UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Comparando redes complexas para clustering de dados com métodos tradicionais

Gustavo Godoy de Lima

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Gustavo Godoy de Lima

Comparando redes complexas para clustering de dados com métodos tradicionais

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Diego Raphael Amancio

Versão original

São Carlos

2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Lima, Gustavo Godoy de
L732c Comparando redes complexas para clustering de dados com métodos tradicionais / Gustavo Godoy de Lima; orientador Diego Raphael Amancio. -- São Carlos, 2022.
76 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Redes complexas. 2. Detecção de comunidades. 3. Aprendizado computacional. 4. Inteligência artificial. I. Amancio, Diego Raphael, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Gustavo Godoy de Lima

Comparing complex networks for data clustering with traditional methods

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Diego Raphael Amancio

Original version

São Carlos

2022

Dedico este trabalho aos meus pais, Ubiratan e Telma, que foram essenciais na minha formação, às minhas filhas, Clara e Bruna, pela compreensão nos momentos em que a dedicação aos estudos foi exclusiva e a minha esposa Carla pelo apoio.

AGRADECIMENTOS

A todos aqueles que contribuíram, de alguma forma, para a realização deste trabalho.

A todos meus familiares que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

Aos meus colegas de turma Antonio Alves Lopes Filho, Robson Buratti e Henrique Pedrosa Chagas por me auxiliarem com ensinamentos quando precisei.

A Agência Nacional de Energia Elétrica (ANEEL) pelo Programa de Incentivo Educacional que possibilitou a minha participação nesse excelente curso.

A todos os professores do MBA em Inteligência Artificial e Big Data que comprovaram, por meio desse curso, porque o Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo de São Carlos é referência no assunto.

"Só se pode alcançar um grande êxito
quando nos mantemos fiéis a nós mesmos."
Friedrich Nietzsche

RESUMO

LIMA, G. G. **Comparando redes complexas para clustering de dados com métodos tradicionais**. 2022. 76p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

A demanda crescente por métodos mais ágeis e eficientes na detecção de comunidades em grandes volumes de dados conduz a busca de novos algoritmos que atendam essas necessidades. Para tanto, os métodos baseados em redes complexas merecem estudos para que se atestem seus desempenhos frente a métodos tradicionais já disponíveis no mercado. A busca de clustering de dados, que se dá por aprendizado de máquina não supervisionado, é essencial para que se atinja os objetivos no mundo dominado pelo Big Data onde o volume de dados cresce exponencialmente assim como surgem novos tipos de dados.

Palavras-chave: Redes complexas. Aprendizado de máquina não supervisionado. Detecção de comunidades.

ABSTRACT

LIMA, G. G. **Comparing complex networks for data clustering with traditional methods**. 2022. 76p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2022.

The growing demand for more agile and efficient methods for detecting communities in large volumes of data leads to the search for new algorithms that meet these needs. Therefore, methods based on complex networks deserve studies to attest to their performance against traditional methods already available on the market. The pursuit of data clustering, which takes place through unsupervised machine learning, is essential to achieve the goals in a world dominated by Big Data where the volume of data grows exponentially as new types of data emerge.

Keywords: Complex networks. Unsupervised machine learning. Community detection.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentados justificativas e motivos para o desenvolvimento deste estudo, além dos objetivos definidos a partir de uma questão de pesquisa. No final é disposta a organização da presente tese.

1.1 Contextualização

A gradativa geração e acumulação de dados realizada pela humanidade, principalmente com a popularização da internet, conduziu a criação de diversos métodos de *clustering* de dados nos últimos anos que buscam extrair conclusões e informações relevantes sobre esses dados. Métodos mais ágeis e precisos são cada vez mais demandados, considerando que o volume desses dados cresce de forma não-linear e em velocidade, até pouco tempo atrás, inimaginável.

Paralelamente, uma nova área vem se desenvolvendo devido a observação empírica de que tudo que nos cerca apresenta relações em rede que podem ser representadas em forma de grafos. Essa área é conhecida como teoria das redes complexas e apresenta promissoras aplicações no reconhecimento de comunidades em bases de dados.

Neste projeto de pesquisa serão avaliadas algumas técnicas de *clustering* de dados utilizando a teoria de redes complexas, onde se buscará reconhecimento de padrões em alguns conjuntos de dados por meio de algoritmos específicos. Com este trabalho, espera-se contribuir com o universo do *clustering* de dados ao se pesquisar a eficiência de métodos de redes quando comparados com os métodos tradicionais.

1.2 Justificativa e Motivação

Identificar padrões em bases de dados é uma necessidade do homem moderno, pois o volume de dados gerado pela humanidade se avoluma em grandezas absurdas, de modo não linear e, muitas vezes, não homogêneo. Esses dados, que há algumas décadas poderiam parecer irrelevantes, ganharam espaço no mundo moderno quando grandes conglomerados do setor de tecnologia perceberam que era possível extrair informações úteis visando principalmente o lucro. A importância desse cenário é tanta que a era em que vivemos é classificada, por estudiosos, como a era da informação. Objetivando, portanto, resolver problemas de agrupamento de dados em bases distintas, diversas técnicas de *clustering* de dados foram desenvolvidas nos últimos anos. No entanto, há uma necessidade de se desenvolver algoritmos cada vez mais velozes e precisos devido a esse imenso volume de informação que cresce sem limites. Tais algoritmos devem buscar uma generalização em suas aplicações.

Dado que um dos principais desafios é ter que aplicar métodos de aglomeração ágeis e concisos em distintas bases de dados, os métodos baseados em redes merecem uma análise comparativa frente aos métodos tradicionais de modo a aferir se os seus desempenhos são aceitáveis e necessários para o aprofundamento de estudos nesse campo. Neste contexto, é proposto o uso de algoritmos baseados em métodos de redes complexas para buscar comunidades, de modo generalizado, em diferentes conjuntos de dados. Espera-se que as aplicações atinjam desempenhos comparáveis ao de algoritmos tradicionais.

1.3 Questão de Pesquisa e Objetivos

Neste trabalho, espera-se que os métodos de detecção de comunidades baseados em redes apresentem resultados satisfatórios quando aplicados a bases de dados diversas. Diante do incipiente uso dessas técnicas na atualidade para sistemas de detecção de comunidades, foi elaborada a seguinte questão de pesquisa que norteará este projeto:

Q1 “Considerando o amplo uso de métodos que não utilizam técnicas de redes, transformar em grafos tem ganho ou não na detecção de comunidades?”

Diante desta questão de pesquisa, são definidos os seguintes objetivos para o desenvolvimento deste trabalho:

- Mapear algoritmos de *clustering* baseados em redes disponíveis na literatura, analisando suas lacunas e desempenhos na detecção de comunidades. Este objetivo está relacionado à questão de pesquisa Q1.
- Comparar os desempenhos obtidos aos desempenhos de algoritmos de *clustering* tradicionais. Este objetivo está relacionado à questão de pesquisa Q1.

A partir do estudo proposto, espera-se que os resultados apontem a importância ou não do aprofundamento do uso de redes complexas para *clustering* de dados.

1.4 Organização do trabalho

No capítulo seguinte, será realizada uma revisão de conceitos e trabalhos relacionados ao tema. Para tanto, será definido o conceito de rede complexa, seguido por sua representação, pela detecção de comunidades em redes, por medidas e modelos de redes. Uma breve revisão de aprendizado de máquina baseado em redes seguirá o tópico anterior, onde serão abordados a formação do grafo a partir de um conjunto de dados e como se realiza a detecção de comunidades. No final do capítulo 2, demonstra-se uma revisão dos principais trabalhos da literatura.

No capítulo 3 será apresentada uma breve descrição do problema tratado no projeto, quais foram as atividades desempenhadas e algoritmos implementados, bem como serão

exibidos os resultados em forma de gráficos que, no caso dos métodos baseados em redes complexas, são grafos com vértices relacionados por arestas. Os vértices desses grafos, quando da aplicação dos algoritmos de detecção de comunidades, estarão com cores que marcaram as comunidades atribuídas a cada um deles pelos algoritmos de forma que possamos ter uma interpretação visual do processo.

Após os itens citados, teremos uma apresentação, no capítulo 4, dos resultados obtidos durante este projeto, já indicando quais se destacaram por meio de análise desses resultados.

No capítulo final será feito uma interpretação dos procedimentos e resultados de todo o processo.

2 REVISÃO DE CONCEITOS E TRABALHOS RELACIONADOS

O objetivo nesse capítulo é fazer uma revisão da teoria das redes complexas, além de apresentar, ao final, alguns trabalhos relacionados ao tema de estudo proposto nessa monografia.

2.1 Redes complexas

Apresentando características topológicas não triviais, as redes complexas não são nem completamente regulares nem completamente aleatórias e envolvem conceitos de estatística, sistemas dinâmicos e teoria dos grafos. As redes complexas são descritas como um objeto composto de elementos e as conexões entre estes elementos. Matematicamente, essas redes podem ser modeladas por grafos.

A teoria dos grafos é uma forma natural e matemática adequada para representar as redes complexas. Esses grafos podem ser definidos como estruturas compostas por conjunto de vértices (ou nós) ligados por arestas que são usadas para indicar alguma relação entre os nós. Na Figura 1 representa-se um grafo com 8 nós e 10 arestas.

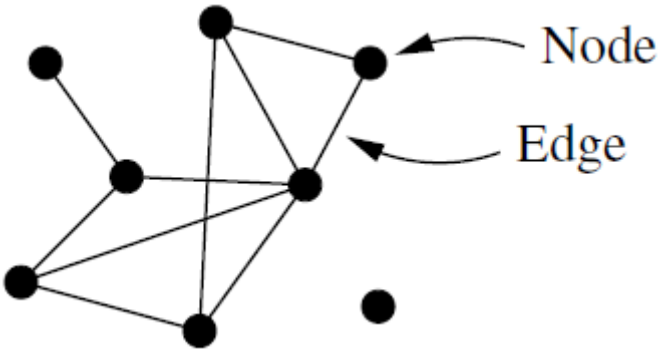

Figura 1 – Elementos de um Grafo, reproduzido de (NEWMAN, 2010)

Há várias formas de se classificar e analisar um grafo. De acordo com o tipo de aresta admitido, por exemplo, os grafos podem ser classificados em direcionados, não direcionados ou mistos. Os grafos podem ser não direcionados se suas arestas apresentam relação simétrica. Contudo, se suas arestas apresentam origem e destino, serão classificados como direcionados ou dígrafos. Ainda podem ter a classificação de grafos mistos se admitir tanto arestas direcionadas quanto não direcionadas. Na Figura 2, há exemplos de vários tipos de redes: (a) uma rede não direcionada com apenas um tipo de vértice e um único tipo de aresta; (b) uma rede com vários tipos discretos de vértices e arestas; (c) uma rede com pesos variados de vértices e arestas; (d) uma rede direcionada na qual cada aresta tem uma direção.

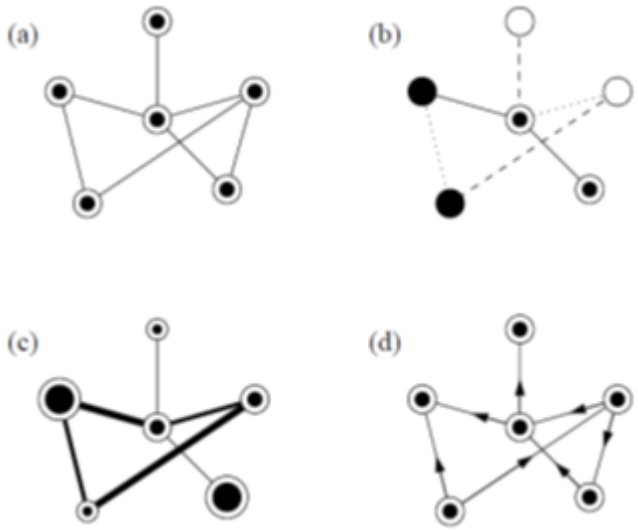

Figura 2 – Tipos de Rede, reproduzido de (NEWMAN, 2003)

2.1.1 Representação de redes complexas

De modo matemático, um grafo é um par de conjuntos G = (V, E), onde V é o conjunto de vértices e E é o conjunto de arestas do grafo. O número de vértices indica a ordem do grafo enquanto o tamanho do grafo é dado pelo número de arestas. Em grafos não direcionados, cada aresta é representada por um par não ordenado, enquanto em grafos direcionados (dígrafos) os pares são ordenados. Em um grafo com pesos é definida uma função para designar um peso para cada aresta.

Dizemos que dois vértices v e u são vizinhos, ou adjacentes, caso $(u, v) \in E$. O conjunto de vizinhos de v é chamado vizinhança de v e pode ser denotado por $\Gamma(v)$. O número de arestas que incidem sobre um dado vértice v determina o grau de v que é denotado como deg(v), no caso do grafo não direcionado. Caso o grafo seja direcionado, o grau de um vértice é a soma das arestas que chegam subtraídas das que saem do vértice. Ainda temos o grafo regular se todos os vértices têm o mesmo grau.

Ao modelarmos redes complexas por meio de grafos, os elementos da rede são representados por vértices e a similaridade entre pares é representada pelas conexões ou pelo peso das conexões no caso de conexões com peso. Essa similaridade entre um par de vértices pode ser representada através da distância entre eles dado um ou mais atributos numéricos dos elementos da rede. Tais medidas de distância devem obedecer aos seguintes critérios: (i) a distância de um vértice para ele mesmo é zero; (ii) as distâncias devem ser simétricas; e (iii) a desigualdade triangular deve ser válida, ou seja, a distância entre dois vértices será sempre menor que a soma das distâncias de cada um desses vértices para um terceiro vértice.

Como, em determinadas aplicações, os vértices sozinhos não têm informações suficientes para o cálculo de uma matriz de similaridades, as arestas podem ser utilizadas

para derivar medidas similares. Para isso, pode-se calcular a similaridade entre dois vértices usando apenas a informação de adjacência, verificando-se a sobreposição de suas respectivas vizinhanças. Para a compreensão de como seria essa matriz de adjacência, se atribuirmos o valor 1 para o caso de existir uma aresta entre dois vértices e 0 para caso contrário então para o grafo da Figura 3 teremos a matriz de adjacência da Figura 4

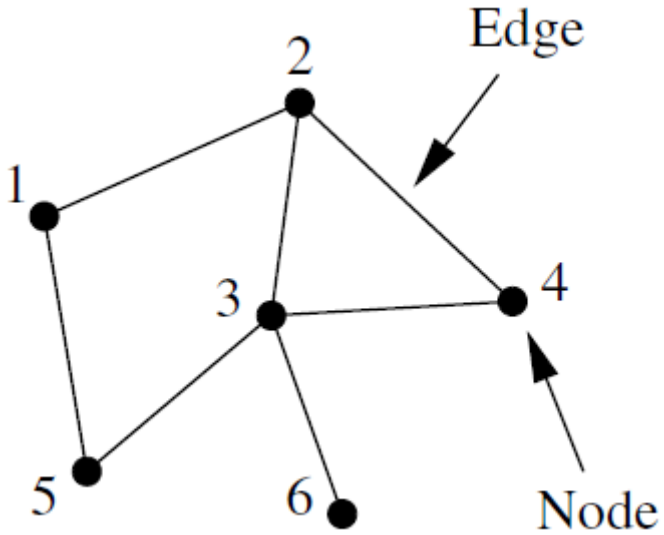

Figura 3 – Grafo exemplo, reproduzido de (NEWMAN, 2018)

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figura 4 – Matriz de Adjacência para o Grafo do exemplo anterior, reproduzido de (NEWMAN, 2018)

Caso a ligação entre os vértices fosse ponderada, no lugar dos valores unitários, a matriz poderia considerar cada um desses pesos de cada aresta que liga os vértices. Do mesmo modo, poderiam ser considerados na matriz de adjacência o número de arestas entre cada vértice no caso de um grafo com múltiplas arestas. Também pode-se fazer o uso da matriz de adjacência para indicar a direção de arestas em grafos direcionados.

