiliza-se um algoritmo guloso para resolver o problema da Maximização da Modularidade por Densidade com Sinal. A estratégia proposta compreende na aplicação de uma busca local monótona sobre uma solução construída por um algoritmo de construção. Um procedimento de busca guloso, aleatório e adaptativo (GRASP) é uma meta-heurística multi-partida que aplica o método de busca local repetidamente a partir de soluções construídas por um algoritmo guloso aleatório (FEO; RESENDE, 1989).

4.1 DESENVOLVIMENTO DO GRASP PROPOSTO

Um pseudocódigo para o GRASP proposto pode ser visualizado no Algoritmo 2. Inicialmente, o algoritmo define uma variável S^* que armazenará a melhor solução encontrada na busca. Depois, o algoritmo gera m soluções (uma para cada repetição no laço que compreende as linhas 6 e 13). Para cada solução, seleciona-se iterativamente um componente da lista de vértices do grafo aleatorizada e os coloca na melhor solução possível com o Algoritmo 3. Quando a solução estiver completa, uma busca local é aplicada a solução recém criada (linha 9, Algoritmo 4) que obterá a ótima local S'. Ao final, verifica-se a solução encontrada durante a i-ésima iteração (variável do laço da linha 5) realizando uma comparação com a solução de referência S^*, que é substituída por S' caso essa última seja melhor que a de referência (linhas 11 e 12). Ao final, a busca retorna S^* (linha 15).

Algoritmo 2 GRASP proposto

1: **Entrada:** um grafo não-dirigido e ponderado $G = (V, E, w)$
2: *// Melhor solução encontrada até o momento*
3: $S^* = \{\}$
4: *// Gera m soluções*
5: **for all** $i \in 1, 2 \ldots, m$ **do**
6: *// Cria-se uma solução S que busca a melhor solução para cada vértice. A ordem de vértices a ser analisada é sempre aleatória*
7: $S =$ ConstruirSolução(G)
8: *// Cria-se uma solução S' que busca a melhor solução local para a solução criada acima*
9: $S' =$ BuscaLocal(G, S)
10: *// Guarda a solução gerada, caso seja a melhor encontrada*
11: **if** $D_\lambda(S')$ melhor que $D_\lambda(S^*)$ **then**
12: $S^* = S'$
13: **end if**
14: **end for**
15: **return** S^*

O Algoritmo 3 na etapa de exploração, busca a melhor comunidade para cada vértice, tendo para cada solução criada uma lista aleatória de vértices. Inicialmente, o algoritmo define uma variável S^* que armazenará a melhor solução encontrada. A primeira solução guarda apenas o próprio vértice que está sendo analisado em uma comunidade. Em seguida, o algoritmo busca o próximo vértice da lista aleatória e o adiciona na comunidade do vértice anterior. A cada passo, é aplicada a função da modularidade para encontrar e armazenar o valor da densidade de cada solução. Caso a densidade para o vértice em uma comunidade isolada seja melhor, o mesmo é mantido na comunidade isolada. No entanto, se o valor da densidade for melhor para o vértice incluso em uma outra comunidade com outros vértices, ele é mantido na comunidade desse conjunto. Assim é construída uma solução, buscando, para cada vértice, o melhor valor possível de densidade a cada iteração, na tentativa de inserir o mesmo em cada uma das comunidades existentes e verificando o quão boa é a solução em cada etapa.

Algoritmo 3 ConstruirSolução

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$
// Melhor solução encontrada até o momento
$S^* = \{\}$
// Tenta encontrar a melhor solução para cada vértice (dentro da lista de vértices V do grafo aleatorizada)
for all $v \in v1, v2 \ldots$ **do**
  // Cria-se uma solução S' com apenas uma comunidade contendo v e calcula sua densidade
  $S' = \{[v]\}$
  Para cada comunidade c da solução S^*, adicionar o vértice v em busca da sua comunidade ideal
  for all $c \in c1, c2 \ldots$ **do**
    // Adiciona o vértice na comunidade c da solução S^* e calcula sua densidade
    $c = c + [v]$
    $S' = \{c\}$
    // Guarda a solução gerada, caso seja a melhor encontrada
    if $D_\lambda(S')$ melhor que $D_\lambda(S^*)$ **then**
      $S^* = S'$
    end if
  end for
end for
return S^*

No algoritmo da busca local, na etapa de intensificação, busca-se encontrar o melhor vizinho para a solução S, guardando a melhor solução em S'. Caso encontre uma solução vizinha melhor, a melhor solução S' é atualizada. Caso não encontre um melhor, o laço de repetição termina retornando para S' a última melhor solução encontrada.

Algoritmo 4 BuscaLocal

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$ e uma solução S, sendo d sua densidade
// Nova solução S' recebe S
$S' = S$
// Enquanto encontrar um melhor vizinho, atualizar a solução, sendo dS' a densidade do vizinho
while dS' melhor que d **do**
    // Calcula a densidade do melhor vizinho
    $S' =$ MelhorVizinho(S')
end while
return S'

Já para encontrar o melhor vizinho, definido no Algoritmo 5, um laço itera sobre todos os vértices de todas as comunidades de uma solução. Para cada vértice, o mesmo é inserido em cada uma das outras comunidades da solução, e verifica-se o valor da densidade utilizando a função da Modularidade com o vértice na comunidade diferente da sua original. Caso encontre uma comunidade melhor, ele é trocado de comunidade, do contrário, permanece na mesma. Nesse caso, o vértice troca de comunidade na primeira melhor, sem buscar o melhor valor possível.

Algoritmo 5 MelhorVizinho

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$ e uma solução S, sendo d sua densidade
// Nova solução S' recebe S
$S' = S$
// Buscar a primeira melhor solução S'' em cada vértice vc de cada comunidade c
for all $c \in c1, c2 \ldots, communities$ **do**
    for all $cv \in cv1, cv2 \ldots, vertices$ **do**
        for all $nc \in nc1, nc2 \ldots, communities$ **do**
            // Adiciona v na próxima comunidade nc
            $nc.add(v)$
            // Calcula a densidade com v em nc e guarda a solução gerada, caso seja a melhor encontrada
            if $D_\lambda(S')$ melhor que $D_\lambda(S'')$ **then**
                return (S'')
            else
                $nc.remove(v)$
            end if
        end for
    end for
end for
return d

Para a realização dos experimentos, foram utilizadas instâncias reais de grafos utilizados no trabalho De Santiago e Lamb (2020), sendo elas instâncias reais ***Slovene***

Figura 5 – Valores ótimos de Gahuku-Gama Subtribes e Slovene Parliamentary Party.

λ	Slovene Parliamentary Party Time(s)	Slovene Parliamentary Party D*	Gahuku-Gama Subtribes Time(s)	Gahuku-Gama Subtribes D*
0.1	0.051	2.000	0.032	3.037
0.2	0.056	5.800	0.061	7.445
0.3	0.035	11.200	0.159	11.854
0.4	0.016	16.600	0.286	17.076
0.5	0.019	23.000	0.400	24.238
0.6	0.057	36.000	0.427	36.520
0.7	0.050	53.999	0.305	55.749
0.8	0.052	72.000	0.185	75.500
0.9	0.056	89.999	0.209	95.466

Fonte: De Santiago e Lamb (2020)

Parliamentary Party e ***Gahuku-Gama Subtribes*** (Tabela 1), onde serão referenciadas nessa seção como $Parlamento$ e $Gahuku$, respectivamente. A rede $Gahuku$ representa a relação das tribos da Nova Guiné (READ, 1964), e a rede $Parlamento$ representa a relação entre dez partidos políticos do Parlamento Esloveno em 1994 (KROPIVNIK; MRVAR, 1996). Três conjuntos de configurações de experimento foram definidas para aplicar a heurística, onde cada configuração é dada por um valor de λ e um valor de m (que define a quantidade de soluções que serão criadas no Algoritmo 2). Cada configuração do experimento foi repetida uma quantidade de 30 vezes, uma vez que executar o experimento múltiplas vezes garante medidas de tempo de execução mais precisas, e também permite mais abrangência de resultados para realizar a análise considerando o componente randômico no Algoritmo 3). Os experimentos foram executados com o valor de m sendo definido como 20, 30 e 40, escolhidos arbitrariamente, e o valor de λ variando para $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$.

Tabela 1 – Quantidades de vértices (V) e arestas (E) de cada rede.

	vertices	**arestas**
gahuku.net	16	120
parlamento.net	10	45

Realizou-se também comparações dos valores obtidos pelos experimentos com os valores ótimos (chamados nesse trabalho também de valores ideais) encontrados em De Santiago e Lamb (2020). Para isso, a Figura 5 definida no estudo foi utilizada como referência de valores ideais de densidade (D) para cada valor de $lambda$ (λ).

O desenvolvimento da heurística foi feito na linguagem Python. Os experimentos foram realizados de forma paralela utilizando a biblioteca ***multiprocessing*** do Python. O componente probabilístico do Algoritmo 3 foi obtido utilizando a função ***shuffle*** fornecido pela biblioteca ***random*** da linguagem Python para transformar uma lista de vértices em uma lista aleatória de vértices.

Abaixo a tabela com as médias de densidade obtidas por configuração, para cada rede estudada. Destacado em vermelho estão as médias de densidade mais distantes do valor ótimo. Em laranja, as médias são valores um pouco mais próximos, no máximo 2.6 de diferença da valor ideal. Em azul as médias obtidas foram ainda mais próximas, com no máximo 0.2 de diferença, e em verde os valores ideais foram alcançados.

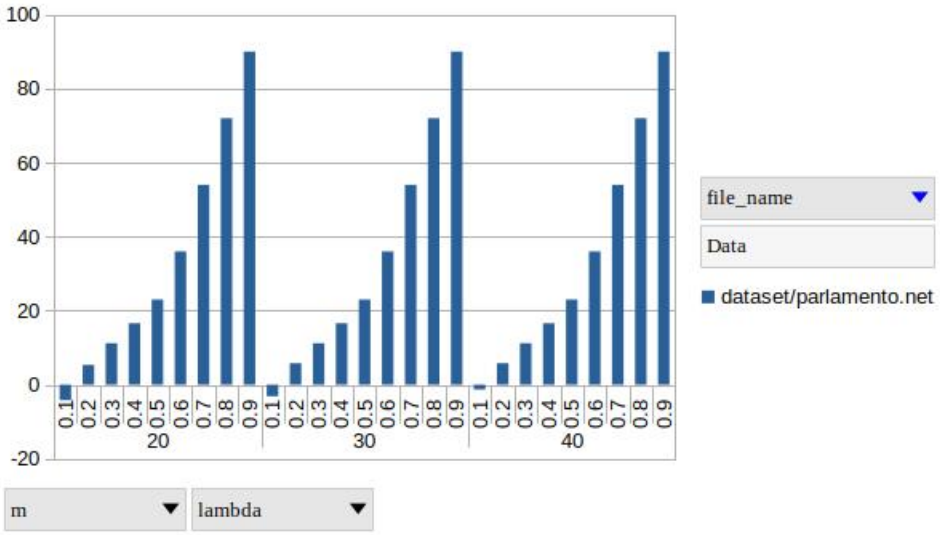
Tabela 2 – Valores médios de densidade obtidos aplicando a heurística para cada instância e parâmetro.

lambda	m	gahuku	parlamento
0.1	**20**	-5.800	-4.178
	30	-5.800	-3.123
	40	-5.800	-1.302
0.2	**20**	0.511	5.335
	30	1.047	5.793
	40	2.774	5.787
0.3	**20**	9.120	**11.200**
	30	8.668	**11.200**
	40	9.815	**11.200**
0.4	**20**	16.480	**16.600**
	30	16.690	**16.600**
	40	16.588	**16.600**
0.5	**20**	**24.238**	**23.000**
	30	**24.238**	**23.000**
	40	**24.238**	**23.000**
0.6	**20**	**36.520**	**36.000**
	30	**36.520**	**36.000**
	40	**36.520**	**36.000**
0.7	**20**	**55.750**	**54.000**
	30	**55.750**	**54.000**
	40	**55.750**	**54.000**
0.8	**20**	**75.500**	**72.000**
	30	**75.500**	**72.000**
	40	**75.500**	**72.000**
0.9	**20**	**95.467**	**90.000**
	30	**95.467**	**90.000**
	40	**95.467**	**90.000**

As Figuras 6 e 7 mostram as médias de densidade de cada rede, e a Figura 8 faz o comparativo entre as duas.

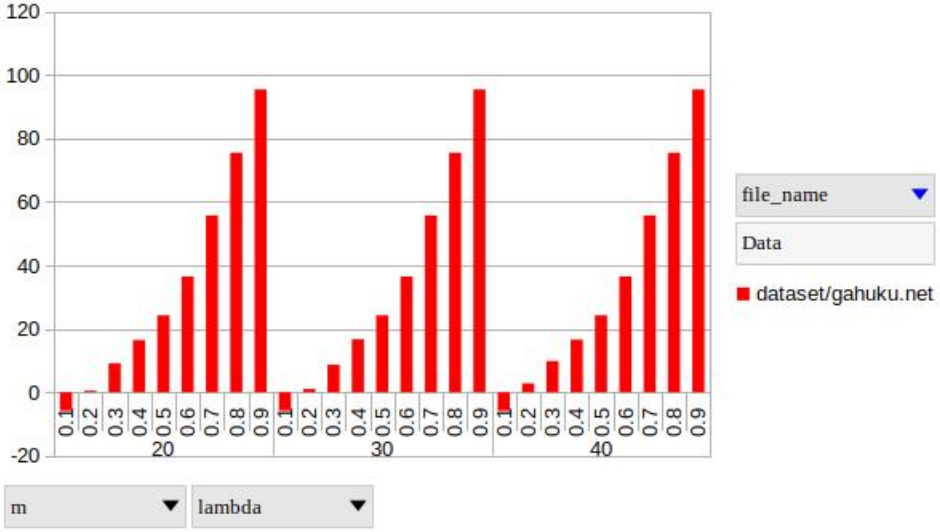
Com base nesses experimentos é possível observar que a média dos valores da densidade (Tabela 2) para valores menores de λ (entre 0.1 e 0.4) ficam mais longe do valor ideal (solução ótima, vide Figura 5) para a rede $Gahuku$, que possui mais ligações que a rede $Parlamento$. Já para a rede $Parlamento$, a média dos valores da densidade para valores da λ até 0.2 ficam mais longe do valor ideal. A partir do λ definido como 0.5 as médias de densidade se estabilizam mais próximas do valor ideal para as duas redes.

Figura 6 – Gráfico de médias de densidade da rede Slovene Parliamentary Party.



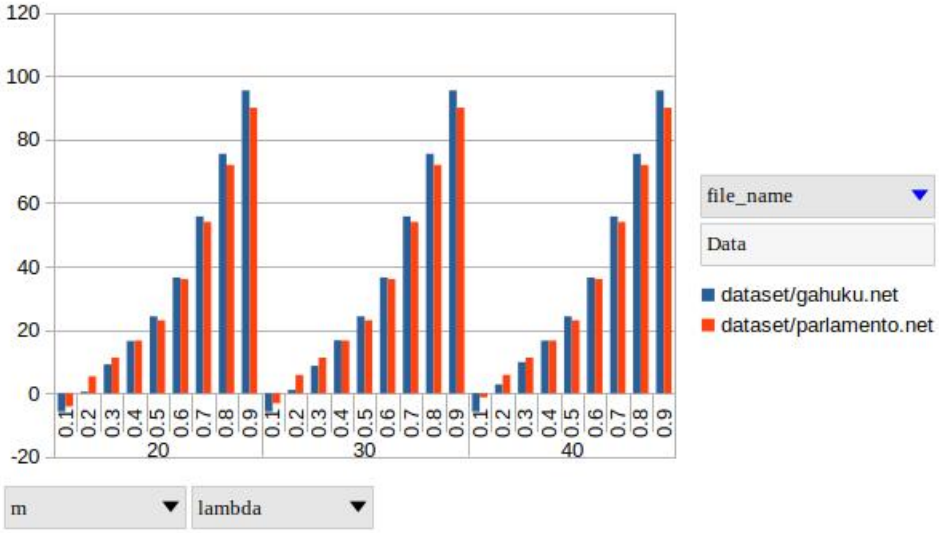
Fonte: Autoria própria.

Figura 7 – Gráfico de médias de densidade da rede Gahuku-Gama Subtribes.



Fonte: Autoria própria.

Figura 8 – Gráfico de médias de densidade relacionado todas as redes do experimento.



Fonte: Autoria própria.

Para a maioria dos cenários, houve ao menos alguma execução dentre as 30 realizadas onde o valor ideal é obtido com exatidão, como pode ser observado na Tabela 3 que demonstra o valor máximo obtido para cada configuração de teste. As exceções que atingiram o valor ideal estão na cor verde.

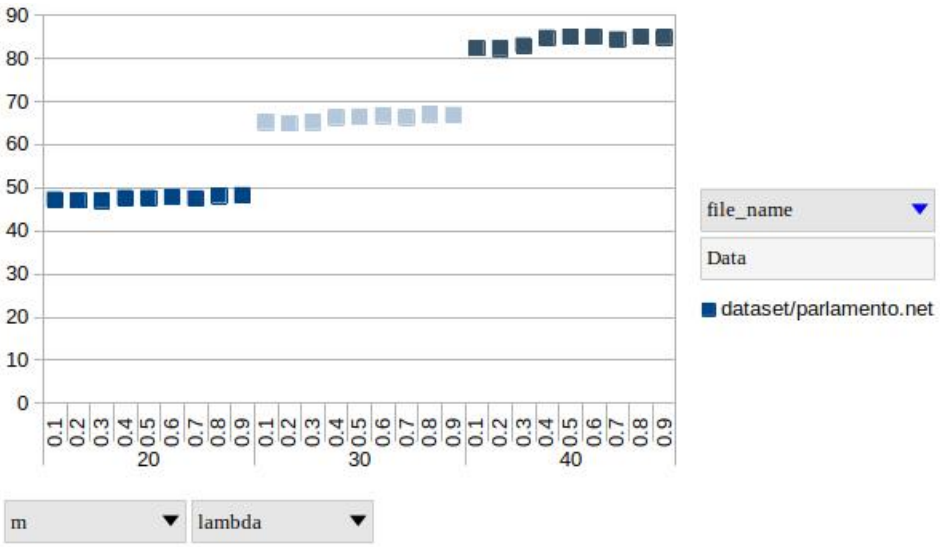
Tabela 3 – Valores máximos de densidade obtidos aplicando a heurística para cada instância e parâmetro.

lambda	m	gahuku	parlamento
0.1	**20**	-5.800	**2.000**
	30	-5.800	**2.000**
	40	-5.800	**2.000**
0.2	**20**	3.800	**5.800**
	30	3.800	**5.800**
	40	3.800	**5.800**
0.3	**20**	**11.854**	**11.200**
	30	**11.854**	**11.200**
	40	**11.854**	**11.200**
0.4	**20**	**17.076**	**16.600**
	30	**17.076**	**16.600**
	40	**17.076**	**16.600**
0.5	**20**	**24.238**	**23.000**
	30	**24.238**	**23.000**
	40	**24.238**	**23.000**
0.6	**20**	**36.520**	**36.000**
	30	**36.520**	**36.000**
	40	**36.520**	**36.000**
0.7	**20**	**55.750**	**54.000**
	30	**55.750**	**54.000**
	40	**55.750**	**54.000**
0.8	**20**	**75.500**	**72.000**
	30	**75.500**	**72.000**
	40	**75.500**	**72.000**
0.9	**20**	**95.467**	**90.000**
	30	**95.467**	**90.000**
	40	**95.467**	**90.000**

Como esperado, quanto maior é o m na execução, maior também a média de tempo (observe na Figura 11). Além disso, mesmo as redes $Gahuku$ e $Parlamento$ tendo quantidades próximas de vértices, a quantidade de arestas implica diretamente no tempo de execução da heurístíca, sendo que quanto maior for a quantidade de ligações da rede, maior o tempo de execução. Os tempos de execução da Figura 11 se tratam de uma média de tempo de execução dentro de 30 execuções, portanto, uma nova configuração de testes foi executada para fazer uma comparação com os valores referência da Imagem 5 com apenas uma repetição, que será detalhada posteriormente.