2.1.2 Detecção de comunidades em redes

Uma propriedade bastante presente em redes complexas é a presença de estruturas modulares chamadas de comunidades, que são como sub-grafos onde existem muitas conexões entre os elementos de um mesmo grupo, conhecida por similaridade, e poucas entre elementos de grupos distintos, conhecida por dissimilaridade. A decomposição de uma rede em comunidades é importante para a compreensão das relações entre diferentes componentes. Na Figura 5 é possível observar essas comunidades em um grafo.

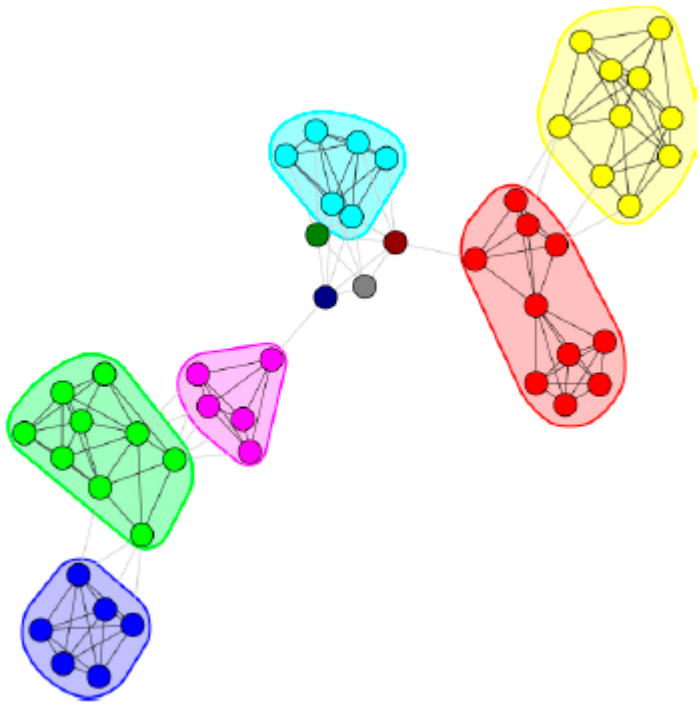

Figura 5 – Sub-grafos ou comunidades, reproduzido de (JUNIOR, 2018)

Basicamente, os métodos de detecção de comunidades em redes complexas podem se basear na adição, métodos aglomerativos, ou remoção, métodos divisivos, de arestas da rede. No método aglomerativo, arestas são adicionadas de acordo com alguma medida de similaridade calculada entre os pares de nós, a partir do par de maior similaridade. No método divisivo, os pares de nós menos similares são identificados e a aresta que os une é removida. Esse procedimento é repetido dividindo a rede em comunidades cada vez menores.

Algoritmos desenvolvidos para identificação da comunidade também podem ser usados para partição de grafos e avaliação de *clusters*. Desse modo, como os algoritmos de clusterização hierárquica, a divisão da rede em comunidades pode ser ilustrada por um dendrograma, conforme Figura 6, onde cortes horizontais revelam diferentes divisões da rede em comunidades e os cortes mais próximos à base do dendrograma mostram um maior grupo de comunidades formadas por poucos nós.

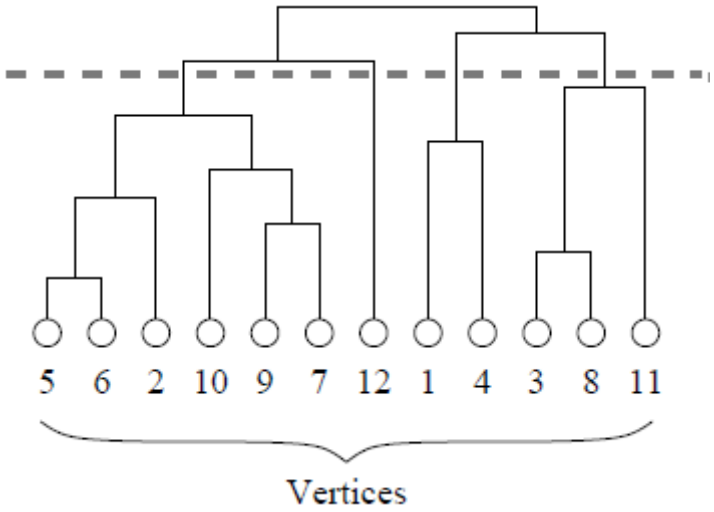

Figura 6 – Dendrograma, reproduzido de (NEWMAN, 2010)

Informalmente, a detecção será um problema onde encontraremos conjuntos de vértices que se ligam mais entre eles mesmos do que com vértices pertencentes a outros conjuntos. Portanto, diferentemente do particionamento de grafos, na detecção de comunidades não é informado o número final de comunidades ou o número de membros de cada comunidade.

Para calcularmos quão bem dividida uma rede está, usamos o conceito de modularidade. Supondo que a conexão, para cada par de vértices, indica que eles fazem parte de uma comunidade, chegamos a um limite superior de modularidade. Apesar de não ser verdade a conclusão anterior, ela fornece um valor que a ser utilizado para verificar quão boa são as comunidades propostas para a rede. A modularidade M é definida conforme Equação 2.1, onde m representa a quantidade de arestas na rede, A_{ij} representa a matriz de adjacência, k_i representa o grau do nó i, k_j representa o grau do nó j e $\delta_{s_i s_j}$ é o delta de Kronecker que será 1 se os nós estiverem na mesma comunidade e 0 se em comunidades distintas.

$$M = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta_{s_i s_j} \tag{2.1}$$

2.1.3 Medidas de redes complexas

As estruturas das redes complexas são caracterizadas pelo uso de medidas. Desse modo, portanto, são analisadas as propriedades estatísticas que descrevem a estrutura e o comportamento de sistemas em rede.

Dentre diversas medidas de rede, destacam-se: o grau que quantifica estatisticamente o grau médio dos vértices em uma rede; a assortatividade que quantifica a tendência de conexões entre vértices; o coeficiente de agrupamento que quantifica o grau para o qual vértices locais em uma rede tendem a se agrupar; a proximidade que mede o inverso do

caminho mínimo (conhecido como distância geodésica) médio entre um nó e todos demais nós da rede; o grau de intermediação ou intermedialidade (amplamente conhecida como *betweenness*) que indica a centralidade de vértices na rede; o grau de proximidade de um vértice que seria a média dos caminhos mínimos de determinado vértice para qualquer outro; a modularidade que quantifica quão boa é uma determinada divisão da rede; menor caminho médio que quantifica o caminho médio entre um dado nó e todos os outros; e o PageRank que trata-se de uma medida de centralidade baseada em passeios aleatórios.

Em redes com pesos, a força do vértice é uma medida importante. Ainda, se gerarmos um histograma para todos os vértices da rede baseado em suas probabilidades de possuir determinado grau, teremos uma distribuição de graus que, de acordo com a complexidade da rede, pode seguir uma lei de potências.

Buscando-se os vértices mais importantes de uma rede, ganham visibilidade medidas de centralidade. Quando se entende que um vértice importante é um vértice que tem bastantes ligações na rede, usa-se a centralidade de grau como medida. Contudo, se a importância dos vértices conectados a um vértice em análise também possui importância, recomenda-se utilizar a centralidade de autovalor. Em redes direcionais, a solução é a utilização da centralidade de Katz para que vértices unidirecionais de origem não sejam negligenciados.

Como, na centralidade de Katz, os vértices importantes espalham importância igualmente, independente do grau de saída, os fundadores da Google, Larry Page e Sergey Brin, desenvolveram uma importante medida de centralidade conhecida como PageRank que busca espalhar a importância proporcionalmente. o modelo acabou evoluindo para o Personalized PageRank (PPR) onde busca-se saber a importância do ponto de vista de cada vértice.

2.1.4 Modelos de redes complexas

Os modelos de rede estão normalmente relacionados ao entendimento do significado das propriedades que são percebidas por meio das medidas de rede. Dessa forma, alguns modelos de redes complexas se destacam na literatura, sendo eles: Redes Aleatórias, Redes Pequeno Mundo, Redes Livres de Escala, Redes Modulares e Otimização Estrutural de Redes.

Os modelos de redes aleatórias são muito utilizados no estudo de sistemas reais, onde as topologias são complexas ou os princípios de organização não são conhecidos. Nesses modelos de rede, as conexões entre os vértices são realizadas de modo aleatório a partir de um conjunto específico de parâmetros definidos. Esse modelo foi primeiramente proposto por dois matemáticos húngaros (ERDŐS; RÉNYI, 1961). Na Figura 7 exemplifica-se com uma rede aleatória.

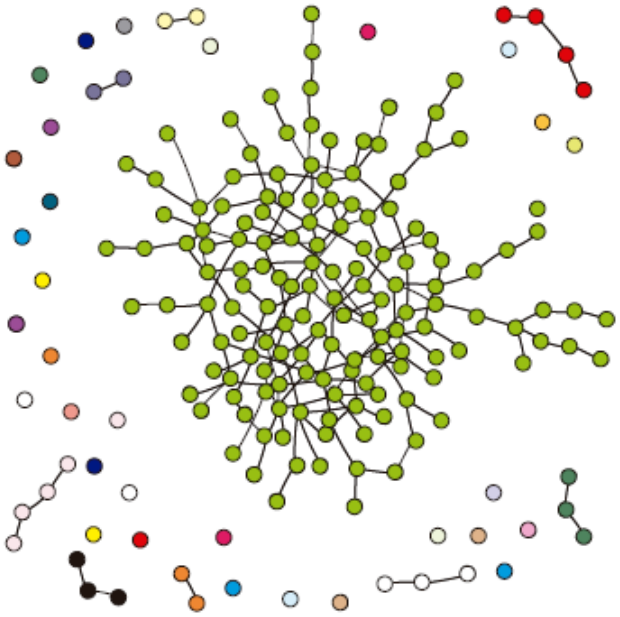

Figura 7 – Exemplo de rede aleatória, reproduzido de (STROGATZ, 2001)

As redes do tipo pequeno mundo têm suas propriedades observadas em redes onde a maioria dos vértices pode ser alcançada por outros com um pequeno número de conexões. A partir de um estudo realizado pelo psicólogo Stanley Milgram, que conduziu o experimento "pequeno mundo" na década de 1960, concluiu-se a distância entre quaisquer duas pessoas no mundo seria de seis passos ou indivíduos. Em resumo, o modelo de rede pequeno mundo é caracterizado por registrar alto grau de agrupamento e baixa distância média entre os vértices da rede. Na Figura 8 tem-se um exemplo de rede pequeno mundo.

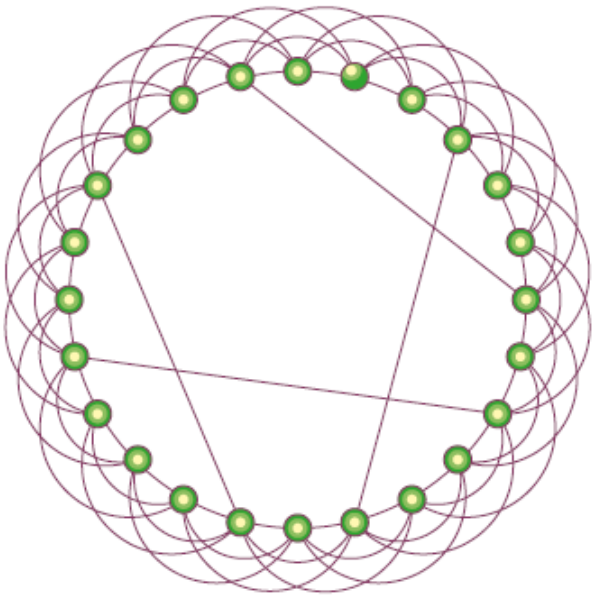

Figura 8 – Exemplo de rede pequeno mundo, reproduzido de (STROGATZ, 2001)

As redes livres de escala são redes complexas cujo grau de distribuição segue a lei de potência, em que a maioria dos vértices tem poucas ligações, contrastando com a existência de alguns vértices que apresentam um elevado número de ligações. Assim, um vértice com grau alto tende a ligar-se a outro nó de grau alto. Como a probabilidade de um vértice se ligar a outro é diretamente proporcional ao seu grau, as redes livres de escala são dominadas por um número pequeno de nós. Essa propriedade presente nas redes livres de escala, onde arestas de um novo vértice adicionado à rede se conectam, preferencialmente, aos vértices que possuem maior grau é denominada conexão preferencial. Albert-László Barabási e Réka Albert (BARABÁSI; ALBERT, 1999) propuseram o primeiro algoritmo, denominado modelo Barabási-Albert, para a construção de redes livre de escala. A Figura 9 demonstra uma rede livre de escala.

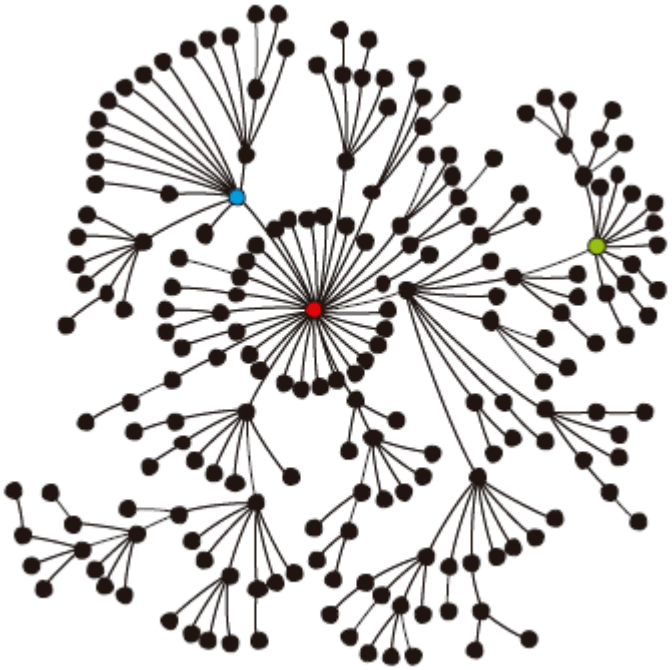

Figura 9 – Exemplo de rede pequena escala, reproduzido de (STROGATZ, 2001)

As redes modulares caracterizam-se pela presença de estruturas locais denominadas módulos ou comunidades que é definida como um grupo de vértices densamente conectados, onde conexões entre grupos distintos são esparsas. Um modelo para gerar redes modulares foi originalmente proposto em (GIRVAN; NEWMAN, 2002), com o objetivo de avaliar métodos para detecção de comunidades em redes.

Na otimização estrutural de redes, percebe-se que a otimização de uma meta ou certas características pode ser desejável para determinados problemas, em vez de se formar uma rede em função de algum mecanismo de crescimento ou da exploração de medidas de rede.

2.2 Aprendizado de máquina baseado em redes

Nesta seção são apresentadas técnicas importantes para o desenvolvimento do trabalho. As aplicações das técnicas contemplam conceitos essenciais para o desenvolvimento da tese. A seção é organizada conforme segue:

- A subseção 2.2.1 apresenta uma breve ideia da utilização do KNN como método para construção de redes a partir de uma base de dados cujas instâncias são diferenciadas por meio de uma medida de distância que permita atestar a similaridade entre as instâncias.
- São explicados como funcionam os algoritmos utilizados para detecção de comunidades são apresentadas na subseção 2.2.2.

2.2.1 Formação do grafo a partir de um conjunto de dados

Para construção das redes, utiliza-se o kNN (*k-nearest neighbors algorithm*) para construir as arestas entre os vértices. O algoritmo inicia fixando um ponto no plano e calculando a distância euclidiana em relação aos demais vértices no plano. Posteriormente, são selecionados um determinado número de vértices mais próximos deste ponto fixado e, então, são criadas arestas entre estes vértices e o ponto fixado. Tal procedimento é repetido para todos os pontos no plano. No entanto, para o desenvolvimento dessa técnica, primeiramente, os dados, a partir de seus atributos, tem suas distâncias referenciadas em relação às demais instâncias da base de dados.

No nosso trabalho foi desenvolvido um algoritmo, que será explicado no capítulo de metodologia, onde se combina a técnica de kNN com a de raio para se determinar uma matriz de similaridade para os vértices. Essa matriz, conforme explicado, deriva de uma distância escolhida para se entender quão similar ou dissimilar seria cada instância da base de dados em relação às demais.

Apesar da possibilidade de utilização o iGraph do Python, no desenvolvimento do projeto foi utilizada a biblioteca NetworkX que assim como a anterior tem diversas funcionalidades úteis ao projeto, além de ser de fácil utilização.

2.2.2 Detectando as Comunidades

Nesse ponto já se sabe que a modularidade é um conceito de suma importância dado que o trabalho visa comparar métodos baseados em redes complexas com métodos tradicionais e, no caso das redes complexas, a modularidade indica quão bem dividida está uma rede. A princípio, essa medida foi proposta por (CLAUSET; NEWMAN; MOORE, 2004), onde foi utilizada para detecção de comunidades no contexto de aprendizado não supervisionado.

Há alguns métodos que podem ser utilizados durante este projeto e que merecem explicação. Após a execução dos algoritmos citados abaixo, é esperado encontrar o grafo dividido em comunidades.

O *Edge Betweenness* é uma medida que indica a centralidade de uma aresta ou de um vértice em uma rede. Sua medida é baseada nos menores caminhos do grafo. O algoritmo de detecção de comunidades proposto por (GIRVAN; NEWMAN, 2002) utiliza essa medida. O método consiste em calcular a medida de *betweenness* para todas as arestas do grafo, obtendo-se a aresta com maior valor de *betweenness* da rede e retirando-a do grafo. Após a remoção, repete-se o procedimento, com o algoritmo terminando quando a modularidade atinge um valor máximo. Ao final, as componentes ainda conectadas formam as comunidades da rede.

O método proposto por (GIRVAN; NEWMAN, 2002), conhecido como *Fast Greedy*, baseia-se no aumento da modularidade. Seu algoritmo se resume aos vértices da rede não possuírem arestas no início, sendo o número de comunidades na rede incialmente igual ao número de vértices. A cada passo, duas comunidades são agrupadas caso exista pelo menos uma aresta entre quaisquer dois elementos destas diferentes comunidades.

O *Walktrap* foi proposto por (PONS; LATAPY, 2005) e utiliza os caminhos mínimos de um grafo para o seu funcionamento, realizando caminhadas aleatórias na rede para geração das comunidades. O primeiro passo é atribuir uma comunidade diferente para cada vértice da rede, e posteriormente, são executadas uma sequência de passos, que consistem em agrupar duas comunidades adjacentes.

O Método *Leading Eigenvector* foi proposto por (NEWMAN, 2006) e consiste em aumentar a modularidade através do cálculo dos autovetores e autovalores da matriz de modularidade da rede.

Proposto por (RAGHAVAN; ALBERT; KUMARA, 2007), no *Label Propagation* são definidas comunidades para todos os vértices da rede. Após, cada vértice passa a pertencer a comunidade que ocorre mais frequentemente entre seus vizinhos. A escolha de um vértice é feita de modo aleatório e o método é finalizado se cada nó tiver um rótulo que o número máximo de seus vizinhos tenha.

O *Infomap* é um algoritmo que utiliza caminhadas aleatórias para analisar o fluxo de informação na rede. O algoritmo foi proposto em (LANCICHINETTI; FORTUNATO, 2009).

Um algoritmo muito útil à detecção de comunidades é o Louvain (BLONDEL *et al.*, 2008), conhecido também como *Fast Unfolding*, por ser rápido e eficiente, já foi aplicado em redes com milhões de nós em poucos minutos. Na primeira etapa do algoritmo é atribuída uma comunidade diferente a cada nó em uma rede. Então, para cada nó, é avaliado o ganho em modularidade removendo o nó analisado de sua comunidade e colocando-o em outra

comunidade. O nó é, então, colocado na comunidade para a qual ganha modularidade máxima, mas o ganho deve ser positivo. Caso o ganho for negativo, o nó permanece na mesma comunidade. Na segunda etapa do algoritmo, uma nova rede é construída, mas os nós, nesse momento, são as comunidades encontradas na primeira fase. Os pesos da ligação entre os nós são dados pela soma dos pesos das ligações entre os nós nas duas comunidades correspondentes. A ligação entre nós da mesma comunidade leva a auto-loops para a comunidade na nova rede. Finalizada a segunda etapa, repete-se o procedimento desde a primeira etapa até que nenhuma outra melhoria possa ser alcançada. O ganho em modularidade é avaliado removendo o vértice de sua comunidade e movendo-o para uma comunidade vizinha. Se o ganho for positivo, esse nó é colocado na comunidade vizinha.

Como vantagens do *Fast Unfolding*, temos que suas etapas são intuitivas e fáceis de implementar e o resultado não é supervisionado. Além disso, o algoritmo é extremamente rápido. Ainda se observa que simulações de computador em redes modulares muito grandes sugerem que sua complexidade é linear nos dados típicos e esparsos. Isso pode ocorrer porque o ganho na modularidade é fácil de calcular e o número de comunidades diminui drasticamente após apenas algumas passagens. No entanto, a limitações do algoritmo seria que a otimização da modularidade falha em identificar comunidades menores do que uma determinada escala. Portanto, isso causa um limite de resolução na comunidade calculada usando uma abordagem de otimização de modularidade pura.

Como as escalas das redes estão se tornando maiores, é necessário um método eficiente para representar e analisar as redes complexas. Reduzir uma rede de larga escala a um *backbone* essencial pode ajudar a resolver os conflitos entre a grande escala das redes complexas e o entendimento da estrutura da rede. O *backbone* de uma rede é um componente central que é extraído filtrando informações redundantes da rede e preservando muito menos arestas e vértices da rede original.

Os métodos de filtragem para extração de *backbone* podem ser divididos em duas categorias principais: métodos globais e métodos locais. Alguns métodos globais usam certas medidas globais para filtrar as arestas, como o método baseado intermediação e o método baseado no peso das conexões. Esses métodos aplicam um limite global nos pesos ou na intermediação das conexões de forma que apenas aqueles que excedem o limite são preservados. Em suma, o principal objetivo da extração de *backbones* é reduzir o número de arestas nas redes, mantendo mais vértices para que se possibilite a simplificação da análise de informação relevante de determinada rede.

2.3 Revisão dos principais trabalhos da literatura

O principal trabalho (ARRUDA; COSTA; RODRIGUES, 2018) analisou algumas medidas de similaridade para representar um conjunto de dados como uma rede e comparou cinco algoritmos de detecção de comunidades diferentes para obter os *clusters*. O trabalho

concluiu que, quando comparado com os métodos tradicionais de *clustering*, a abordagem baseada em rede encontrava *clusters* com menores taxas de erro e menor tempo de processamento. Outra conclusão seria que a metodologia permite a identificação do número de *clusters* automaticamente levando em consideração o valor máximo da medida de modularidade.

Ao comparar diferentes métricas, observou que a exponencial da distância de Minkowski era a métrica mais adequada para quantificar a similaridade entre objetos em termos de seus vetores de características. Dentre os algoritmos de identificação de comunidades, o *fastgreedy* se mostrou o mais adequado, devido à sua precisão e rapidez de execução.

Outro trabalho (JUNIOR, 2018) foi desenvolvido em uma monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação (ICMC-USP). O trabalho constatou que para uma rede com mais de 1000 vértices há queda de desempenho no uso do kNN. Ademais, afirma que a maioria dos algoritmos de detecção de comunidades não são determinísticos, ou seja, para dadas duas entradas, as saídas podem ser diferentes.

3 METODOLOGIA

Considerando toda a exposição do capítulo anterior, primeiramente, devemos desenvolver códigos mais tradicionais de *clustering* de dados para que, em uma etapa posterior, possamos comparar o desempenho desses algoritmos tradicionais com os algoritmos baseados em redes complexas.

Ao desenvolver e testar o primeiro estágio, partimos para desenvolver um código que seja capaz de transformar as bases de dados em uma matriz de similaridades para, a partir dessa matriz, construir uma matriz de adjacência. Com a matriz de adjacência é possível conhecer os graus de cada nó de modo a ligá-los nos demais conforme a matriz. Para tanto, optou-se pela busca por nós mais próximos localizados a um raio determinado de distância, seguido por um algoritmo KNN para se atingir, pelo menos, um número mínimo de nós próximos a cada nó analisado.

Concluídas as etapas precedentes, a proposta seria avaliar algumas técnicas de identificação de comunidades em redes complexas, buscando reconhecer padrões em alguns conjuntos de dados por meio de algoritmos específicos. Após, deverá ser feita uma comparação de desempenho com algoritmos tradicionais, onde o trabalho estará concluído.

Em uma etapa prévia a construção dos algoritmos, decidiu-se pela escolha de bases de dados amplamente utilizadas no estudo de *machine learning* tanto para classificação quanto para agrupamento de dados. Após, foram aplicados algoritmos tradicionais de agrupamento sobre elas, seguido dos algoritmos e técnicas de redes complexas sobre esses mesmos dados e, posteriormente, foram realizadas comparações entre os resultados.

3.1 Escolha das bases de dados

Por estarem presentes no *scikit-learn*, biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python amplamente utilizada em estudos de técnicas de *machine learning*, foram escolhidas as seguintes bases: *Iris Plant, Wine Recognition* e *Breast Cancer Wisconsin.*

O *dataset Iris Plant* é provavelmente a base de dados mais utilizada para estudos referentes a classificação e agrupamento por ser uma base simples com 150 registros contendo apenas 4 atributos, quais sejam: comprimentos e larguras de pétalas e sépalas. Como, para se identificar o tipo de flor (setosa, virgínica e versicolor), todos atributos são importantes e pretendemos não trabalhar com a identificação do tipo para simular um agrupamento, serão adotados todos os atributos.

Tabela 1 – Atributos do *dataset Iris Plant*

Nome	**Tipo**	**Descrição**	**Valores**
Sepal length in cm	Numérico	Comprimento da sépala em cm	4,3 - 7,9
Sepal width in cm	Numérico	Largura da sépala em cm	2,0 - 4,4
Petal length in cm	Numérico	Comprimento da pétala em cm	1,0 - 6,9
Petal width in cm	Numérico	Largura da pétala em cm	0,1 - 2,5

Fonte: Elaborada pelo autor.

O *dataset Wine Recognition* é uma base de dados com mais instâncias e atributos que o *Iris Plant*, de modo que seus atributos podem trazer uma maior complexidade na detecção de informações. São 13 atributos para representar constituintes encontrados em cada um dos três tipos de vinhos da base. Esses 13 constituintes são álcool, ácido málico, cinzas, alcalinidade das cinzas, magnésio, fenóis totais, flavonóides, fenóis não flavonóides, proantocianinas, intensidade da cor, matiz, OD280 / OD315 de vinhos diluídos e prolina. Esta base de dados é composta por 178 exemplos de vinhos cultivados na Itália derivados de três cultivares diferentes, em que 59 são da classe 1, 71 da classe 2 e 48 da classe 3. Serão adotados todos os atributos para facilitar a busca de similaridades entre as tuplas.

Tabela 2 – Atributos do *dataset Wine Recognition*

Nome	**Tipo**	**Descrição**	**Valores**
Alcohol	Numérico	Álcool	11,0 - 14,8
Malic Acid	Numérico	Ácido málico	0,74 - 5,80
Ash	Numérico	Cinza	1,36 - 3,23
Alcalinity of Ash	Numérico	Alcalinidade da Cinza	10,6 - 30,0
Magnesium	Numérico	Magnésio	70,0 - 162,0
Total Phenols	Numérico	Fenóis totais	0,98 - 3,88
Flavanoids	Numérico	Flavonóides	0,34 - 5,08
Nonflavanoid Phenols	Numérico	Fenóis não flavonóides	0,13 - 0,66
Proanthocyanins	Numérico	Proantocianinas	0,41 - 3,58
Colour Intensity	Numérico	intensidade da cor	1,3 - 13,0
Hue	Numérico	Matiz	0,48 - 1,71
OD280 / OD315	Numérico	DO280 / OD315	1,27 - 4,00
Proline	Numérico	Prolina	278 - 1680

Fonte: Elaborada pelo autor,

O *dataset Wisconsin Breast Cancer* é uma base de dados com 569 tuplas e bastante atributos quando comparamos com as demais. São 30 atributos que representam a média, o desvio padrão e o pior cenário para 10 elementos. Como essa base de dados possui atributos numéricos, assim como as demais, serão adotados todos os atributos para facilitar a busca de similaridades entre as tuplas.

Tabela 3 – Atributos do *dataset Breast Cancer*

Nome	**Tipo**	**Descrição**	**Valores**
Radius (mean)	Numérico	Média do raio	6,981 - 28,11
Texture (mean)	Numérico	Média da textura	9,71 - 39,28
Perimeter (mean)	Numérico	Média do perímetro	43,79 - 188,5
Area (mean)	Numérico	Média da área	143,5 - 2501,0
Smoothness (mean)	Numérico	Média da suavidade	0,053 - 0,163
Compactness (mean)	Numérico	Média da compacidade	0,019 - 0,345
Concavity (mean)	Numérico	Média da concavidade	0,0 - 0,427
Concave points (mean)	Numérico	Média dos pontos côncavos	0,0 - 0,201
Symmetry (mean)	Numérico	Média da simetria	0,106 - 0,304
Fractal dimension (mean)	Numérico	Média da dimensão fractal	0,05 - 0,097
Radius (standard error)	Numérico	Erro padrão do raio	0,112 - 2,873
Texture (standard error)	Numérico	Erro padrão da textura	0,36 - 4,885
Perimeter (standard error)	Numérico	erro padrão do perímetro	0,757 - 21,98
Area (standard error)	Numérico	Erro padrão da área	6,802 - 542,2
Smoothness (standard error)	Numérico	Erro padrão da suavidade	0,002 - 0,031
Compactness (standard error)	Numérico	Erro padrão da compacidade	0,002 - 0,135
Concavity (standard error)	Numérico	Erro padrão da concavidade	0,0 - 0,396
Concave points (standard error)	Numérico	Erro padrão dos pontos côncavos	0,0 - 0,053
Symmetry (standard error)	Numérico	Erro padrão da simetria	0,008 - 0,079
Fractal dimension (standard error)	Numérico	Erro padrão da dimensão fractal	0,001 - 0,03
Radius (worst)	Numérico	Pior raio	7,93 - 36,04

Texture (worst)	Numérico	Pior textura	12,02 - 49,54
Perimeter (worst)	Numérico	Pior perímetro	50,41 - 251,2
Area (worst)	Numérico	Pior área	185,2 - 4254,0
Smoothness (worst)	Numérico	Pior suavidade	0,071 - 0,223
Compactness (worst)	Numérico	Pior compacidade	0,027 - 1,058
Concavity (worst)	Numérico	Pior concavidade	0,0 - 1,252
Concave points (worst)	Numérico	Piores pontos côncavos	0,0 - 0,291
Symmetry (worst)	Numérico	Pior simetria	0,156 - 0,664
Fractal dimension (worst)	Numérico	Pior dimensão fractal	0,055 - 0,208

Fonte: Elaborada pelo autor.