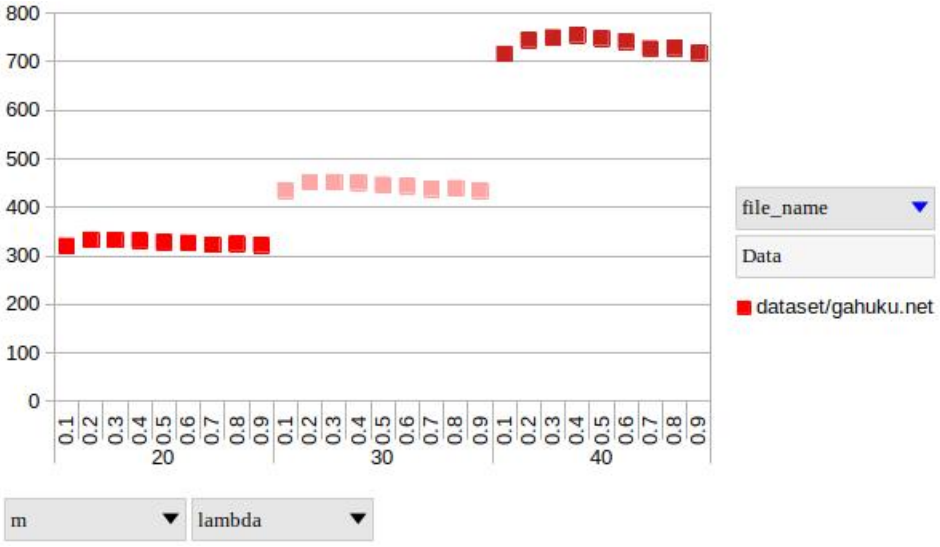
Também é possível observar que, para as redes estudadas, valores maiores de m não tem influência significativa nos resultado das médias de densidade na aplicação da heurística, e não garantem que o melhor resultado de densidade será obtido aumentando o m. Ou seja, para a proposta deste trabalho, maiores quantidades de soluções geradas buscando encontrar a melhor comunidade individualmente para cada vértice

Figura 9 – Gráfico de médias de tempo (s) da rede Slovene Parliamentary Party.



Fonte: Autoria própria.

Figura 10 – Gráfico de médias de tempo (s) da rede Gahuku-Gama Subtribes.

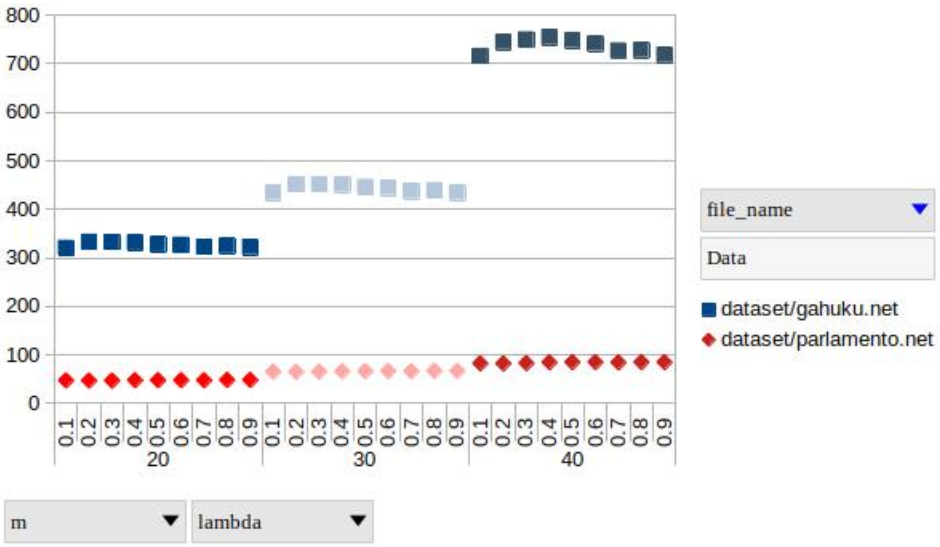


Fonte: Autoria própria.

não implica em uma eficácia maior, mas sim em um tempo de execução maior.

Além dos experimentos acima, para fazer uma comparação de tempo com os valores referência da Imagem 5, foram realizadas para cada rede no experimento execuções de apenas uma repetição, com o valor de m sendo definido como 20, 30 e 40, escolhidos arbitrariamente, e o valor de λ variando para $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$.

Figura 11 – Gráfico de médias de tempo (s) relacionado todas as redes do experimento.



Fonte: Autoria própria.

Tabela 4 – Valores de tempos de execução em segundos obtidos aplicando a heurística para cada instância e parâmetro para apenas uma única repetição.

m	lambda	gahuku	parlamento
20	**0.1**	16.841	2.396
	0.2	16.527	1.915
	0.3	11.738	1.195
	0.4	9.068	1.990
	0.5	8.894	1.897
	0.6	4.707	1.705
	0.7	6.956	1.722
	0.8	9.308	2.268
	0.9	11.790	2.693
30	**0.1**	25.981	3.719
	0.2	26.540	1.896
	0.3	20.255	2.612
	0.4	15.732	3.120
	0.5	8.945	3.024
	0.6	11.317	2.723
	0.7	11.769	2.593
	0.8	16.083	3.938
	0.9	20.340	4.235
40	**0.1**	36.940	4.777
	0.2	35.775	3.002
	0.3	21.329	2.789
	0.4	19.009	3.878
	0.5	15.712	3.601
	0.6	16.538	3.400
	0.7	16.429	3.382
	0.8	27.410	4.613
	0.9	27.026	5.065

Analisando os tempos de execução nesse segundo experimento, a heurística desenvolvida teve uma demora consideravelmente maior em relação aos valores referência da Imagem 5. Isso se dá provavelmente devido aos valores de m, que implicam em mais repetições dentro da heurística. Observe que, novamente, quanto maior for o m, maior o tempo de execução. Além disso, o trabalho relacionado utilizado como referência de valores ótimos, teve seu desenvolvimento feito em outra linguagem, sendo ela o C. Um fator importante para um bom tempo de exeução dos experimentos utilizando essas redes é a máquina onde os experimentos estão sendo executados, pois como a análise é feita com execução paralela, quanto mais núcleos o computador fornecer, melhor a paralelização.

5 CONCLUSÕES

No presente trabalho foi realizado um levantamento teórico dos principais tópicos e estudos referentes ao problema da Modularidade por Densidade com Sinais, além de ressaltar a importância e aplicação em diversas áreas desses problemas na atualidade. Foram apresentados três trabalhos relacionados para trazer algumas comparações e inspirações para desenvolver uma heurística para resolver o problema da Modularidade por Densidade com Sinais, sendo eles (CHEN *et al.*, 2017), (SANTIAGO; LAMB, 2017) e (ZHOU; WANG, 2016). Uma proposta foi modelada e desenvolvida, com baseada nos trabalhos de (CHEN *et al.*, 2017) e (SANTIAGO; LAMB, 2017) utilizando conceitos de busca gulosa, aleatória e adaptativa (GRASP).

Seguindo do desenvolvimento da heurística, foi levantada uma análise e comparação dos resultados com heurísticas e soluções já existentes para compreender em quais aspectos a heurística pode auxiliar no problema de otimização. A aplicação da heurística para duas instâncias reais ***Slovene Parliamentary Party*** e ***Gahuku-Gama Subtribes*** alcançou os valores de densidade considerados ideais referenciados pelo trabalho De Santiago e Lamb (2020) em quase todos os cenários para ao menos alguma entre as 30 execuções realizadas no experimento. No entanto, a estratégia de gerar um valor m de soluções não resulta necessariamente em uma boa média de densidade conforme m aumenta. A partir de um λ de 0.5, as médias de densidade se estabilizaram para as duas redes estudadas, onde o valor de m não teve impacto relevante. Os tempos de execução estão diretamente ligados com o valor de m, com a quantidade de execuções que a heurística é aplicada, e também com o tamanho da rede que está sendo analisada.

Por fim, este projeto pode ser futuramente expandido, utilizando-se em seus experimentos em redes maiores, sejam elas reais ou artificiais. A heurística pode ser desenvolvida em alguma outra linguagem para observar se afetaria de forma significativa o tempo de execução, por exemplo. Outra abordagem que pode ser seguida em um projeto futuro seria experimentar outros critérios gulosos no seu desenvolvimento.