Para importar esses *datasets* e convertê-los em *Dataframes*, foi utilizado o Algoritmo 1 no Google Colab utilizando a linguagem Python que foi adotada para todo o projeto.

Algorithm 1 Importação de *datasets* com conversão em *Dataframes*

```
#Dataset Iris Plant do Sklearn
import pandas as pd
from sklearn.datasets import load_iris
data = load_iris()
df_iris = pd.DataFrame(data=data.data, columns=data.
    feature_names)

#Dataset Wine Recognition do Sklearn
from sklearn.datasets import load_wine
data_wine = load_wine()
df_wine = pd.DataFrame(data=data_wine.data, columns=data_wine.
    feature_names)
df_alvo_wine = data_wine['target']
wine_completo=pd.DataFrame(data=data_wine.data, columns=
    data_wine.feature_names)

#Dataset Wisconsin Breast Cancer do Sklearn
from sklearn.datasets import load_breast_cancer
data_breast_cancer = load_breast_cancer()
df_breast_cancer = pd.DataFrame(data=data_breast_cancer.data,
    columns=data_breast_cancer.feature_names)
df_alvo_breast_cancer = data_breast_cancer['target']
breast_cancer_completo = pd.DataFrame(data=data_breast_cancer.
    data, columns=data_breast_cancer.feature_names)
```

3.2 Pré-processamento de dados

Como as bases não possuem muitos registros e seus atributos eram numéricos, não foi necessário o uso de recursos para pré-processar as bases de dados. Contudo, um pequeno código, apresentado no Algoritmo 2, muito utilizado para dimensionamento e padronização foi deixado pronto para uso preliminar apesar do seu uso nesse projeto não ter apresentado alterações significativas no resultado. Com o projeto mais desenvolvido e sua aplicação em bases diversas com atributos não numéricos, seria interessante o desenvolvimento de um algoritmo de seleção de atributos ou, então, um algoritmo para transformar dados não estruturados em estruturados. Acredito que com o refinamento do código, será necessário balancear dados de diferentes classes.

Algorithm 2 Pré-processamento dos dados

```
scaler = preprocessing.StandardScaler()

#Aplicado ao Iris Plant
X_iris=df_iris.iloc[:,:].values
scaler.fit(X_iris)
X_scaled_array = scaler.transform(X_iris)
X_iris = pd.DataFrame(X_scaled_array)

#Aplicado ao Wine Recognition
X_wine=df_wine.iloc[:,:].values
scaler.fit(X_wine)
X_scaled_array = scaler.transform(X_wine)
X_wine = pd.DataFrame(X_scaled_array)

#Aplicado ao Wisconsin Breast Cancer
X_breast_cancer=df_breast_cancer.iloc[:,:].values
scaler.fit(X_breast_cancer)
X_scaled_array = scaler.transform(X_breast_cancer)
X_breast_cancer = pd.DataFrame(X_scaled_array)
```

3.3 Métodos de clustering tradicionais

Como métodos tradicionais para fins de comparação foram escolhidos os algoritmos K-Means, Expectation Maximization (EM) viabilizado pelo Gaussian Mixture e DBSCAN. Todos esses algoritmos foram aplicados sobre o *dataset Iris Plant* e, posteriormente, sobre os *datasets Wine Recognition* e *Wisconsin Breast Cancer*.

Para o K-Means, fizemos o uso do Algoritmo 3. Observa-se que o código calcula a quantidade de *clusters* ótima que serve de entrada à função K-Means da bilioteca *scikit-learn*.

Algorithm 3 K-Means aplicado ao *dataset Iris Plant*

```
def Soma_Quadrados_Intra_Clusters(data):
    wcss = []
    for n in range(2, 6):
        kmeans = KMeans(n_clusters=n)
        kmeans.fit(X=data)
        wcss.append(kmeans.inertia_)
    return wcss

def Numero_Clusters_Otimizado(wcss):
    x1, y1 = 2, wcss[0]
    x2, y2 = 6, wcss[len(wcss)-1]
    distancias = []
    for i in range(len(wcss)):
        x0 = i+2
        y0 = wcss[i]
        numerador = abs((y2-y1)*x0 - (x2-x1)*y0 + x2*y1 - y2*x1)
        denominador = math.sqrt((y2 - y1)**2 + (x2 - x1)**2)
        distancias.append(numerador/denominador)
    return distancias.index(max(distancias)) + 2

X_iris=df_iris.iloc[:,:].values

# Soma dos quadrados para todas as quantidades de clusters
Soma_Quadrados = Soma_Quadrados_Intra_Clusters(X_iris)

# Calcula a quantidade otima de clusters
n_iris = Numero_Clusters_Otimizado(Soma_Quadrados)

# Roda o K-Means para a quantidade otima de clusters
kmeans = KMeans(n_clusters=n_iris)
kmeans_clusters_iris = kmeans.fit_predict(X_iris)

# Exibe clusters encontrados
print(kmeans_clusters_iris)

# Avaliando a qualidade dos Clusters
s = metrics.silhouette_score(X_iris, kmeans_clusters_iris,
    metric='euclidean')
print(f"\nCoeficiente de Silhueta para os Clusters da Base de
    Dados Iris Plant: {s:.2f}\n")

# Plota clusters em cima dos atributos mais visiveis
plt.figure(1, figsize=(12, 9))
plt.scatter(X_iris[:,3], X_iris[:,2],s=100, c=
    kmeans_clusters_iris, cmap='rainbow')
plt.show()
```

Ao rodarmos o código, obtemos a Figura 10 como saída para o *Iris Plant*. Com o

código alterado para os *datasets Wine Recognition* e *Wisconsin Breast Cancer* obtemos a Figura 11 e a Figura 12, respectivamente.

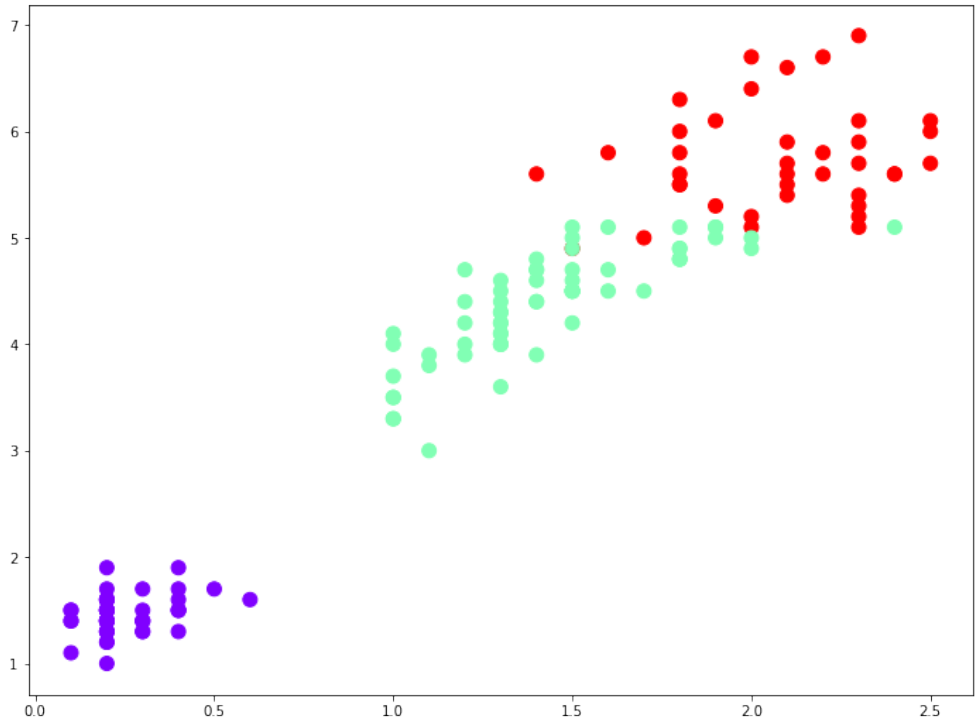

Figura 10 – Resultado do K-Means para o *dataset Iris Plant*

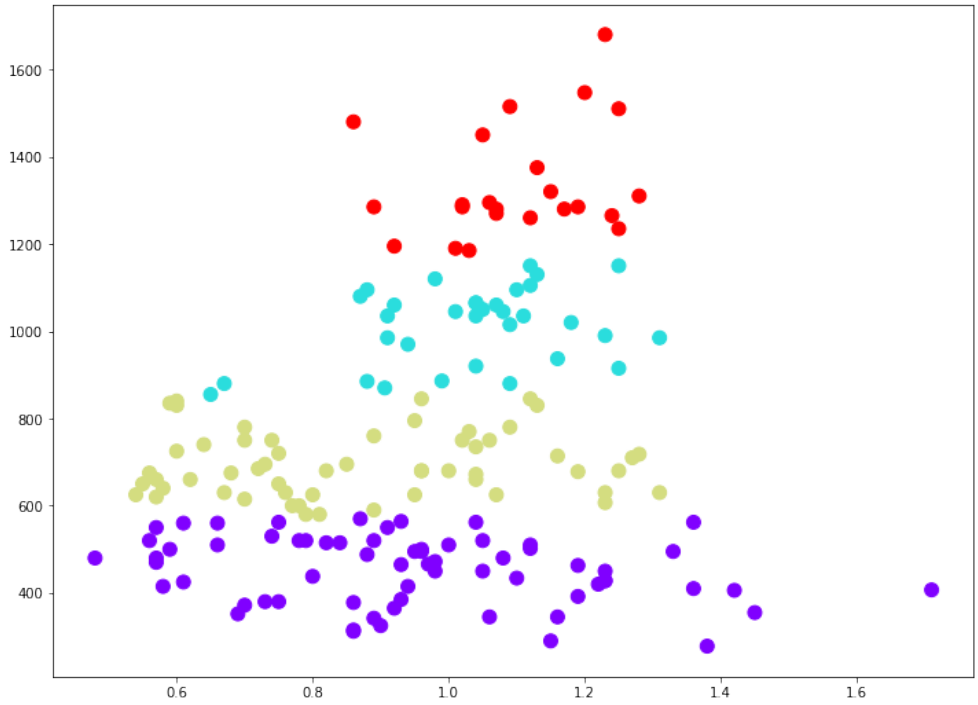

Figura 11 – Resultado do K-Means para o *dataset Wine Recognition*

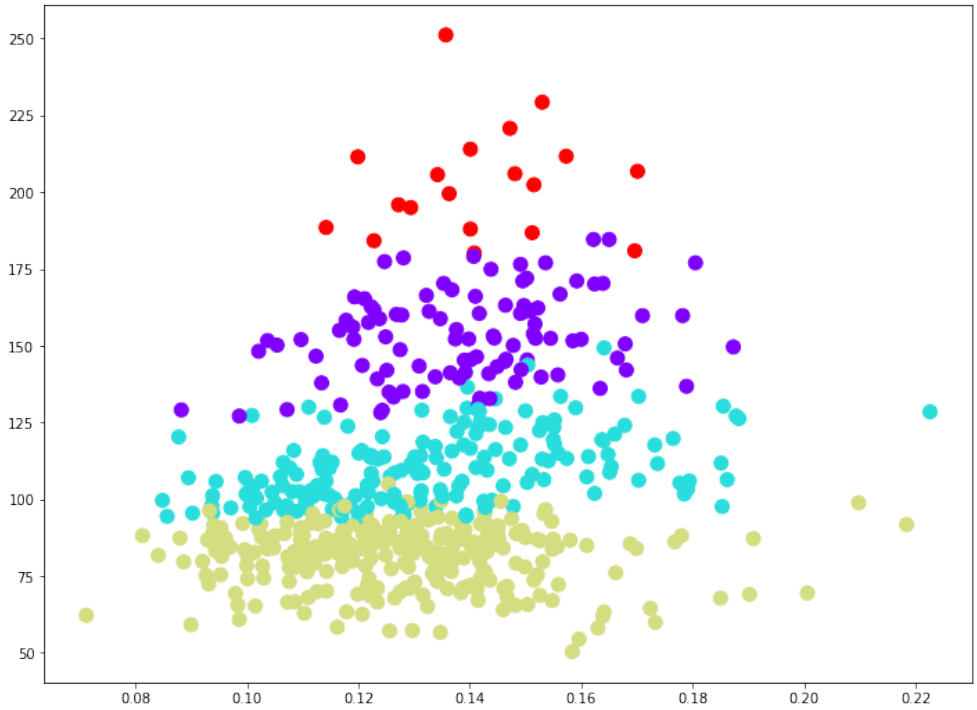

Figura 12 – Resultado do K-Means para o *dataset Wisconsin Breast Cancer*

Apesar de, visualmente, o resultado do Algoritmo 3 parecer bom para o *dataset Iris Plant*, aparentemente não dá para concluirmos somente olhando para as imagens que os resultados para os *datasets Wine Recognition* e *Wisconsin Breast Cancer* foram bons dado que encontram 4 *clusters* cada, em vez de 3 e 2, respectivamente. No entanto, o coeficiente de silhueta ficou em 0,55 para o *Iris Plant*, 0,56 para o *Wine Recognition* e 0,53 para o *Wisconsin Breast Cancer*.

Para o DBSCAN, fizemos o uso de um código mais simples sem buscar valores ótimos de raio máximo em que dois pontos podem ser considerados do mesmo *cluster*, conhecido com Epsilon (EPS), e número mínimo de pontos em uma região necessários para garantir uma densidade desejada (MinPts). No entanto, o código não rodou para os *datasets Wine Recognition* e *Wisconsin Breast Cancer*. Portanto, foi elaborado o Algoritmo 4 que buscava o maior EPS baseado nos retornos dos melhores valores de Normalized mutual Information (NMI). Esse maior EPS calculado, foi aproveitado no cálculo das matrizes de adjacência para a formação das redes complexas dado que a ideia, como será mostrada mais a frente, é ter uma distância entre cada vértice que se baseie não apenas no kNN, mas também em um raio que traga um resultado otimizado.

Algorithm 4 DBSCAN aplicado ao *dataset Iris Plant*

```
#Buscando um EPS que maximize o NMI
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(n_neighbors=3)
nbrs = neigh.fit(X_iris)
distances, indices = nbrs.kneighbors(X_iris)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
from sklearn.metrics.cluster import normalized_mutual_info_score
maior_eps_iris=0
maior_nmi_iris=0
maior_samples_iris=0
for i in enumerate(distances):
  for j in range(1,20):
    if i[1]>0:
      db_iris = DBSCAN(algorithm='auto',eps=i[1], min_samples=j,
    metric='euclidean')
      db_iris.fit(X_iris)
      db_clusters_iris = db_iris.fit_predict(X_iris)
      if normalized_mutual_info_score(df_alvo_iris,
   db_clusters_iris)>maior_nmi_iris:
        maior_nmi_iris=normalized_mutual_info_score(df_alvo_iris
   ,db_clusters_iris)
        maior_eps_iris=i[1]
        maior_samples_iris=j

db_iris = DBSCAN(algorithm='auto',eps=maior_eps_iris,
   min_samples=maior_samples_iris, metric='euclidean')
db_iris.fit(X_iris)
db_clusters_iris = db_iris.fit_predict(X_iris)

# Avaliando a qualidade dos Clusters
s = metrics.silhouette_score(X_iris, db_clusters_iris, metric='
   euclidean')
print(f"\nCoeficiente de Silhueta para os Clusters da Base de
   Dados iris: {s:.2f}\n")

#Plota clusters em cima dos atributos mais vis veis
plt.figure(1, figsize=(12, 9))
plt.scatter(X_iris[:,3], X_iris[:,2],s=100, c=db_clusters_iris,
   cmap='rainbow')
plt.show()
```

Obtemos, portanto, a Figura 13 como saída. No caso dos *datasets Wine Recognition* e *Wisconsin Breast Cancer*, foi adaptado o algoritmo, como nos demais casos, para cada uma dessas bases de modo que obtemos a Figura 14 e a Figura 15, respectivamente, ao rodá-los. O coeficiente de silhueta ficou em 0,51 para o *Iris Plant*, 0,54 para o Wine Recognition e 0,57 para o *Wisconsin Breast Cancer*. Nesse caso, o coeficiente de silhueta

aumenta com a complexidade do *dataset* e podemos observar que, enquanto o resultado não parece interessante para o *dataset Iris Plant*, no caso do do *dataset Wisconsin Breast Cancer* o resultado tem uma melhora significativa.

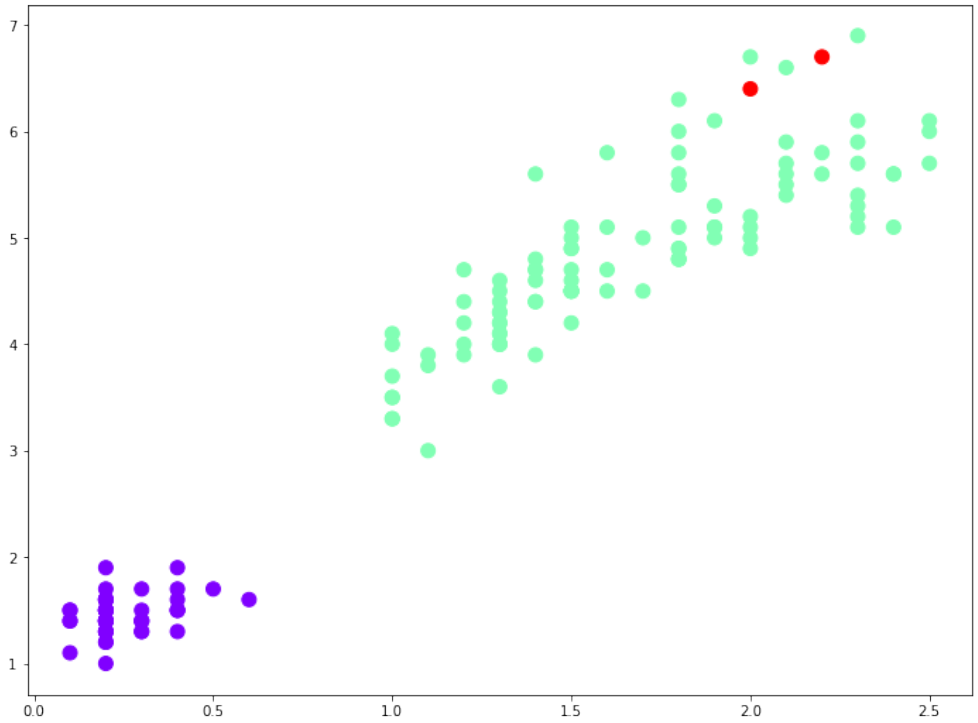

Figura 13 – Resultado do DBSCAN para o *dataset Iris Plant*

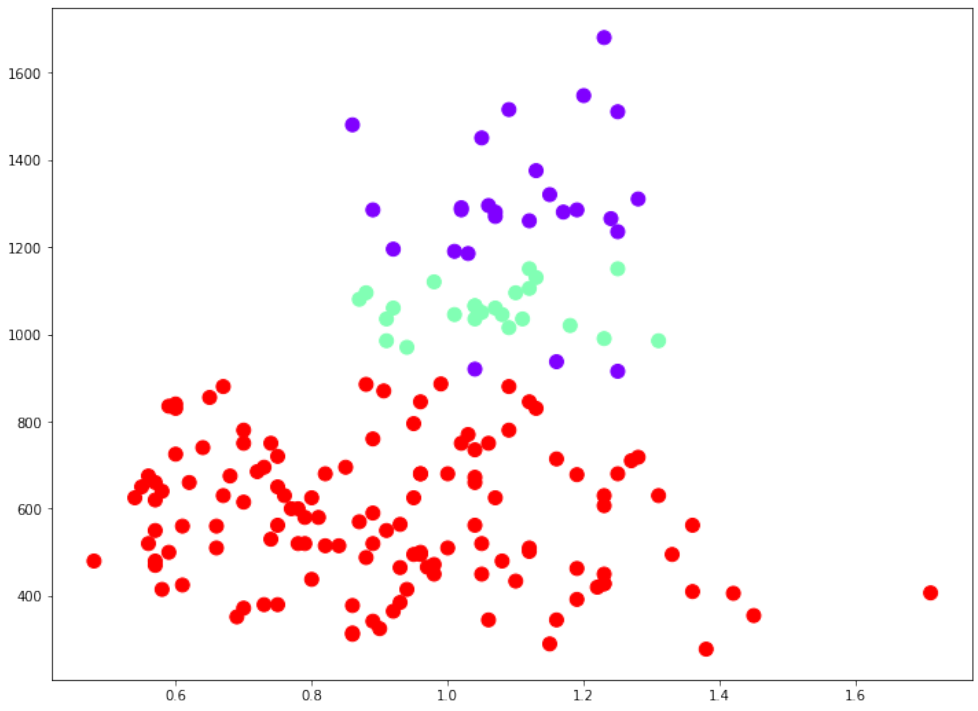

Figura 14 – Resultado do DBSCAN para o *dataset Wine Recognition*

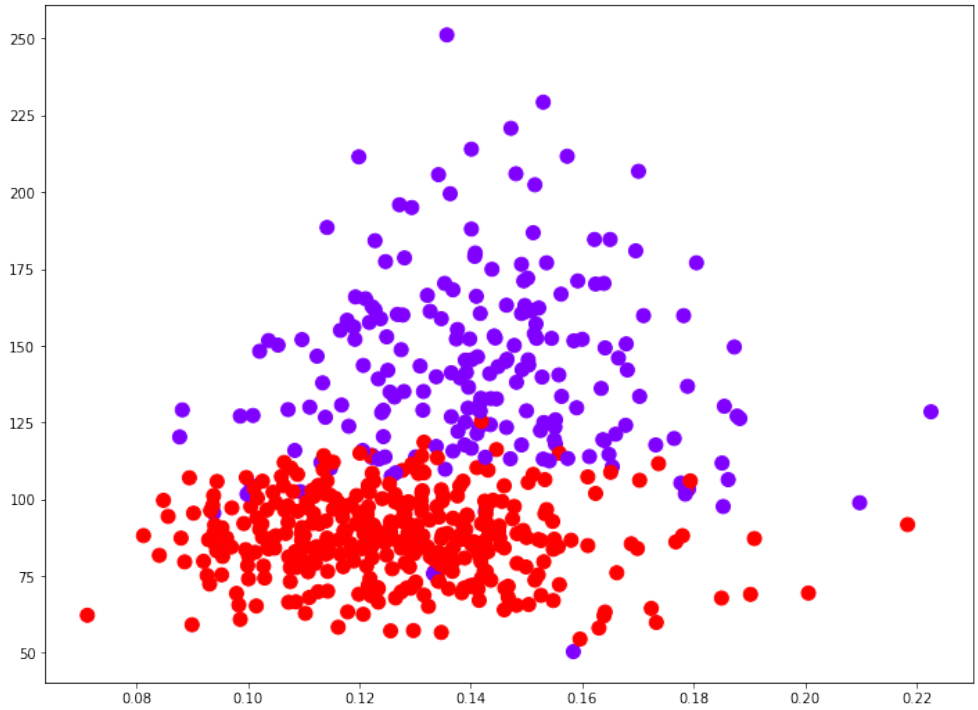

Figura 15 – Resultado do DBSCAN para o *dataset Wisconsin Breast Cancer*

Ainda com os métodos tradicionais, foi utilizado o Gaussian Mixture, um modelo probabilístico que assume que todos os pontos de dados são gerados a partir de uma mistura de um número finito de distribuições gaussianas com parâmetros desconhecidos, para implementar o Expectation Maximization (EM), uma abordagem para realizar estimativas de máxima verossimilhança na presença de variáveis latentes . Ele faz isso primeiro estimando os valores para as variáveis latentes, depois otimizando o modelo e repetindo essas duas etapas até a convergência.

Do mesmo modo, que os métodos anteriores, foi buscado um valor ótimo que maximizasse o valor do Normalized mutual Information (NMI). Para tanto, foi desenvolvido o Algoritmo 5.

Algorithm 5 Gaussian Mixture aplicado ao *dataset Iris Plant*

```
#Buscando um n_components que maximize o GaussianMixture
from sklearn.metrics.cluster import normalized_mutual_info_score
from sklearn.mixture import GaussianMixture
maior_n_components=0
maior_nmi=0
for i in range(1,20):
  gm_iris = GaussianMixture(n_components=i, init_params='kmeans')
  gm_iris.fit(X_iris)
  gm_clusters_iris = gm_iris.fit_predict(X_iris)
  if normalized_mutual_info_score(df_alvo_iris,gm_clusters_iris)>maior_nmi:
    maior_nmi=normalized_mutual_info_score(df_alvo_iris,gm_clusters_iris)
    maior_n_components=i

from sklearn.mixture import GaussianMixture
gm = GaussianMixture(n_components=maior_n_components, init_params='kmeans')
gm_clusters_iris = gm.fit_predict(X_iris)

# Exibe clusters encontrados
print(gm_clusters_iris)

# Avaliando a qualidade dos Clusters
s = metrics.silhouette_score(X_iris, gm_clusters_iris, metric='euclidean')
print(f"\nCoeficiente de Silhueta para os Clusters da Base de Dados iris Plant: {s:.2f}\n")

# Plota clusters em cima dos atributos mais visiveis
plt.figure(1, figsize=(12, 9))
plt.scatter(X_iris[:,2], X_iris[:,3],s=100, c=gm_clusters_iris, cmap='rainbow')
plt.show()
```

Os resultados para o Expectation Maximization (EM) por meio da função Gaussian Mixture estão plotados na Figura 16, Figura 17 e Figura 18 que representam respectivamente os *datasets Iris Plant, Wine Recognition* e *Wisconsin Breast Cancer*. O coeficiente de silhueta ficou em 0,50 para o *Iris Plant*, 0,56 para o *Wine Recognition* e 0,53 para o *Wisconsin Breast Cancer*, lembrando os coeficientes do método K-Means.

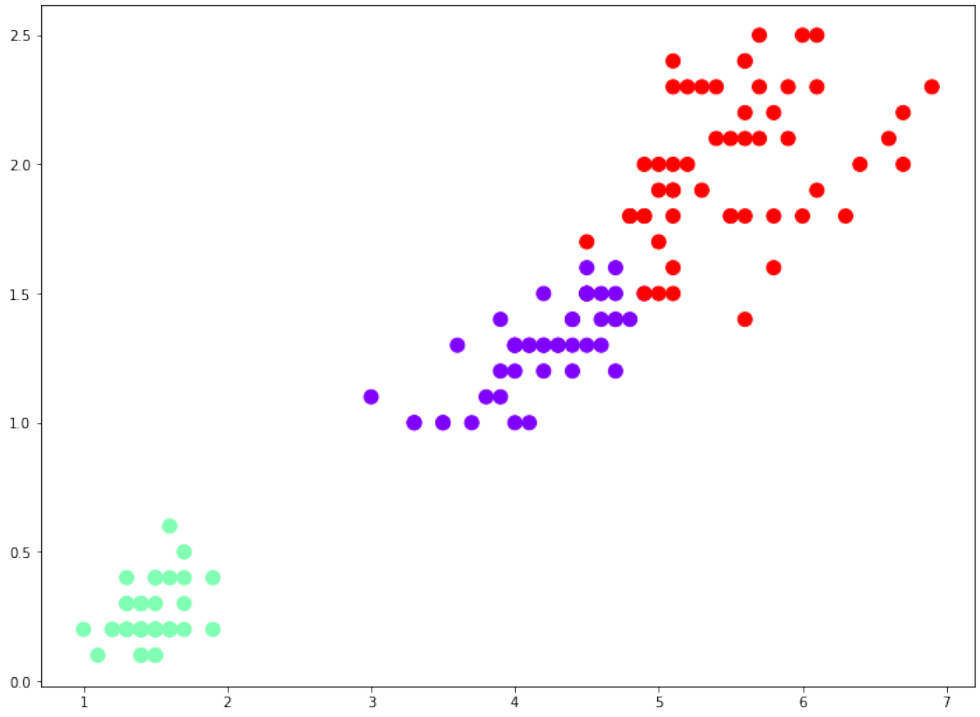

Figura 16 – Resultado do EM para o *dataset Iris Plant*

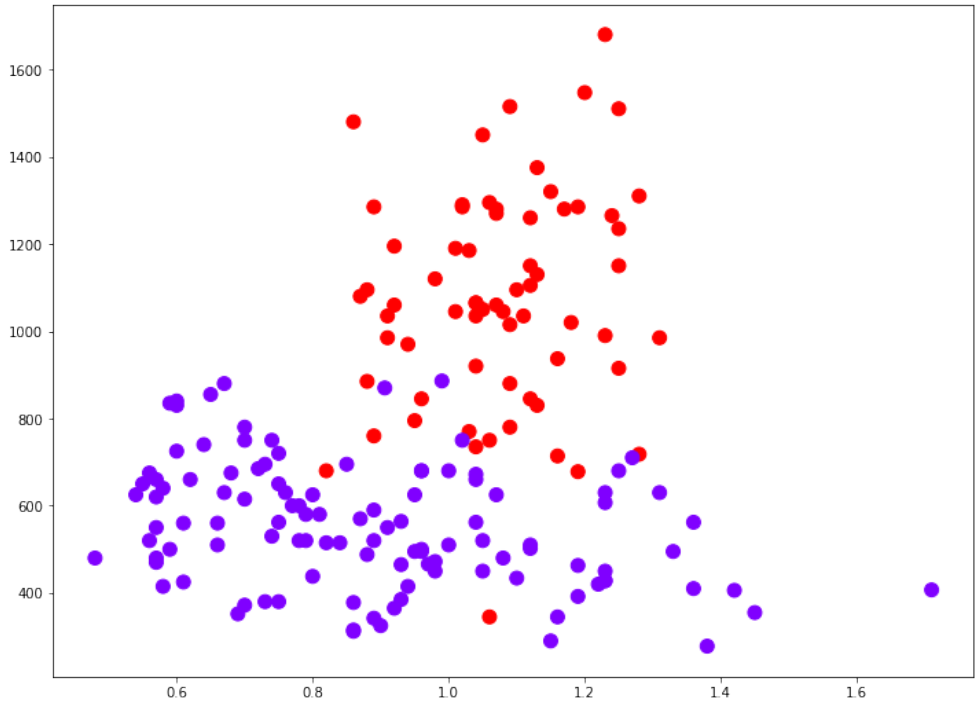

Figura 17 – Resultado do EM para o *dataset Wine Recognition*

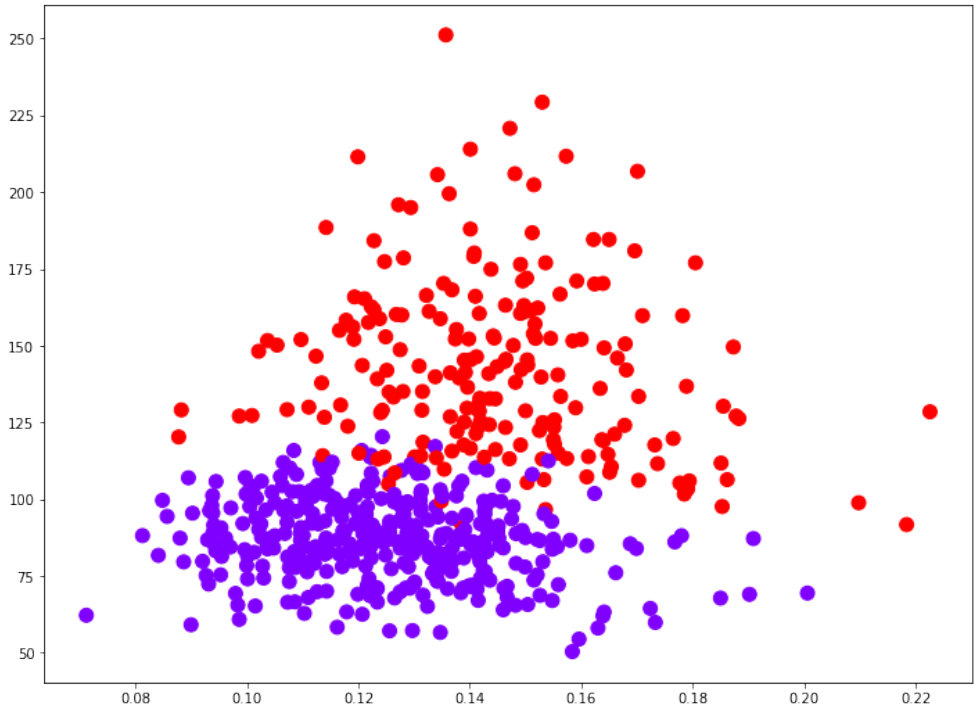

Figura 18 – Resultado do EM para o *dataset Wisconsin Breast Cancer*

3.4 Método baseado em redes complexas

Para o desenvolvimento do trabalho foi utilizada a técnica descrita em (COMIN *et al.*, 2020) que segue os passos da Figura 19.