REFERÊNCIAS

ALOISE, Daniel *et al.* Column generation algorithms for exact modularity maximization in networks. **Physical Review E**, v. 82, n. 4, p. 046112, out. 2010. DOI: 10.1103/PhysRevE.82.046112. Citado na p. 25.

BLONDEL, Vincent D *et al.* Fast unfolding of communities in large networks. **Journal of Statistical Mechanics: Theory and Experiment**, v. 2008, n. 10, p10008, out. 2008. ISSN 1742-5468. DOI: 10.1088/1742-5468/2008/10/P10008. Citado na p. 10.

BRANDES, U. On Modularity Clustering. **IEEE Transactions on Knowledge and Data Engineering**, v. 20, n. 2, p. 172–188, fev. 2008. DOI: 10.1109/TKDE.2007.190689. Citado nas pp. 10, 15, 18.

CHEN, Y. *et al.* Uncovering the community structure in signed social networks based on greedy optimization. **Modern Physics Letters B**, v. 31, n. 14, 2017. DOI: 10.1142/S0217984917501585. Citado nas pp. 11, 15, 20, 21, 38.

CLAUSET, Aaron; NEWMAN, M. E. J.; MOORE, Cristopher. Finding community structure in very large networks. **Physical Review E**, v. 70, n. 6, p. 066111, dez. 2004. DOI: 10.1103/PhysRevE.70.066111. Citado na p. 10.

DE SANTIAGO, R.; LAMB, L.C. Exact Signed Modularity Density Maximization Solutions and Their Real Meaning*. *In:* DOI: 10.1109/CEC48606.2020.9185736. Citado nas pp. 11, 25, 29, 30, 38.

DIESTEL, Reinhard. **Graph Theory**. [*S.l.*]: Springer, 2017. Citado nas pp. 13, 14.

FEO, T.A.; RESENDE, M.G.C. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, v. 8, n. 2, p. 67–71, 1989. DOI: 10.1016/0167-6377(89)90002-3. Citado na p. 27.

FERRARA, Emilio *et al.* Detecting criminal organizations in mobile phone networks. **Expert Systems with Applications**, Elsevier Ltd, v. 41, n. 13, p. 5733–5750, out. 2014. DOI: 10.1016/j.eswa.2014.03.024. Citado nas pp. 14, 15.

FORTUNATO, Santo; BARTHÉLEMY, Marc. Resolution limit in community detection. **Proceedings of the National Academy of Sciences of the United States of America**, v. 104, n. 1, p. 36–41, jan. 2007. DOI: 10.1073/pnas.0605965104. Citado nas pp. 10, 16, 17.

FORTUNATO, Santo; CASTELLANO, Claudio. Community Structure in Graphs. *In:* **Computational Complexity: Theory, Techniques, and Applications**. Edição: Robert A. Meyers. New York, NY: Springer New York, 2012. P. 490–512. DOI: 10.1007/978-1-4614-1800-9_33. Citado nas pp. 10, 13.

GENDREAU, Michel; POTVIN, Jean-Yves. **Handbook of Metaheuristics**. 2nd. [*S.l.*: *s.n.*], 2010. Citado na p. 19.

GONG, M. *et al.* Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. **IEEE Transactions on Evolutionary Computation**, v. 18, n. 1, p. 82–97, 2014. DOI: 10.1109/TEVC.2013.2260862. Citado na p. 24.

GUIMERÀ, Roger *et al.* The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 102, n. 22, p. 7794–7799, 2005. Citado na p. 10.

KARRER, Brian; NEWMAN, Mark EJ. Stochastic blockmodels and community structure in networks. **Physical Review E**, APS, v. 83, n. 1, p. 016107, 2011. Citado na p. 17.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. *In:* PROCEEDINGS of ICNN'95 - International Conference on Neural Networks. [*S.l.*: *s.n.*], 1995. 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968. Citado na p. 20.

KROPIVNIK, Simona; MRVAR, Andrej. An Analysis of the Slovene Parliamentary Parties Network. **Developments in Statistics and Methodology, Metodološki zvezki**, v. 12, p. 209–216, 1996. Citado na p. 30.

LANCICHINETTI, Andrea; FORTUNATO, Santo; RADICCHI, Filippo. A unified view of performance metrics: transitions and generalizations. **Journal of Statistical Mechanics: Theory and Experiment**, IOP Publishing, v. 2011, n. 09, p09008, 2011. Citado na p. 17.

LESKOVEC, Jure; HUTTENLOCHER, Daniel; KLEINBERG, Jon. Signed networks in social media. *In:* PROCEEDINGS of the 28th international conference on Human factors in computing systems - CHI '10. New York, New York, USA: ACM Press, 2010. P. 1361. DOI: 10.1145/1753326.1753532. Citado na p. 10.

LI, Yadong; LIU, Jing; LIU, Chenlong. A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks. **Soft Computing**, v. 18, n. 2, p. 329–348, fev. 2014. DOI: 10.1007/s00500-013-1060-4. Citado nas pp. 11, 17.

LI, Zhenping; ZHANG, Shihua *et al.* Quantitative function for community detection. **Physical Review E**, v. 77, n. 3, p. 036109, mar. 2008. DOI: 10.1103/PhysRevE.77.036109. Citado nas pp. 10, 14, 17, 18.

MANIEZZO, V.; STÜTZLE, T.; VOSS, S. **Matheuristics: Hybridizing Metaheuristics and Mathematical Programming**. [*S.l.*]: Springer US, 2009. (Annals of Information Systems). Citado na p. 18.

MEUNIER, David; FONLUPT, Pierre *et al.* Modular structure of functional networks in olfactory memory. **NeuroImage**, Elsevier Inc., p. 264–75, jul. 2014. DOI: 10.1016/j.neuroimage.2014.03.041. Citado nas pp. 10, 15.

MEUNIER, David; LAMBIOTTE, Renaud; BULLMORE, Edward T. Modular and hierarchically modular organization of brain networks. **Frontiers in neuroscience**, Frontiers, v. 4, p. 200, 2010. Citado na p. 10.

NEPUSZ, Tamás; YU, Haiyuan; PACCANARO, Alberto. Detecting overlapping protein complexes in protein-protein interaction networks. **Nature Methods**, v. 9, n. 5, p. 471–472, mai. 2012. DOI: 10.1038/nmeth.1938. Citado na p. 10.

NEWMAN, M.; GIRVAN, M. Finding and evaluating community structure in networks. **Physical Review E**, v. 69, n. 2, p. 026113, fev. 2004. DOI: 10.1103/PhysRevE.69.026113. Citado nas pp. 10, 15.

READ, Kenneth E. Cultures of the Central Highlands, New Guinea. **Southwestern Journal of Anthropology**, v. 10, 1964. Citado na p. 30.

ROTHLAUF, Franz. **Design of Modern Heuristics: Principles and Application**. 1st. [*S.l.*]: Springer Publishing Company, Incorporated, 2011. Citado na p. 18.

ROTTA, Randolf; NOACK, Andreas. Multilevel local search algorithms for modularity clustering. **Journal of Experimental Algorithmics**, v. 16, n. 2, p. 2.1, mai. 2011. DOI: 10.1145/1963190.1970376. Citado na p. 10.

SANTIAGO, R. de; LAMB, Luís C. A ground truth contest between modularity maximization and modularity density maximization. **Artificial Intelligence Review**, jan. 2020. DOI: 10.1007/s10462-019-09802-8. Citado nas pp. 15, 20.

SANTIAGO, Rafael de; LAMB, Luís C. Exact computational solution of Modularity Density Maximization by effective column generation. **Computers & Operations Research**, v. 86, p. 18–29, out. 2017. DOI: 10.1016/j.cor.2017.04.013. Citado na p. 25.

SANTIAGO, Rafael; LAMB, Luís C. Efficient Quantitative Heuristics for Graph Clustering. *In:* PROCEEDINGS of the Genetic and Evolutionary Computation Conference Companion. Berlin: ACM New York, 2017. P. 117–118. DOI: 10.1145/3067695.3075995. Citado nas pp. 15, 25.

SANTIAGO, Rafael; LAMB, Luís C. Efficient modularity density heuristics for large graphs. **European Journal of Operational Research**, v. 258, n. 3, p. 844–865, 2017. DOI: https://doi.org/10.1016/j.ejor.2016.10.033. Citado na p. 38.

SCHMEJA, S. Identifying star clusters in a field: a comparison of different algorithms. **Astronomische Nachrichten**, v. 332, n. 2, p. 172–184, fev. 2011. DOI: 10.1002/asna.201011484. Citado na p. 10.

TRAAG, Vincent A; VAN DOOREN, Paul. Surprise maximization reveals the community structure of complex networks. **Scientific Reports**, Nature Publishing Group, v. 1, n. 1, p. 50, 2011. Citado na p. 17.

WOLPERT, D.H.; MACREADY, W.G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, abr. 1997. DOI: 10.1109/4235.585893. Citado na p. 19.

ZHOU, D.; WANG, X. A neighborhood-impact based community detection algorithm via discrete PSO. **Mathematical Problems in Engineering**, v. 2016, 2016. DOI: 10.1155/2016/3790590. Citado nas pp. 15, 20, 23, 38.