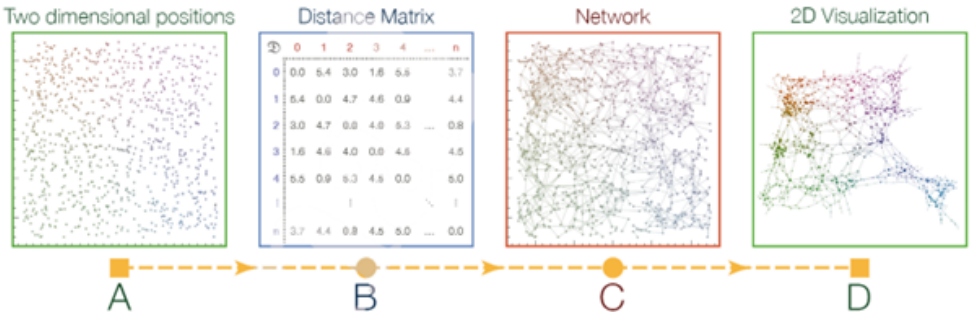

Figura 19 – Caminho para a transformação de dados em rede complexa, reproduzido de (COMIN *et al.*, 2020)

Optou-se nesse trabalho em transformar os *datasets* em uma matriz de similaridade para que fosse possível formar as redes. Essas matrizes eram transformadas em matrizes de adjacência quando um algoritmo buscava os pontos mais próximos de cada vértice a partir de um raio e da execução do kNN.

O passo seguinte era fazer uma limiarização da rede, buscando o *backbone* global da rede construída. A ideia era aproveitar a própria matriz de similaridades, que possuía

a distância euclidiana entre cada vértice, e retirar um percentual dos menores valores e ir comparando a implicação desse procedimento sobre o Normalized Mutual Information (NMI) que foi escolhido como principal forma de comparação nesse projeto. No entanto, o desenvolvimento do código demonstrou que o algoritmo trouxe um custo computacional alto ao dataset Wisconsin Breast Cancer, que possui 569 instâncias, enquanto não melhorou o resultado para os *datasets* menores de forma a se justificar seu uso.

Para ir do passo A ao B, foi necessário escolher uma forma de medir as distâncias entre cada instância da base aplicando uma das distâncias da família Minkwoski. Como em (ARRUDA; COSTA; RODRIGUES, 2018) já haviam detectado que a distância euclidiana trazia melhores resultados, optou-se pelo uso dela e, para tanto, foi construído o Algoritmo 6 para auxiliar a encontrar a distância euclidiana entre duas instâncias dado seus atributos.

Algorithm 6 Similaridade usando a distância euclidiana

```
def Similaridade(data_1,data_2,data_len,alpha):
  distancia=0
  for i in range(data_len):
    distancia=distancia+np.square(data_1[i]-data_2[i])
    if distancia==0:
      similaridade=alpha*np.e**(-alpha*np.sqrt(distancia))
    else:
      similaridade=1/np.sqrt(distancia)
  return similaridade
```

Dada a função apresentada no Algoritmo 6, foi, assim, construída as matrizes de similaridades sobre as instâncias dos *datasets*, conforme o Algoritmo 7.

Algorithm 7 Construção da matriz de similaridade para o *dataset Iris Plant*

```
#Construindo matriz de similaridade
matriz_similaridade_iris = np.zeros((len(df_iris), len(df_iris))
    )
for i in range(len(df_iris)):
  for j in range(len(df_iris)):
    matriz_similaridade_iris[i,j]=Similaridade(df_iris.iloc[i],
   df_iris.iloc[j], df_iris.shape[1],100)
```

Para formar a rede a partir das distâncias/similaridades, foi realizada uma combinação de estratégias, onde conecta-se os pontos dentro de um raio especificado e, após, caso o raio contenha menos de k elementos estabelecidos, completa-se com os vizinhos mais próximos conforme o Algoritmo 8.

Algorithm 8 Função para formar as redes complexas a partir dos *datasets*

```
#Funcao para forma  o da rede
def Raio_KNN(input_list, raio, k, corte):
    x = np.zeros((len(input_list), len(input_list)))
    y = np.zeros((len(input_list), len(input_list)))
    for i in range(len(input_list)):
      for j in range(len(input_list)):
        x[i,j]=input_list[i,j]
        y[i,j]=input_list[i,j]
    for i in range(len(x)):
      for j in range(len(x)):
        if ((x[i,j]>=1/raio) and (x[i,j]<=corte)):
          x[i,j] = 1
        else:
          x[i,j] = 0
    posicao = np.ones((len(input_list), len(input_list)))
    for i in range(len(y)):
      for j in range(len(y)):
        for w in range(len(y)):
          if y[i,j]>y[i,w]:
            posicao[i,j] = posicao[i,j] + 1
    for i in range(len(posicao)):
      for j in range(len(posicao)):
          if (posicao[i,j] >= (len(posicao)-k))and (posicao[i,j]
    != (len(posicao))):
            posicao[i,j] = 1
          else:
             posicao[i,j] = 0
    for i in range(len(posicao)):
      for j in range(len(posicao)):
          if posicao[i,j] != x[i,j]:
            posicao[i,j] = posicao[i,j] + x[i,j]
    return posicao
```

Ato contínuo, foi desenvolvida uma matriz de adjacência para a formação da rede complexa sobre o *dataset Iris Plant* com o uso do Algoritmo 9. Nesse mesmo algoritmo é realizado a construção do grafo a partir dessa matriz de adjacência e, logo após, é plotado o grafo seguindo o Kamada Kawai Layout para que o grafo não fosse plotado de modo aleatório cada vez que fosse rodado esse algoritmo.

Algorithm 9 Formação da rede para *dataset Iris Plant*

```
#Formando a rede
import networkx as nx
import matplotlib.pyplot as plt
matriz_adjacencia_iris = Raio_KNN(matriz_similaridade_iris,
   maior_eps_iris, 9, 80)
graph_iris = nx.Graph()
for i in range(len(matriz_adjacencia_iris)):
  for j in range(len(matriz_adjacencia_iris)):
    if (matriz_adjacencia_iris[i,j]==1) and not(graph_iris.
   add_edge(j,i)):
      graph_iris.add_edge(i,j)
# Plotando a rede
#definindo o tamanho da figurap
lt.figure(1, figsize=(12, 9))
#definindo o algoritmo do layout
pos=nx. kamada_kawai_layout(graph_iris)
plt.axis('off')
#plota os nodes
nx.draw_networkx_nodes(graph_iris, pos, node_size=100)
#plota as arestas
nx.draw_networkx_edges(graph_iris, pos, alpha=0.4)
plt.show()
```

Como resultado do algoritmo anterior, foi plotada a rede da Figura 20 para o *dataset Iris Plant*.

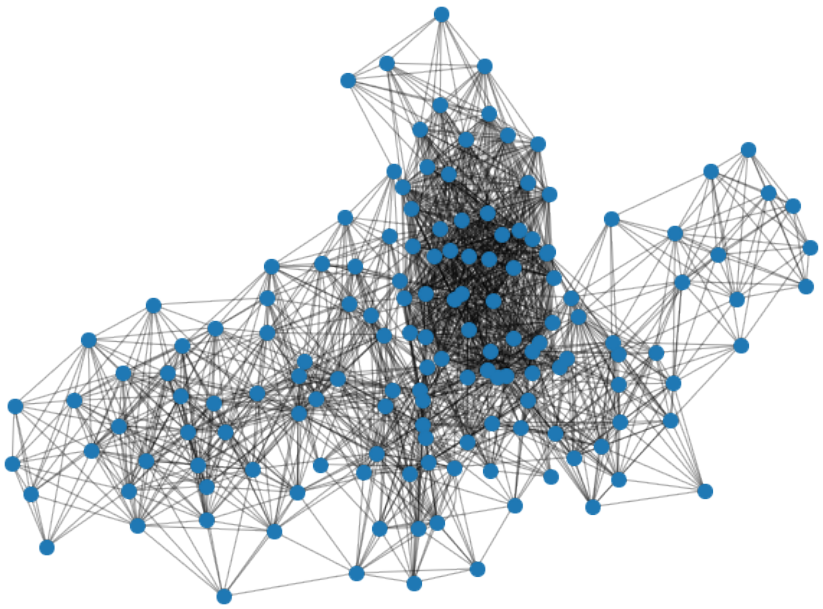

Figura 20 – Rede formada para o *dataset Iris Plant*

A Figura 21 e a Figura 22 são resultados do mesmo algoritmo empregado para os *datasets Wine Recognition* e *Wisconsin Breast Cancer*.

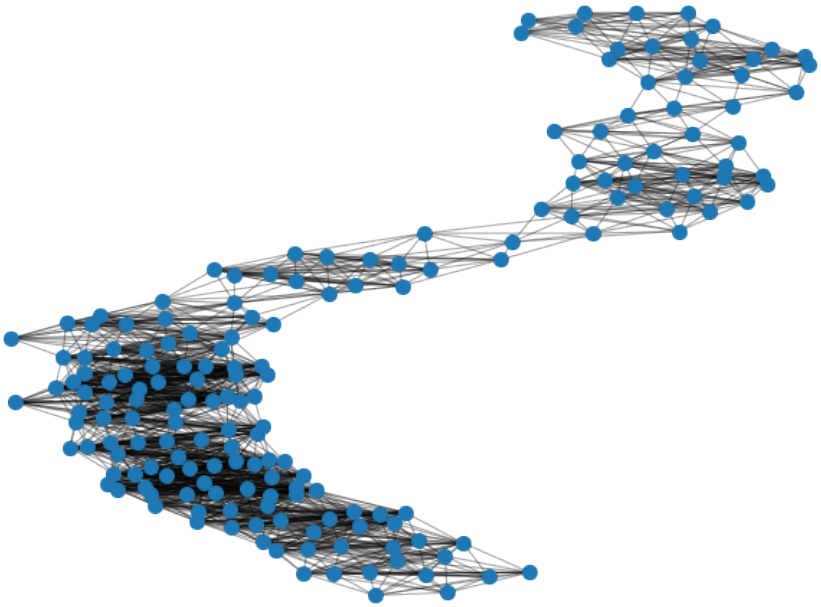

Figura 21 – Rede formada para o *dataset Wine Recognition*

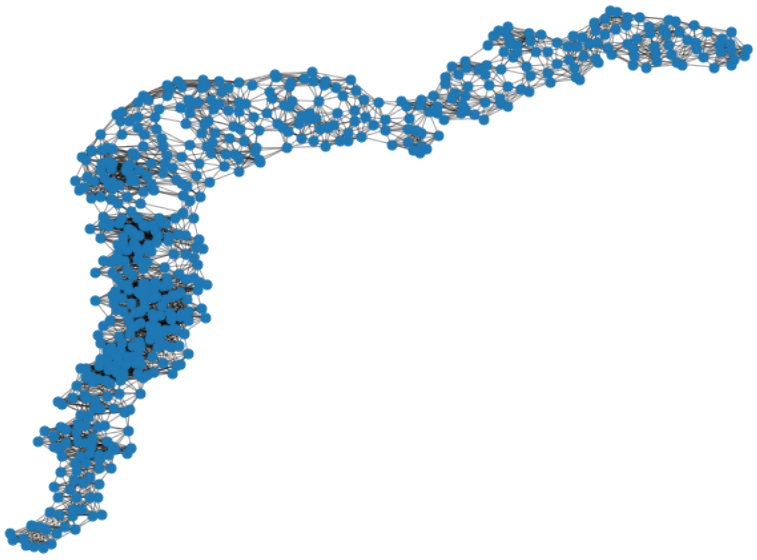

Figura 22 – Rede formada para o *dataset Wisconsin Breast Cancer*

3.4.1 Detecção de comunidades

Alguns métodos de detecção de comunidade, apresentados na disciplina de redes complexas do MBA em Inteligência Artificial e Big Data do ICMC, foram testados nesse trabalho. O primeiro foi o Girvan Newman que é um método que retorna um iterador com as comunidades encontradas a cada iteração do algoritmo, conforme o Algoritmo 10.

Algorithm 10 Método Girvan Newman

```
partition_GN_iris = None
for lvl, comms in enumerate(nx.algorithms.community.centrality.
    girvan_newman(graph_iris)):
     if lvl == 1:
         partition_GN_iris = comms
         colors_iris = dict()
         for i, comm in enumerate(comms):
             for vtx in comm:
                 colors_iris[vtx] = i

# vamos colocar o grafo conforme as comunidades indentificadas
    no lvl = 1
node_color_iris = []
for vtx in graph_iris.nodes():
     node_color_iris.append(colors_iris[vtx])
# Plotando a rede
#definindo o tamanho da figura
plt.figure(1, figsize=(12, 9))
#definindo o algoritmo do layout
pos=nx. kamada_kawai_layout(graph_iris)
plt.axis('off')
#plota os nodes
nx.draw_networkx_nodes(graph_iris, pos,node_color=
    node_color_iris, node_size=100)
nx.draw_networkx_edges(graph_iris, pos, alpha=0.4)
plt.show()
```

Ao executar o código anterior, obtêm-se a Figura 23, que identifica 4 comunidades para o *Iris Plant* que só possui apenas 3 tipos possíveis. A Figura 24 e a Figura 25 são os resultados do algoritmo para os *datasets Wine Recognition* e *Wisconsin Breast Cancer*, respectivamente.

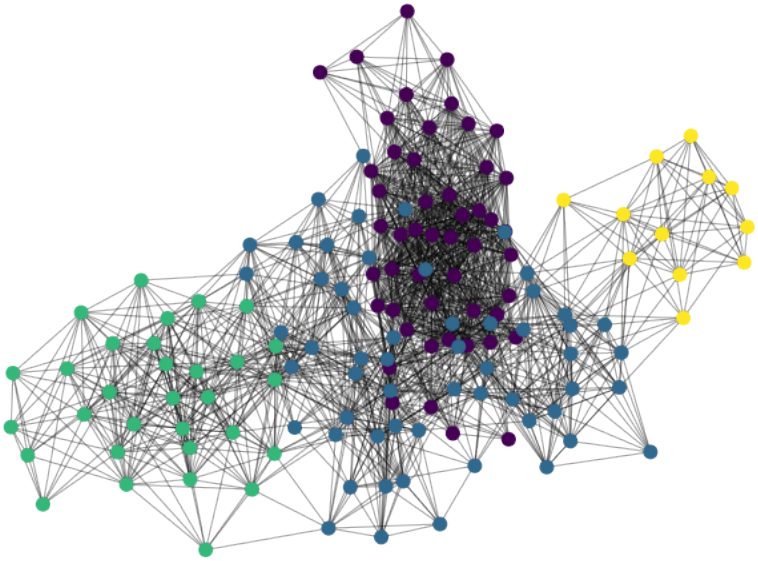

Figura 23 – Girvan Newman para o *dataset Iris Plant*

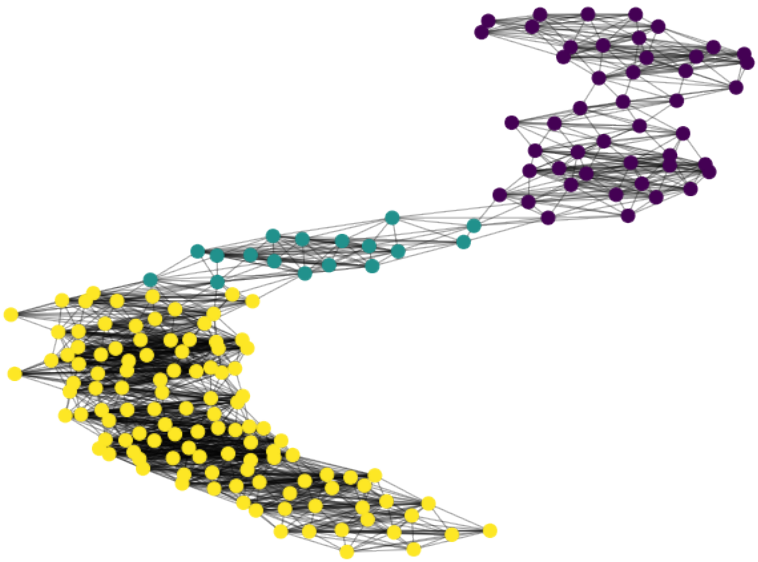

Figura 24 – Girvan Newman para o *dataset Wine Recognition*

Figura 25 – Girvan Newman para o *dataset Wisconsin Breast Cancer*

Se calcularmos a aresta mais central, aresta essa conhecida como *Most Valuable Edge*, com o acréscimo do Algoritmo 11 ao algoritmo 10 obteremos a Figura 26 quando aplicado ao *dataset Iris Plant*.

Algorithm 11 Método Girvan Newman com cálculo da aresta mais central

```
1 from networkx import edge_betweenness_centrality
2 from random import random
3 def aresta_mais_central(G):
4     centralidade = edge_betweenness_centrality(G)
5     max_cent = max(centralidade.values())
6     centralidade = {e: c / max_cent for e, c in centralidade.
    items()}
7     centralidade = {e: c + random() for e, c in centralidade.
    items()}
8     return max(centralidade, key=centralidade.get)
```

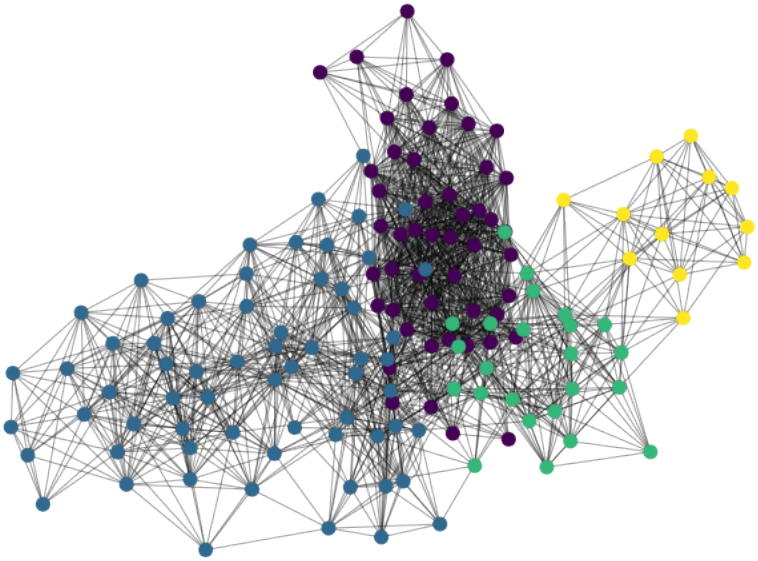

Figura 26 – Girvan Newman com cálculo de aresta mais central para o *dataset Iris Plant*

Se calcularmos a partir da aresta mais central, o método Girvan Newman retornará a Figura 27 e a Figura 28 para os *datasets Wine Recognition* e *Wisconsin Breast Cancer*, respectivamente, a partir da combinação do algoritmos 10 e 11.

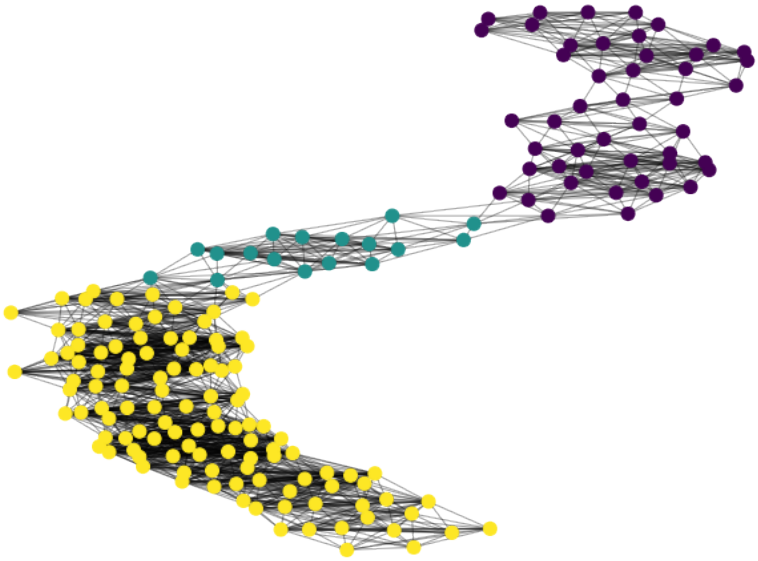

Figura 27 – Girvan Newman com cálculo de aresta mais central para o *dataset Wine Recognition*

Figura 28 – Girvan Newman com cálculo de aresta mais central para o *dataset Wisconsin Breast Cancer*

Combinando o algoritmo de Girvan Newman com a medida de modularidade, podemos indicar qual é a melhor partição encontrada e assim teremos o Greedy Modularity Optimization. Ao contrário do método anterior que retorna um iterador, aqui já recebemos a partição ótima, conforme o algoritmo 12

Algorithm 12 Método greedy modularity optimization

```
partition = nx.algorithms.community.modularity_max.
   greedy_modularity_communities(graph_iris)
def idx_partition(partition, node):
    for i, p in enumerate(partition):
        if node in p:
            return i
    return None
partition_GMO_iris=[]
for i,j in enumerate(partition):
  partition_GMO_iris.append(set(partition[i]))
partition_GMO_iris=tuple(partition_GMO_iris)
```

Ao executarmos o código anterior sobre o *dataset Iris Plant*, obtêm-se a Figura 29. Obteremos a Figura 30 para o *dataset Wine Recognition* e a Figura 31 para o *dataset Wisconsin Breast Cancer*.

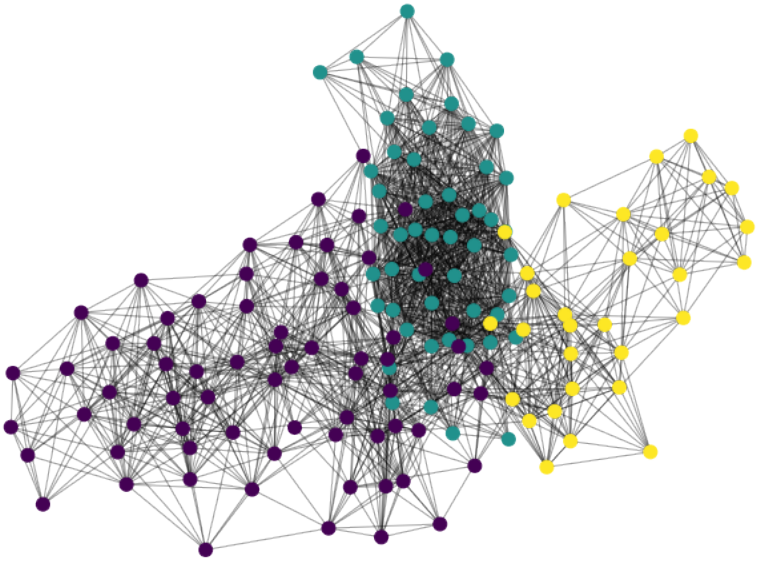

Figura 29 – Greedy Modularity Optimization para o *dataset Iris Plant*

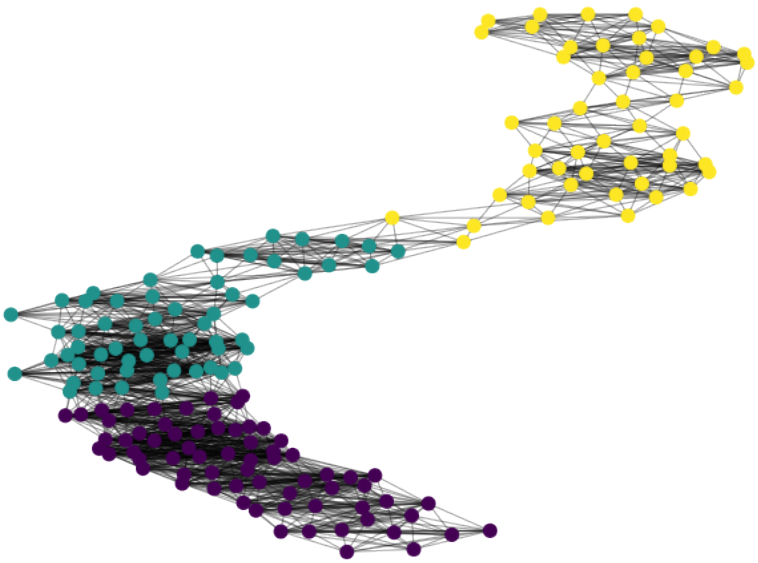

Figura 30 – Greedy Modularity Optimization para o *dataset Wine Recognition*

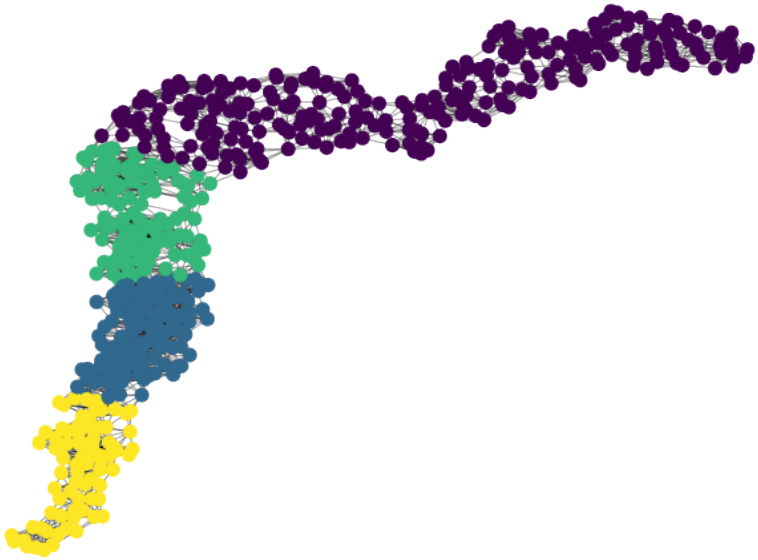

Figura 31 – Greedy Modularity Optimization para o *dataset Wisconsin Breast Cancer*

Como a versão estável do Networkx não possui uma implementação do método Louvain, foi necessário utilizar a biblioteca python-louvain por ela ser compatível com Networkx. O algoritmo 13 implementa o método Louvain de detecção de comunidades.

Algorithm 13 Método Louvain

```
import community.community_louvain as community_louvain
def transformar_conjunto(dicionario):
  dict_response = []
  false_dict_sort = list(dicionario.items())
  false_dict_sort.sort(key=lambda e: e[1])
  def getDuplicateValue(index, false_dict):
    if index + 1 < len(false_dict) and false_dict[index][1] ==
   false_dict[index + 1][1]:
      return [false_dict[index][0]] + getDuplicateValue(index +
   1, false_dict)
    else:
      return [false_dict[index][0]]
  i = 0
  while i < len(false_dict_sort):
    keys_list = set(getDuplicateValue(i, false_dict_sort))
   dict_response.append((keys_list))#, false_dict_sort[i][1]))
   lista de chaves (keys_list)
    i += len(keys_list)
  return tuple(dict_response)
partition = community_louvain.best_partition(graph_iris)
node_color_iris = list(partition.values())
partition_L_iris=tuple(transformar_conjunto(partition))
```

Ao executarmos o código anterior, obtemos a Figura 32 para o *dataset Iris Plant*. O código leva a Figura 33 para o *dataset Wine Recognition*. E, no caso do *dataset Wisconsin Breast Cancer*, teremos a Figura 34.

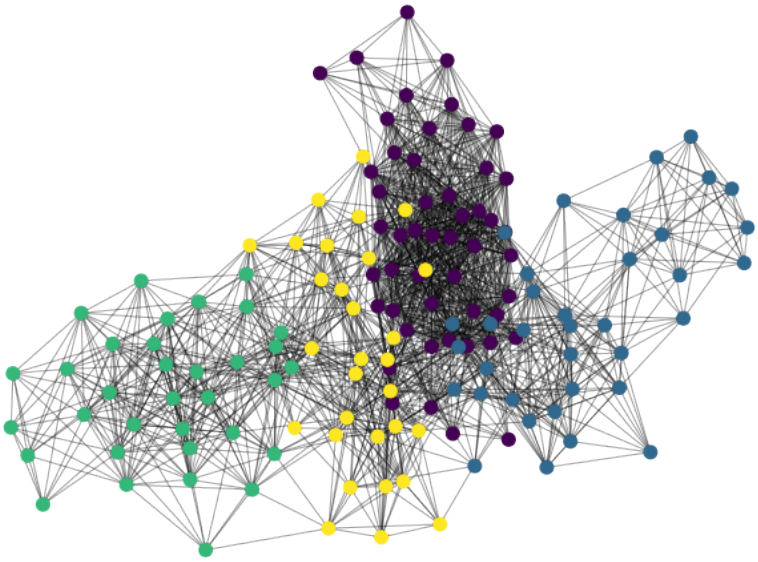

Figura 32 – Louvain para o *dataset Iris Plant*

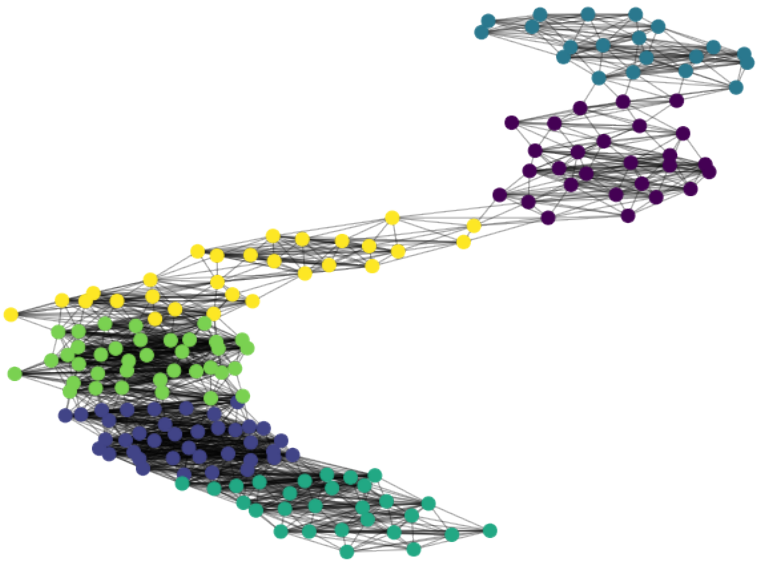

Figura 33 – Louvain para o *dataset Wine Recognition*

Figura 34 – Louvain para o *dataset Wisconsin Breast Cancer*

O último método de detecção de comunidades em redes complexas testado foi o Label Propagation. O algoritmo 14 implementa esse método no trabalho.