Apêndices

APÊNDICE A – ARTIGO

Desenvolvimento de uma heurística para o problema da maximização da modularidade por densidade com sinais

Letícia do Nascimento[1], **Rafael de Santiago**[2]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Santa Catarina – SC – Brasil

leticia.nascimento@ufsc.br, r.santiago@ufsc.br

Abstract. *Clustering problems in graphs are prevalent in various fields, where a popular approach to defining these clusters is identifying sets of highly related vertices. Among the strategies developed to tackle these problems, those based on maximizing modularity are prominent. Formulations for computational problems of modularity maximization aim to capture the positive and negative relationships between vertices, reflecting the proximity and repulsion among the entities represented in the graphs. In this context, this work aims to propose and analyze a computational heuristic for one of the modularity-based clustering problems, known in the literature as the "modularity maximization by density problem." To address this objective, a survey of computational methods present in the literature was conducted. A heuristic method based on GRASP (Greedy Randomized Adaptive Search Procedure) algorithm was specified, proposed, and developed in the Python language. Subsequently, it was analyzed and compared with other computational methods found in the literature. The results of the experiments were able to achieve the ideal values referenced by [De Santiago and Lamb 2020]) in most of their executions, with stability in density values at* λ *of 0.5. However, the strategy of generating a quantity* m *of solutions in the applied GRASP did not prove to be effective in guaranteeing good density values as* m *increases. The complete system is available on the GitHub platform.*

Resumo. *Problemas de agrupamento em grafos são comuns em diversas áreas, e uma abordagem frequente para definir esses agrupamentos é identificar conjuntos de vértices altamente relacionados. Dentre as estratégias desenvolvidas para lidar com esses problemas, destacam-se aquelas baseadas na maximização da modularidade. Existem formulações para problemas computacionais de maximização da modularidade que buscam capturar os relacionamentos positivos e negativos entre os vértices, refletindo a proximidade e a repulsa entre as entidades representadas nos grafos. Nesse contexto, esse trabalho propõe e analisa uma heurística computacional para um dos problemas de agrupamento por modularidade por sinal, conhecido na literatura como "problema da maximização da modularidade por densidade". Para isto, foi realizado um levantamento dos métodos computacionais presentes na literatura para tratar o problema, especificado, proposto e desenvolvido na linguagem Python um método heurístico baseado em GRASP (Greedy Randomized Adaptive Search Procedure), e posteriormente analisado e comparado*

com outros métodos computacionais presentes na literatura. Os resultados dos experimentos conseguiram atingir os valores ideais referenciados por [De Santiago and Lamb 2020] na maior parte das suas execuções, tendo estabilidade nos valores de densidade com λ de 0.5. No entanto, a estratégia de gerar uma quantidade m de soluções no GRASP aplicado não se demonstrou efetivo para garantir bons valores de densidade conforme m aumenta. Todos os códigos-fonte do trabalho foram disponibilizados na plataforma GitHub.

1. Introdução

Encontrar grupos bem relacionados de entidades na vida real é um problema útil em diversos domínios, e o interesse nesses estudos vem crescendo nos últimos anos. O problema pode ser aplicado na na medicina, para identificar a memória funcional envolvida no reconhecimento olfativo [Meunier et al. 2014], na astronomia, para identificação de grupos de estrelas [Schmeja 2011], na biologia, para encontrar complexos de proteínas [Nepusz et al. 2012], entre muitas outras aplicações.

Esses grupos são denominados na literatura de comunidades e são geralmente formalizados em grafos, nos quais cada vértice representa uma entidade e as relações entre elas são definidas por arcos ou arestas [Newman and Girvan 2004, Leskovec et al. 2010, Fortunato and Castellano 2012]. A relação entre entidades pode incorporar um peso positivo ou negativo. Um exemplo de conexão com sinal seria no estudo de mídias sociais no qual, onde as ligações podem representar uma relação positiva (de amizade) ou negativa (de antagonismo) entre os usuários. A atitude de um usuário em relação a outro pode ser estimada a partir de evidências fornecidas por seus relacionamentos com outros membros da rede social [Leskovec et al. 2010].

Com a motivação principal sendo evitar o problema da Maximização da Modularidade chamado de resolução limite, uma reformulação para o problema da Maximização da Modularidade foi desenvolvida [Li et al. 2008], chamada de Maximização da Modularidade por Densidade. A reformulação utiliza a diferença entre a densidade interna e externa de ligações para cada agrupamento, avaliando assim as soluções do problema. Sua variação para lidar com grafos com sinais foi proposta em [Li et al. 2014]. Alguns métodos exatos e heurísticos foram propostos na literatura [Li et al. 2014, De Santiago and Lamb 2020].

Este trabalho propõe uma heurística para o problema da Maximização da Modularidade por Densidade com Sinais a partir do estado atual da arte, levantando os métodos computacionais presentes na literatura e analisando os resultados, comparando a heurística desenvolvida com os métodos existentes. A estratégia proposta compreende na aplicação de uma busca local monótona à solução inspirada por [Chen et al. 2017], que utiliza conceitos de busca gulosa, aleatória e adaptativa (GRASP, do inglês ***greedy randomized adaptive search procedure***. Algumas configurações de teste foram definidas, cujos resultados foram analisados e comparados com o estudo [De Santiago and Lamb 2020].

2. Objetivos

Propor uma heurística para o problema de maximização da modularidade por densidade com sinais.

2.1. Objetivos específicos

Para atingir o objetivo geral, cumpriram-se os seguintes objetivos específicos:

- Especificar nova heurística para o problema de maximização da modularidade por densidade com sinais;
- Desenvolver nova heurística para o problema de maximização da modularidade por densidade com sinais;
- Analisar os resultados.

3. Método de Pesquisa

Primeiramente, foi efetuado um levantamento teórico com intuito de fornecer maior credibilidade à pesquisa. Esta etapa foi realizada com a busca de artigos e livros no portal da SCOPUS. A partir dos estudos encontrados na busca, foram escolhidos três trabalhos cujo critério de seleção foi a relevância de acordo com o SCOPUS e passando por uma filtragem da autora, buscando garantir que os trabalhos sejam relacionados com a pesquisa aqui desenvolvida.

Concluída a primeira etapa, o próximo passo foi a especificação da heurística. Com a especificação concluída, foi realizado o desenvolvimento da heurística na linguagem Python. A escolha da ferramenta foi com base na afinidade pela mesma.

A última etapa consiste na análise dos resultados, na qual obtiveram-se informações sobre o desempenho do método proposto, comparando os resultados obtidos com outros da literatura. Mais especificamente, os resultados foram comparados com um dos trabalhos relacionados que, a partir do desenvolvimento de um algoritmo exato, obteve resultados ótimos em seus experimentos.

4. Fundamentação Teórica

O capítulo de fundamentação teórica consiste na contextualização dos fundamentos da Maximização da Modularidade por Densidade com Sinais, bem como um breve histórico de evolução da detecção de comunidades, aplicações e importância.

4.1. Maximização da Modularidade por Densidade com Sinais

A Maximização da Modularidade por Densidade (MMD) [Li et al. 2008] surgiu com a motivação de evitar o problema da resolução limite da MM [Fortunato and Barthélemy 2007], utilizando o conceito de grau médio de modularidade, chamada de medida quantitativa da densidade de modularidade. A MMD é uma medida alternativa de modularidade que considera a densidade das comunidades em relação à densidade global da rede. Em contrapartida à fórmula da modularidade original, que utiliza a diferença entre o número real e o número esperado de arestas, a modularidade por densidade considera a razão entre essas duas quantidades, permitindo que a medida seja menos influenciada pela densidade global da rede e mais sensível a comunidades menos densas em redes esparsas. Uma variação da modularidade por densidade para lidar com grafos com sinais foi proposta em [Li et al. 2014], chamada de Maximização da Modularidade por Densidade com Sinais (MMDS). Sua função é dada na Equação 1.

Dado um grafo com sinal G e uma comunidade C, considere $G = (V, E^+, E^-)$, onde V é o conjunto de vértices, E^+ é o conjunto de arestas positivas, e E^- é o conjunto

de arestas negativas. Considere que $w_{uv}^{+} = 1$ se $\{u, v\} \in E^{+}$, caso contrário $w_{uv}^{+} = 0$, e $w_{uv}^{-} = 1$ se $\{u, v\} \in E^{-}$, caso contrário $w_{uv}^{-} = 0$. Os conjuntos E_c^{+} e E_c^{-} são compostos pelos vértices positivos e negativos, respectivamente, de uma comunidade $c \in C$. Portanto, $|E_c^{+}|$ representa o número de arestas positivas (ou arestas dentro da comunidade) em uma determinada comunidade c e $|E_c^{-}|$ é o número de arestas negativas (ou arestas fora da comunidade) em uma determinada comunidade c. Dividimos o grau de cada vértice entre positivos e negativos. Logo, o grau positivo de v é dado por $d_v^{+} = \sum_{u \in V} w_{uv}^{+}$ (soma dos graus positivos), e o grau negativo é dado por $d_v^{-} = \sum_{u \in V} w_{uv}^{-}$ (soma dos graus negativos). A somatória $\sum_{c \in C}$ indica que a fórmula deve ser avaliada para cada comunidade no conjunto. O número de vértices na comunidade c é dado por $|c|$.

$$D_\lambda(C) = \sum_{c \in C} \left(\frac{2\lambda|E_c^{+}| - 2(1-\lambda)(\sum_{v \in c} d_v^{+} - |E_c^{+}|)}{|c|} \right. \\ \left. \frac{-2(1-\lambda)|E_c^{-}| + 2\lambda(\sum_{v \in c} d_v^{-} - |E_c^{-}|)}{|c|} \right). \tag{1}$$

Nessa função, $D_\lambda(C)$ é a medida de modularidade de identidade para o conjunto de comunidades C com um parâmetro de ajuste λ. O parâmetro λ é usado para obter a "associação de proporção" para encontrar comunidades pequenas quando $\lambda > 0.5$, e o "recorte de proporção" para encontrar comunidades grandes quando $\lambda < 0,5$. O estudo [Li et al. 2008] sugeriu que esta função pode ser usada para encontrar o nível apropriado da estrutura topológica de grafos para encontrar partições apropriadas. Em outras palavras, λ é um parâmetro de ajuste que controla o equilíbrio entre as contribuições das arestas positivas e negativas para a modularidade.