Algorithm 14 Método Label Propagation

```
partition_iter = nx.community.label_propagation.
    asyn_lpa_communities(graph_iris)
partition_LP_iris = []
for comm in partition_iter:
    partition_LP_iris.append(comm)
partition_LP_iris=tuple(partition_LP_iris)
node_color_iris = []
for node in graph_iris.nodes():
    node_color_iris.append(idx_partition(partition_LP_iris, node
    ))
```

Com a execução do código anterior, obtemos a Figura 35 para o *dataset Iris Plant*, da Figura 36 para o *dataset Wine Recognition* e da Figura 37 para o *dataset Wisconsin Breast Cancer*.

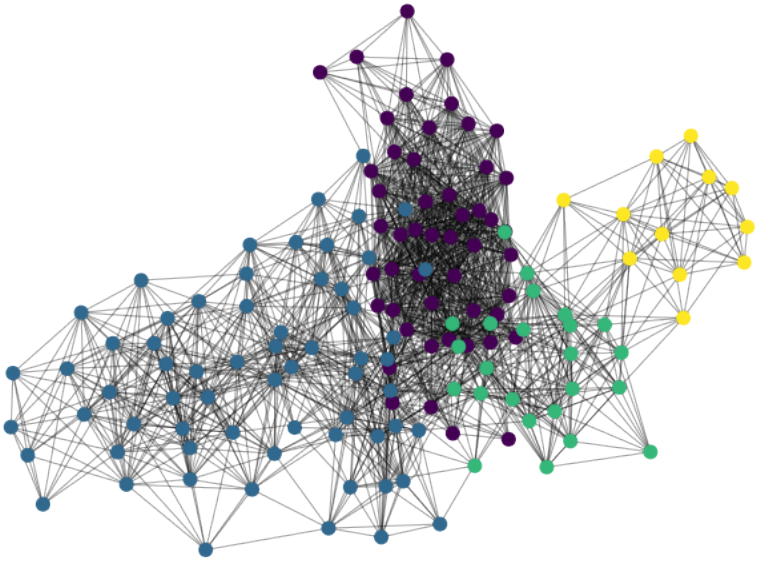

Figura 35 – Label Propagation para o *dataset Iris Plant*

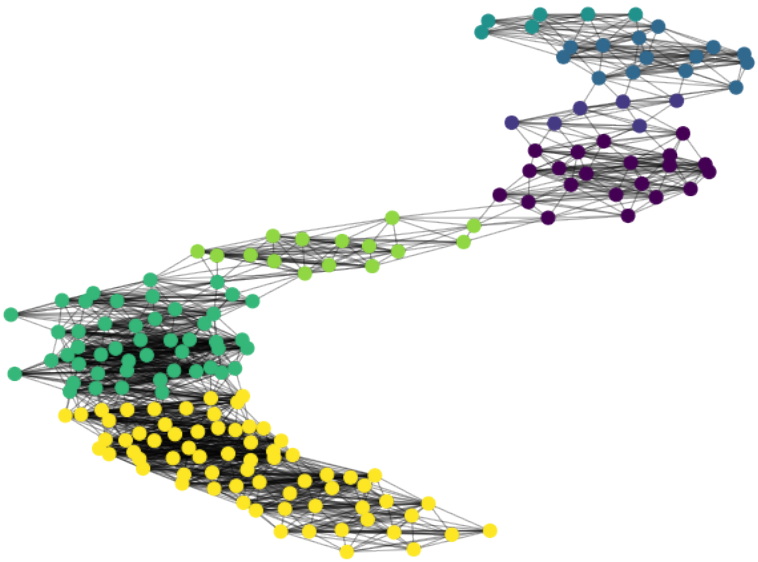

Figura 36 – Label Propagation para o *dataset Wine Recognition*

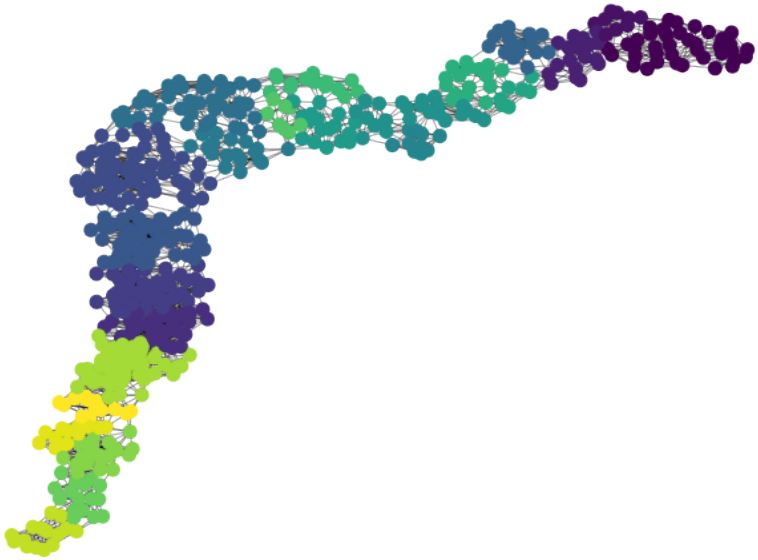

Figura 37 – Label Propagation para o *dataset Wisconsin Breast Cancer*

4 RESULTADOS

Nesse capítulo são apresentados os resultados da metodologia aplicada sobre os *datasets*. Foram escolhidas como medidas de comparação dos resultados a Normalized Mutual Information, o Adjusted Mutual Information e o Adjusted Rand Score. Os dados para os *datasets Iris Plant, Wine Recognition* e *Wisconsin Breast Cancer* são apresentados respectivamente na Tabela 4, Tabela 5 e Tabela 6.

Como pode-se observar, o método Expectation Maximization obteve o melhor desempenho sobre todas os *datasets*. No caso do *dataset Iris Plant*, os métodos Greedy Modularity Optimization, K-Means, Louvain, Girvan Newman (com Aresta Mais Central) e Label Propagation tiveram, nessa ordem, os melhores resultados após o EM. Todos com resultados muito próximos. Os dois piores resultados foram Girvan Newman e o DBSCAN. Contudo, foram resultados próximos dos demais métodos.

Para o *dataset Wine Recognition*, todos os resultados foram inferiores aos obtidos no *dataset Iris Plant*. Do melhor para o pior a ordem foi Expectation Maximization, Girvan Newman, Girvan Newman (Aresta Mais Central), Greedy Modularity Optimization, DBSCAN, Kmeans, Louvain e Label Propagation. No entanto, somente se destacou o resultado para o método EM enquanto os demais ficaram em patamares semelhantes.

Por fim, no *dataset Wisconsin Breast Cancer*, obtivemos resultados praticamente idênticos ao do *dataset Wine Recognition*, com o DBSCAN se destacando na segunda posição e com o Louvain e o Label Propagation com resultados muito abaixo dos demais métodos.

Tabela 4 – Comparação dos resultados para o *dataset Iris Plant*

Métdo de Clustering	**Normalized Mutual Information**	**Adjusted Mutual Information**	**Adjusted Rand Score**
K-Means	0,758175680	0,755119168	0,730238272
Expectation Maximization	0,899693545	0,898436103	0,903874232
DBSCAN	0,717464332	0,712576481	0,563751021
Girvan Newman	0,697277259	0,691848700	0,613672138
Girvan Newman (most valuable edge)	0,752721708	0,748206588	0,695596926
Greedy Modularity Optimization	0,764965970	0,761939366	0,684405613
Louvain	0,755735936	0,751607764	0,713600471
Label Propagation	0,752721708	0,748206588	0,695596926

Fonte: Elaborada pelo autor.

Tabela 5 – Comparação dos resultados para o *dataset* *Wine Recognition*

Métdo de Clustering	**Normalized Mutual Information**	**Adjusted Mutual Information**	**Adjusted Rand Score**
Kmeans	0,388433292	0,379466049	0,303442532
EM	0,620766198	0,618272360	0,519924899
DBSCAN	0,408881416	0,401481452	0,277668430
Girvan Newman	0,464441965	0,457979778	0,414017341
Girvan Newman (most valuable edge)	0,464441965	0,457979778	0,414017341
Greedy Modularity Optimization	0,435194212	0,429206239	0,394124337
Louvain	0,383948075	0,371192653	0,238844217
Label Propagation	0,352563995	0,333285493	0,190862753

Fonte: Elaborada pelo autor.

Tabela 6 – Comparação dos resultados para o *dataset* *Wisconsin Breast Cancer*

Métdo de Clustering	**Normalized Mutual Information**	**Adjusted Mutual Information**	**Adjusted Rand Score**
Kmeans	0,421986461	0,420264507	0,412743147
EM	0,706125421	0,705730843	0,811631804
DBSCAN	0,633357447	0,632863608	0,748955571
Girvan Newman	0,488686971	0,487573268	0,578766367
Girvan Newman (most valuable edge)	0,485051698	0,483931757	0,573661885
Greedy Modularity Optimization	0,347591590	0,345813860	0,260453089
Louvain	0,292959226	0,288779021	0,115130394
Label Propagation	0,261151543	0,252782470	0,076342506

Fonte: Elaborada pelo autor.

Por fim, na Tabela 7 é apresentada a modularidade alcançada por cada um dos métodos baseados em redes complexas.

Tabela 7 – Modularidade dos métodos por *dataset*

Método	**Iris**	**Wine**	**Breast Cancer**
Girvan Newman	0,583664568	0,341445542	0,443805156
Girvan Newman com aresta mais central	0,585411408	0,341445542	0,444808456
Greedy Modularity Optimization	0,567799341	0,570860828	0,690578623
Louvain	0,603910473	0,647333754	0,807270317
Label Propagation	0,585411408	0,627492550	0,752708826

Fonte: Elaborada pelo autor.

5 CONCLUSÃO

Para o experimento adotado e algoritmos implementados, verificou-se que, com o aumento das instâncias presentes nos *datasets*, assim como dos atributos, os resultados têm uma redução significativa no NMI. Contudo, os métodos baseados em redes complexas mantêm o patamar que os métodos tradicionais na detecção de *clusters*. A exceção ficou por conta do método Expectation Maximization (método tradicional) que alcançou resultados consideravelmente superiores a todos os demais métodos estudados nesse trabalho.

Vale lembrar que os métodos tradicionais foram trabalhados para buscarem resultados ótimos no presente trabalho e que alguns atributos calculados foram utilizados na construção das redes complexas a partir dos dados.

Como não houve tempo suficiente para verificar condições adicionais aos métodos em rede complexas estudados, não foi possível determinar se as condições poderiam ser melhoradas para aumentar a competitividade desses frente ao Expectation Maximization, por exemplo.

Tinha a expectativa que o uso do mesmo raio ótimo calculado no DBSCAN, que teve melhora significativa no *dataset Wisconsin Breast Cancer*, fosse trazer resultados melhores nos métodos baseados em redes complexas, o que não se confirmou.

Sobre os métodos baseados em redes complexas, os dois métodos baseados no algoritmo de Girvan Newman demandaram mais no quesito processamento de dados, levando a tempos elevados de execução no Google Colaboratory (Colab). Contudo, foram os métodos que apresentaram melhores resultados. Os métodos Louvain e Label Propagation tinham uma resposta muito ágil, assim como o Greedy Modularity Optimization, mas, quanto mais rápido foi o método para detecção dos *clusters*, mais ineficiente foi a detecção das comunidades.

Considerando o exposto, entendo que deveríamos realizar melhoras nos códigos visando execuções mais velozes. Para tanto, pensei na utilização de dicionários para guardar as informações entre vértices em vez de utilizar matrizes cujas qualidades são averiguadas via *for*, pois cada vez que preciso fazer uma interação, em busca de informações, rodarei o número de instâncias da base ao quadrado. Essas informações ordenadas também trariam uma agilidade na execução do código, na criação e análise dos grafos.

A experimentação com medidas de distância diversas da distância euclidiana seria interessante em trabalhos futuros. Acredito que o uso de medidas que ressaltassem a distância entre as distâncias, como medidas exponenciais, traria resultados relevantes. Contudo, algoritmos de pré-processamento dos atributos seriam bem-vindos para que instâncias com características de *outliers* não distorcessem demais o grafo.

Por fim, seria interessante aprofundar em abordagens do tipo *element-centric*, pois essas revelariam quais elementos contribuem para as diferenças entre *clusterings*.

REFERÊNCIAS

ARRUDA, G. F. de; COSTA, L. da F.; RODRIGUES, F. A. A complex networks approach for data clustering. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 391, n. 23, p. 6174–6183, 2018.

BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. **science**, American Association for the Advancement of Science, v. 286, n. 5439, p. 509–512, 1999.

BLONDEL, V. D. *et al.* Fast unfolding of communities in large networks. **Journal of statistical mechanics: theory and experiment**, IOP Publishing, v. 2008, n. 10, p. P10008, 2008.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. **Physical review E**, APS, v. 70, n. 6, p. 066111, 2004.

COMIN, C. H. *et al.* Complex systems: features, similarity and connectivity. **Physics Reports**, Elsevier, v. 861, p. 1–41, 2020.

ERDŐS, P.; RÉNYI, A. On the strength of connectedness of a random graph. **Acta Mathematica Hungarica**, Akadémiai Kiadó, co-published with Springer Science+ Business Media BV . . . , v. 12, n. 1, p. 261–267, 1961.

GIRVAN, M.; NEWMAN, M. E. Community structure in social and biological networks. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 99, n. 12, p. 7821–7826, 2002.

JUNIOR, L. O. Análise de detecção de comunidades de redes complexas para resolver o problema de classificação. Elsevier, 2018.

LANCICHINETTI, A.; FORTUNATO, S. Community detection algorithms: a comparative analysis. **Physical review E**, APS, v. 80, n. 5, p. 056117, 2009.

NEWMAN, M. **Networks: An Introduction**. OUP Oxford, 2010. ISBN 9780191500701. Available at: https://books.google.com.br/books?id=LrFaU4XCsUoC.

NEWMAN, M. **Networks**. [*S.l.: s.n.*]: Oxford university press, 2018.

NEWMAN, M. E. The structure and function of complex networks. **SIAM review**, SIAM, v. 45, n. 2, p. 167–256, 2003.

NEWMAN, M. E. Finding community structure in networks using the eigenvectors of matrices. **Physical review E**, APS, v. 74, n. 3, p. 036104, 2006.

PONS, P.; LATAPY, M. Computing communities in large networks using random walks. *In*: SPRINGER. **International symposium on computer and information sciences**. [*S.l.: s.n.*], 2005. p. 284–293.

RAGHAVAN, U. N.; ALBERT, R.; KUMARA, S. Near linear time algorithm to detect community structures in large-scale networks. **Physical review E**, APS, v. 76, n. 3, p. 036106, 2007.

STROGATZ, S. H. Exploring complex networks. **nature**, Nature Publishing Group, v. 410, n. 6825, p. 268–276, 2001.