Em resumo, a fórmula em si calcula a modularidade de identidade para cada comunidade no conjunto C e, em seguida, soma as contribuições de todas as comunidades para obter o valor total da modularidade de identidade $D_\lambda(C)$. A fórmula considera tanto as arestas positivas quanto as arestas negativas, e o parâmetro λ permite ajustar a importância relativa dessas duas contribuições.

4.2. Heurísticas e Metaheurísticas

O problema da Maximização da Modularidade provavelmente não pode resolver instâncias de tamanho razoável em tempo polinomial, por ser um problema NP-Difícil [Brandes 2008]. Assim, o problema da Maximização da Modularidade é frequentemente resolvido por métodos heurísticos.

O uso de heurísticas e metaheurísticas para resolver problemas do mundo real é amplamente aceito na comunidade de pesquisa operacional. Sabemos que a grande maioria dos problemas complexos de decisão do mundo real, quando modelados como problemas de otimização, pertencem à classe de problemas NP-difíceis. Isto implica que abordagens exatas estão fadadas ao fracasso ao lidar com grandes instâncias de escala real, sejam elas provenientes de negócios, engenharia, economia ou ciência. Hoje, os processos de tomada de decisão são cada vez mais complexos e mais abrangentes, no sentido de que mais variáveis de decisão são usadas para modelar situações complexas e

mais dados de entrada e parâmetros estão disponíveis para capturar a complexidade dos próprios problemas [Maniezzo et al. 2009].

Heurísticas são métodos computacionais que não garantem a solução ótima, mas buscam soluções com características presentes nas melhores soluções. As heurísticas podem parar depois de encontrar soluções chamadas de ótimas locais. As ótimas locais são soluções de referência em uma região no espaço de busca. Por se tratarem da melhor solução da região, pode dificultar que soluções melhores sejam encontradas. Diz-se então que a heurística "ficou presa" em um ótimo local [Rothlauf 2011].

Para evitar que ótimos locais impeçam buscas por melhores soluções, utiliza-se de metaheurísticas. Elas utilizam-se de abordagens conhecidas como intensificação e diversificação. A intensificação explora o espaço de busca, de maneira limitada, procurando por soluções melhores, ou seja, "intensificadas". A diversificação acontece para impedir que um espaço muito limitado de busca seja explorado. Desse modo, uma metaheurística pode se utilizar de mais de uma estratégia heurística [Gendreau and Potvin 2010].

Nesse contexto, o seguinte capítulo traz os principais conceitos de alguns trabalhos relacionados que utilizam abordagens heurísticas no desenvolvimento dos seus estudos para tratar o problema da Maximização da Modularidade por Densidade com Sinais.

5. Trabalhos Relacionados

Utilizando a ferramenta do SCOPUS para pesquisar sobre Maximização da Modularidade por Densidade, foi possível encontrar alguns estudos recentes sobre o tema. Nessa seção serão apresentados algoritmos desenvolvidos para identificar comunidades em redes com sinal utilizando algoritmo guloso [Chen et al. 2017], partículas (PSO) [Zhou and Wang 2016] e algoritmo ***branch-and-price*** [de Santiago and Lamb 2020].

5.1. Descobrindo estruturas de comunidades em redes sociais baseadas em otimização gananciosa

O estudo [Chen et al. 2017] tem como objetivo focar na análise de redes sociais com sinal e na identificação de suas estruturas de comunidades utilizando um algoritmo de otimização guloso.

Nas redes sociais, indivíduos ou entidades são conectados por arestas que representam relacionamentos ou interações entre eles. Esses relacionamentos podem ser positivos (indicando amizade, colaboração, etc.) ou negativos (indicando inimizade, discordância, etc.), formando uma rede social com sinal. Compreender a estrutura comunitária dentro dessas redes é essencial para várias aplicações, como identificar indivíduos influentes, prever comportamentos de usuários ou estudar a difusão de informações.

Os autores propõem um método baseado em otimização gananciosa para revelar a estrutura de comunidades em redes sociais, representadas por redes com sinal. A otimização gananciosa é uma estratégia que faz escolhas localmente ótimas de forma iterativa para resolver um problema.

O algoritmo guloso explorado para identificar comunidades [Chen et al. 2017] é muito eficiente para buscar as mudanças em ΔQ_{ij} que estão relacionados com a fusão entre cada par de comunidades. De acordo com resultados experimentais, o algoritmo

provou ser capaz de funcionar bem. Mais especificamente, o algoritmo proposto é muito eficiente para redes de tamanho médio, tanto densas quanto esparsas.

5.2. Solução para a modularidade por densidade com sinais e análise das soluções ótimas

Este estudo aborda a maximização exata da densidade de modularidade com sinais em redes complexas e explora o significado real das soluções encontradas. Seu objetivo é encontrar soluções ótimas que maximizem a densidade de modularidade, levando em consideração os sinais das conexões entre os vértices da rede.

O algoritmo ***branch-and-price*** desenvolvido [De Santiago and Lamb 2020] utiliza um método de geração de colunas que deriva de um modelo linear inteiro que foi inspirado em modelos anteriores similiares para problemas envolvendo redes sem sinais [Aloise et al. 2010, de Santiago and Lamb 2017, De Santiago and Lamb 2020].

A geração de colunas encontra a solução ótima em espaço de solução contínua. Por essa razão, foi utilizado um algoritmo para encontrar a solução ótima em um espaço de soluções inteiras, relacionado às variáveis z_t.

Este estudo apresenta uma abordagem exata para maximizar a densidade de modularidade com sinais em redes complexas. Nele, um algoritmo de busca exata é desenvolvido e são realizadas análises para compreender o significado real das soluções encontradas. Os resultados mostram que a abordagem proposta é eficaz na maximização da densidade de modularidade e na interpretação das soluções em termos das características da rede.

6. Método Proposto

Este trabalho consiste na implementação de uma heurística inspirada no artigo apresentado na Seção 5.1. Neste artigo, utiliza-se um algoritmo guloso para resolver o problema da Maximização da Modularidade por Densidade com Sinal. A estratégia proposta compreende na aplicação de uma busca local monótona sobre uma solução construída por um algoritmo de construção. Um procedimento de busca guloso, aleatório e adaptativo (GRASP) é uma meta-heurística multi-partida que aplica o método de busca local repetidamente a partir de soluções construídas por um algoritmo guloso aleatório [Feo and Resende 1989].

7. Desenvolvimento do GRASP Proposto

Um pseudocódigo para o GRASP proposto pode ser visualizado no Algoritmo 1.

Algoritmo 1 GRASP proposto

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$
// Melhor solução encontrada até o momento
$S^* = \{\}$
// Gera m soluções
for all $i \in 1, 2 \ldots, m$ **do**
  // Cria-se uma solução S que busca a melhor solução para cada vértice. A ordem de vértices a ser analisada é sempre aleatória
  $S = \text{ConstruirSolução}(G)$
  // Cria-se uma solução S' que busca a melhor solução local para a solução criada acima
  $S' = \text{BuscaLocal}(G, S)$
  // Guarda a solução gerada, caso seja a melhor encontrada
  if $D_\lambda(S')$ melhor que $D_\lambda(S^*)$ **then**
    $S^* = S'$
  end if
end for
return S^*

Inicialmente, o algoritmo define uma variável S^* que armazenará a melhor solução encontrada na busca. Depois, o algoritmo gera m soluções (uma para cada repetição no laço que compreende as linhas 6 e 13). Para cada solução, seleciona-se iterativamente um componente da lista de vértices do grafo aleatorizada e os coloca na melhor solução possível com o Algoritmo 2. Quando a solução estiver completa, uma busca local é aplicada a solução recém criada (linha 9, Algoritmo 3) que obterá a ótima local S'. Ao final, verifica-se a solução encontrada durante a i-ésima iteração (variável do laço da linha 5) realizando uma comparação com a solução de referência S^*, que é substituída por S' caso essa última seja melhor que a de referência (linhas 11 e 12). Ao final, a busca retorna S^* (linha 15).

Algoritmo 2 na etapa de exploração, busca a melhor comunidade para cada vértice, tendo para cada solução criada uma lista aleatória de vértices. Inicialmente, o algoritmo define uma variável S^* que armazenará a melhor solução encontrada. A primeira solução guarda apenas o próprio vértice que está sendo analisado em uma comunidade. Em seguida, o algoritmo busca o próximo vértice da lista aleatória e o adiciona na comunidade do vértice anterior. A cada passo, é aplicada a função da modularidade para encontrar e armazenar o valor da densidade de cada solução. Caso a densidade para o vértice em uma comunidade isolada seja melhor, o mesmo é mantido na comunidade isolada. No entanto, se o valor da densidade for melhor para o vértice incluso em uma outra comunidade com outros vértices, ele é mantido na comunidade desse conjunto. Assim é construída uma solução, buscando, para cada vértice, o melhor valor possível de densidade a cada iteração, na tentativa de inserir o mesmo em cada uma das comunidades existentes e verificando o quão boa é a solução em cada etapa.

Algoritmo 2 ConstruirSolução

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$
// Melhor solução encontrada até o momento
$S^* = \{\}$
// Tenta encontrar a melhor solução para cada vértice (dentro da lista de vértices V do grafo aleatorizada)
for all $v \in v1, v2 \ldots$ **do**
  // Cria-se uma solução S' com apenas uma comunidade contendo v e calcula sua densidade
  $S' = \{[v]\}$
  Para cada comunidade c da solução S^*, adicionar o vértice v em busca da sua comunidade ideal
  for all $c \in c1, c2 \ldots$ **do**
    // Adiciona o vértice na comunidade c da solução S^* e calcula sua densidade
    $c = c + [v]$
    $S' = \{c\}$
    // Guarda a solução gerada, caso seja a melhor encontrada
    if $D_\lambda(S')$ melhor que $D_\lambda(S^*)$ **then**
      $S^* = S'$
    end if
  end for
end for
return S^*

No algoritmo da busca local, na etapa de intensificação, busca-se encontrar o melhor vizinho para a solução S, guardando a melhor solução em S'. Caso encontre uma solução vizinha melhor, a melhor solução S' é atualizada. Caso não encontre um melhor, o laço de repetição termina retornando para S' a última melhor solução encontrada.

Algoritmo 3 BuscaLocal

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$ e uma solução S, sendo d sua densidade
// Nova solução S' recebe S
$S' = S$
// Enquanto encontrar um melhor vizinho, atualizar a solução, sendo dS' a densidade do vizinho
while dS' melhor que d **do**
  // Calcula a densidade do melhor vizinho
  $S' =$ MelhorVizinho(S')
end while
return S'

Já para encontrar o melhor vizinho, definido no Algoritmo 4, um laço itera sobre todos os vértices de todas as comunidades de uma solução. Para cada vértice, o mesmo é inserido em cada uma das outras comunidades da solução, e verifica-se o valor da densidade utilizando a função da Modularidade com o vértice na comunidade diferente da

sua original. Caso encontre uma comunidade melhor, ele é trocado de comunidade, do contrário, permanece na mesma. Nesse caso, o vértice troca de comunidade na primeira melhor, sem buscar o melhor valor possível.

Algoritmo 4 MelhorVizinho

Entrada: um grafo não-dirigido e ponderado $G = (V, E, w)$ e uma solução S, sendo d sua densidade
// Nova solução S' recebe S
$S' = S$
// Buscar a primeira melhor solução S'' em cada vértice vc de cada comunidade c
for all $c \in c1, c2 \ldots, communities$ **do**
  for all $cv \in cv1, cv2 \ldots, vertices$ **do**
    for all $nc \in nc1, nc2 \ldots, communities$ **do**
      // Adiciona v na próxima comunidade nc
      $nc.add(v)$
      // Calcula a densidade com v em nc e guarda a solução gerada, caso seja a melhor encontrada
      if $D_\lambda(S')$ melhor que $D_\lambda(S'')$ **then**
        return (S'')
      else
        $nc.remove(v)$
      end if
    end for
  end for
end for
return d

7.1. Experimentos

Para a realização dos experimentos, foram utilizadas instâncias reais de grafos utilizados no trabalho [De Santiago and Lamb 2020], sendo elas instâncias reais "Slovene Parliamentary Party" e "Gahuku-Gama Subtribes" (Tabela 1), onde serão referenciadas nessa seção como $Parlamento$ e $Gahuku$, respectivamente. A rede $Gahuku$ representa a relação das tribos da Nova Guiné [Read 1964], e a rede $Parlamento$ representa a relação entre dez partidos políticos do Parlamento Esloveno em 1994 [Kropivnik and Mrvar 1996]. Três conjuntos de configurações de experimento foram definidas para aplicar a heurística, onde cada configuração é dada por um valor de λ e um valor de m (que define a quantidade de soluções que serão criadas no Algoritmo 1). Cada configuração do experimento foi repetida uma quantidade de 30 vezes, uma vez que executar o experimento múltiplas vezes garante medidas de tempo de execução mais precisas, e também permite mais abrangência de resultados para realizar a análise considerando o componente randômico no Algoritmo 2). Os experimentos foram executados com o valor de m sendo definido como 20, 30 e 40, escolhidos arbitariamente, e o valor de λ variando para $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$.

Table 1. Quantidades de vértices e arestas de cada rede.

	vertices	arestas
gahuku.net	16	120
parlamento.net	10	45

Realizou-se também comparações dos valores obtidos pelos experimentos com os valores ideais encontrados em [De Santiago and Lamb 2020]. Para isso, a Figura 1 definida no estudo foi utilizada como referência de valores ideais de densidade (D) para cada valor de $lambda$ (λ).

Figure 1. Valores ideias de Gahuku-Gama Subtribes e Slovene Parliamentary Party.

λ	Slovene Parliamentary Party Time(s)	Slovene Parliamentary Party D*	Gahuku-Gama Subtribes Time(s)	Gahuku-Gama Subtribes D*
0.1	0.051	2.000	0.032	3.037
0.2	0.056	5.800	0.061	7.445
0.3	0.035	11.200	0.159	11.854
0.4	0.016	16.600	0.286	17.076
0.5	0.019	23.000	0.400	24.238
0.6	0.057	36.000	0.427	36.520
0.7	0.050	53.999	0.305	55.749
0.8	0.052	72.000	0.185	75.500
0.9	0.056	89.999	0.209	95.466

O desenvolvimento da heurística foi feito na linguagem Python. Os experimentos foram realizados de forma paralela utilizando a biblioteca ***multiprocessing*** do Python. O componente probabilístico do Algoritmo 2 foi obtido utilizando a função ***shuffle*** fornecido pela biblioteca ***random*** da linguagem Python para transformar uma lista de vértices em uma lista aleatória de vértices.

Abaixo a tabela com as médias de densidade obtidas por configuração, para cada rede estudada.

Table 2. Valores de densidade obtidos aplicando a heurística para cada instância e parâmetro.

lambda (λ)	**m**	D_λ **gahuku.net**	D_λ **parlamento.net**
0.1	**20**	-5.800	-4.178
	30	-5.800	-3.123
	40	-5.800	-1.302
0.2	**20**	0.511	5.335
	30	1.047	5.793
	40	2.774	5.787
0.3	**20**	9.120	11.200
	30	8.668	11.200
	40	9.815	11.200
0.4	**20**	16.480	16.600
	30	16.690	16.600
	40	16.588	16.600
0.5	**20**	24.238	23.000
	30	24.238	23.000
	40	24.238	23.000
0.6	**20**	36.520	36.000
	30	36.520	36.000
	40	36.520	36.000
0.7	**20**	55.750	54.000
	30	55.750	54.000
	40	55.750	54.000
0.8	**20**	75.500	72.000
	30	75.500	72.000
	40	75.500	72.000
0.9	**20**	95.467	90.000
	30	95.467	90.000
	40	95.467	90.000

A Figura 2 faz o comparativo entre as duas médias de densidade de cada rede.

Com base nesses experimentos é possível observar que a média dos valores da densidade (Tabela 2) para valores menores de λ (entre 0.1 e 0.4) ficam mais longe do valor ideal (solução ótima, vide Figura 1) para a rede $Gahuku$, que possui mais ligações que a rede $Parlamento$. Já para a rede $Parlamento$, a média dos valores da densidade para valores da λ até 0.2 ficam mais longe do valor ideal. A partir do λ definido como 0.5 as médias de densidade se estabilizam mais próximas do valor ideal para as duas redes.

Para a maioria dos cenários, houve ao menos alguma execução dentre as 30 realizadas onde o valor ideal é obtido com exatidão, como pode ser observado na Tabela 3 que demonstra o valor máximo obtido para cada configuração de teste. As exceções se dão na rede $Gahuku$ para λ de 0.1 e 0.2.

Figure 2. Gráfico de médias de densidade relacionado todas as redes do experimento.

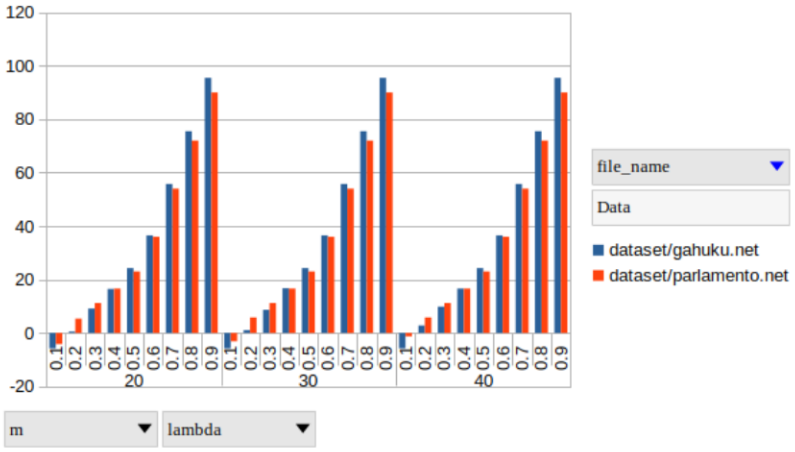


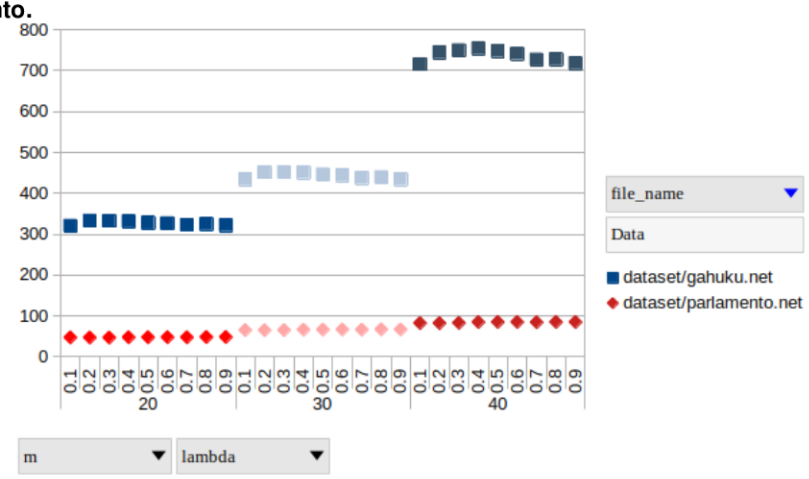
Table 3. Valores máximos de densidade obtidos aplicando a heurística para cada instância e parâmetro.

lambda	m	gahuku.net	parlamento.net
0.1	**20**	-5.800	2.000
	30	-5.800	2.000
	40	-5.800	2.000
0.2	**20**	3.800	5.800
	30	3.800	5.800
	40	3.800	5.800
0.3	**20**	11.854	11.200
	30	11.854	11.200
	40	11.854	11.200
0.4	**20**	17.076	16.600
	30	17.076	16.600
	40	17.076	16.600
0.5	**20**	24.238	23.000
	30	24.238	23.000
	40	24.238	23.000
0.6	**20**	36.520	36.000
	30	36.520	36.000
	40	36.520	36.000
0.7	**20**	55.750	54.000
	30	55.750	54.000
	40	55.750	54.000
0.8	**20**	75.500	72.000
	30	75.500	72.000
	40	75.500	72.000
0.9	**20**	95.467	90.000
	30	95.467	90.000
	40	95.467	90.000

Como esperado, quanto maior é o m na execução, maior também a média de tempo (observe na Figura 3). Além disso, mesmo as redes $Gahuku$ e $Parlamento$ tendo quantidades próximas de vértices, a quantidade de arestas implica diretamente no tempo de execução da heurístíca, sendo que quanto maior for a quantidade de ligações da rede, maior o tempo de execução. Os tempos de execução da Figura 3 se tratam de uma média de tempo de execução dentro de 30 execuções, portanto, uma nova configuração de testes foi executada para fazer uma comparação com os valores referência de 1 com apenas uma repetição, que será detalhada posteriormente.

Também é possível observar que, para as redes estudadas, valores maiores de m não tem influência significativa nos resultado das médias de densidade na aplicação da heurística, e não garantem que o melhor resultado de densidade será obtido aumentando o m. Ou seja, para a proposta deste trabalho, maiores quantidades de soluções geradas buscando encontrar a melhor comunidade individualmente para cada vértice não implica em uma eficácia maior, mas sim em um tempo de execução maior.

Figure 3. Gráfico de médias de tempo (s) relacionado todas as redes do experimento.



Além dos experimentos acima, para fazer uma comparação de tempo com os valores referência de 1, foram realizadas para cada rede no experimento execuções de apenas uma repetição, com o valor de m sendo definido como 20, 30 e 40, escolhidos arbitariamente, e o valor de λ variando para $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$.

Table 4. Valores tempos de execução em segundos obtidos aplicando a heurística para cada instância e parâmetro para apenas uma única repetição.

m	lambda	dataset/gahuku.net	dataset/parlamento.net
20	**0.1**	16.841	2.396
	0.2	16.527	1.915
	0.3	11.738	1.195
	0.4	9.068	1.990
	0.5	8.894	1.897
	0.6	4.707	1.705
	0.7	6.956	1.722
	0.8	9.308	2.268
	0.9	11.790	2.693
30	**0.1**	25.981	3.719
	0.2	26.540	1.896
	0.3	20.255	2.612
	0.4	15.732	3.120
	0.5	8.945	3.024
	0.6	11.317	2.723
	0.7	11.769	2.593
	0.8	16.083	3.938
	0.9	20.340	4.235
40	**0.1**	36.940	4.777
	0.2	35.775	3.002
	0.3	21.329	2.789
	0.4	19.009	3.878
	0.5	15.712	3.601
	0.6	16.538	3.400
	0.7	16.429	3.382
	0.8	27.410	4.613
	0.9	27.026	5.065

Analisando os tempos de execução nesse segundo experimento, a heurística desenvolvida teve uma demora consideravelmente maior em relação aos valores referência da Imagem 1. Isso se dá provavelmente devido aos valores de m, que implicam em mais repetições dentro da heurística. Observe que, novamente, quanto maior for o m, maior o tempo de execução. Além disso, o trabalho relacionado utilizado como referência de valores ótimos, teve seu desenvolvimento feito em outra linguagem, sendo ela o C. Um fator importante para um bom tempo de exeução dos experimentos utilizando essas redes é a máquina onde os experimentos estão sendo executados, pois como a análise é feita com execução paralela, quanto mais núcleos o computador fornecer, melhor a paralelização.

8. Conclusões

No presente trabalho foi realizado um levantamento teórico dos principais tópicos e estudos referentes ao problema da Modularidade por Densidade com Sinais, além de ressaltar a importância e aplicação em diversas áreas desses problemas na atualidade.

Foram apresentados três trabalhos relacionados para trazer algumas comparações e inspirações para desenvolver uma heurística para resolver o problema da Modularidade por Densidade com Sinais, sendo eles [Chen et al. 2017], [De Santiago and Lamb 2020] e [Zhou and Wang 2016]. Uma proposta foi modelada e desenvolvida, com baseada nos trabalhos de [Chen et al. 2017] e [De Santiago and Lamb 2020] utilizando conceitos de busca gulosa, aleatória e adaptativa (GRASP).

Seguindo do desenvolvimento da heurística, foi feita uma análise e comparação dos resultados com heurísticas e soluções já existentes para compreender em quais aspectos a heurística pode auxiliar no problema de otimização. A aplicação da heurística para duas instâncias reais ***Slovene Parliamentary Party*** e ***Gahuku-Gama Subtribes*** alcançou os valores de densidade considerados ideais referenciados pelo trabalho [De Santiago and Lamb 2020] em quase todos os cenários. No entanto, a estratégia de gerar um valor m de soluções não resulta necessariamente em uma boa média de densidade conforme m aumenta. A partir de um λ de 0.5, as médias de densidade se estabilizaram para as duas redes estudadas, onde o valor de m não teve impacto relevante. Os tempos de execução estão diretamente ligados com o valor de m, com a quantidade de execuções que a heurística é aplicada, e também com o tamanho da rede que está sendo analisada.

Por fim, este projeto pode ser futuramente expandido, utilizando-se em seus experimentos em redes maiores, com mais vértices e arestas, sejam elas reais ou artificiais. A heurística pode ser desenvolvida em alguma outra linguagem para observar se afetaria de forma significativa o tempo de execução, por exemplo. Outra abordagem que pode ser seguida em um projeto futuro seria experimentar outros critérios gulosos no seu desenvolvimento.

References

Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., and Liberti, L. (2010). Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112.

Brandes, U. (2008). On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188.

Chen, Y., Yan, J., Yang, Y., and Chen, J. (2017). Uncovering the community structure in signed social networks based on greedy optimization. *Modern Physics Letters B*, 31(14).

De Santiago, R. and Lamb, L. (2020). Exact signed modularity density maximization solutions and their real meaning*.

de Santiago, R. and Lamb, L. C. (2017). Exact computational solution of Modularity Density Maximization by effective column generation. *Computers & Operations Research*, 86:18–29.

de Santiago, R. and Lamb, L. C. (2020). A ground truth contest between modularity maximization and modularity density maximization. *Artificial Intelligence Review*.

Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71.

Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1):36–41.

Fortunato, S. and Castellano, C. (2012). *Community Structure in Graphs*, pages 490–512. Springer New York, New York, NY.

Gendreau, M. and Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. 2nd edition.

Kropivnik, S. and Mrvar, A. (1996). An analysis of the slovene parliamentary parties network. *Developments in Statistics and Methodology, Metodološki zvezki*, 12:209–216.

Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Signed networks in social media. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, page 1361, New York, New York, USA. ACM Press.

Li, Y., Liu, J., and Liu, C. (2014). A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks. *Soft Computing*, 18(2):329–348.

Li, Z., Zhang, S., Wang, R.-S., Zhang, X.-S., and Chen, L. (2008). Quantitative function for community detection. *Physical Review E*, 77(3):036109.

Maniezzo, V., Stützle, T., and Voß, S. (2009). *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Annals of Information Systems. Springer US.

Meunier, D., Fonlupt, P., Saive, A.-L., Plailly, J., Ravel, N., and Royet, J.-P. (2014). Modular structure of functional networks in olfactory memory. *NeuroImage*, pages 264–75.

Nepusz, T., Yu, H., and Paccanaro, A. (2012). Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471–472.

Newman, M. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113.

Read, K. E. (1964). Cultures of the central highlands, new guinea. *Southwestern Journal of Anthropology*, 10.

Rothlauf, F. (2011). *Design of Modern Heuristics: Principles and Application*. Springer Publishing Company, Incorporated, 1st edition.

Schmeja, S. (2011). Identifying star clusters in a field: a comparison of different algorithms. *Astronomische Nachrichten*, 332(2):172–184.

Zhou, D. and Wang, X. (2016). A neighborhood-impact based community detection algorithm via discrete pso. *Mathematical Problems in Engineering*, 2016.

APÊNDICE B – CÓDIGO FONTE

O código fonte do projeto está disponível no endereço: https://github.com/leticia-nascimento/SMDM-heuristics.