UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

WILLIAN MULIA MIRANDA

Representação e caracterização de circuitos amplificadores através de grafos

São Carlos
2022

WILLIAN MULIA MIRANDA

Representação e caracterização de circuitos amplificadores através de grafos

Dissertação apresentada ao Programa de Pós-Graduação em Física do Instituto de Física de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências.

Área de concentração: Física Aplicada
Opção: Física Computacional
Orientador: Prof. Dr. Luciano da Fontoura Costa.

Versão Corrigida
(versão original disponível na Unidade que aloja o Programa)

São Carlos
2022

Miranda, Willian Mulia
Representação e caracterização de circuitos amplificadores através de grafos / Willian Mulia Miranda; orientador Luciano da Fontoura Costa - versão corrigida -- São Carlos, 2022.
115 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Física Aplicada Computacional) -- Instituto de Física de São Carlos, Universidade de São Paulo, 2022.

1. Grafos. 2. Caracterização de grafos. 3. Amplificadores eletrônicos. 4. Imagens. I. Costa, Luciano da Fontoura, orient. II. Título.

À minha esposa e filho, com amor, admiração e gratidão pelo carinho, presença e imensurável apoio ao longo período de elaboração deste trabalho.

AGRADECIMENTOS

Frente aos problemas é onde enxergamos aqueles que são verdadeiros, companheiros e acima de tudo, irmãos. Sendo assim, utilizo este espaço para agradecer às pessoas que tornaram esta dissertação possível, mesmo durante tempos turbulentos.

Primeiramente, agradeço a Deus e aos meus pais José Marcos e Lucia, pelo incomparável e incessante apoio em todos os momentos, pelos conselhos, pela paz e pela constante motivação durante minha trajetória como cientista.

Agradeço a minha esposa Letícia e meu recém-chegado e amado filho Theo, que juntos me deram forças quando as não tinham.

Ao Prof. Dr. Luciano, um grande professor e um gigante orientador que hoje tenho o prazer de chamar de amigo. Saiba que você foi e é essencial para mim, não só na minha trajetória acadêmica, mas também na minha vida pessoal. Sua humildade, paciência, compreensão e paixão pelo que faz é incrível e indiscutível, sendo um exemplo a ser seguido.

Agradeço aos meus colegas de pesquisa Bruno e Ana, que foram essenciais na minha trajetória no mundo acadêmico, assim como no mundo da Física Aplicada.

Agradeço também ao Instituto de Física de São Carlos, pela oportunidade de realização do curso de mestrado, fornecendo o ambiente, as aulas e todas as ferramentas necessárias para a conclusão desta dissertação.

"Se eu vi mais longe, foi por estar sobre ombros de gigantes.”

Isaac Newton

RESUMO

MIRANDA, W. M. **Representação e caracterização de circuitos amplificadores através de grafos**. 2022. 115p. Dissertação (Mestrado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Grafos e redes complexas vêm sendo utilizados para representar os mais diversos sistemas, podendo revelar informações ou características extras em relação ao sistema representado. No ramo da eletrônica, existem diversos circuitos eletrônicos com funções complementares, dentre eles, destacam-se os circuitos amplificadores eletrônicos. No presente trabalho, objetiva-se o estudo, desenvolvimento e aplicação de um programa computacional para o auxílio na obtenção de grafos com base em imagens dos circuitos, com nós e arestas devidamente nomeados e com formato de saída padrão para leitura em outras plataformas ou bibliotecas dedicadas para grafos e/ou redes complexas como, por exemplo, "*iGraph*". Como objetivo complementar, estuda-se os grafos obtidos através de métricas selecionadas, com a finalidade de identificar características que possam auxiliar na identificação dos parâmetros divergentes em cada rede ou circuito. Com isso, desenvolve-se as métricas no ambiente de programação e extrai as medidas pertinentes de cada grafo. Dentre os resultados, observou-se que grafos baseados em circuitos amplificadores eletrônicos apresentam distâncias mínimas médias dentro da faixa entre 1.5 e 3, além de apresentarem uma correlação negativa entre métricas que mensuram complexidade frente à medida de *matching index*. Com base na representação dos dados no espaço Análise de Componentes Principais (PCA), constatou-se que as medidas que mensuram alguma forma de complexidade (como a quantidade de arestas ou grau hierárquico), são as mais relevantes para caracterização dos grafos derivados dos circuitos amplificadores eletrônicos. A partir dos resultados obtidos por um classificador não-supervisionado, com base nas métricas utilizadas, observou-se uma predominante adjacência entre os circuitos analisados.

Palavras-chave: Grafos. Caracterização de grafos. Amplificadores eletrônicos. Imagens.

ABSTRACT

MIRANDA, W. M. **Representation and characterization of amplifier circuits through graphs**. 2022. 115p. Dissertation (Master in Science) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Complex networks and graphs have been used to represent the most diverse systems, revealing important information or characteristics of the represented system. In the field of electronics, there are several electronic circuits with complementary functions, among them, electronic amplifier circuits stand out. In the present work, the main objective is the study, development and application of a computer program to help in obtaining graphs based on images, with nodes and edges properly named and with standard output format for reading on other platforms or libraries dedicated to graphs and/or complex networks such as "iGraph". As a complementary objective, the graphs obtained through the measurement of selected metrics are studied, in order to identify characteristics that may help in the identification of divergent parameters in each network or circuit. The metrics are developed in the programming environment, allowing the quantification of relevant measures in each graph. Among the results, it was observed that graphs based on electronic amplifier circuits present minimum average distances within the range between 1.5 and 3, in addition to presenting an inverse proportionality between metrics that measure complexity against the matching index measure. Based on the representation of data in the Principal Component Analysis (PCA) space, it was found that metrics that measure some form of complexity (such as the number of edges or hierarchical degree) are the most relevant for graphs based on electronic amplifier circuits. In view of the results obtained by an unsupervised classifier, based on the metrics used, the circuits resulted mostly adjacent in the considered measurement space.

Keywords: Graphs. Graph characterization. Electronic amplifiers. Images.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BFS	*Breadth-first Search*
FET	*Field Effect Transistor*
GUI	*Graphical User Interface*
NPN	Negativo-Positivo-Negativo
PCA	Análise de Componentes Principais
PCA1	Primeiro eixo no espaço PCA
PCA2	Segundo eixo no espaço PCA
PNP	Positivo-Negativo-Positivo
TBJ	Transistor Bipolar de Junção
UI	*User Interface*

LISTA DE SÍMBOLOS

A_d	Fator de ganho do amplificador diferencial
β	Ganho de corrente
C	Capacitor
C_{corr}	Coeficiente de correlação
C_i	Coeficiente de aglomeração do nó i
$\bar{C}$	Coeficiente de aglomeração médio
σ_C	Desvio padrão do coeficiente de aglomeração
d_{ij}	Distância mínima entre os nós i e j
$\bar{d}$	Distância mínima média
σ_d	Desvio padrão da distância mínima
h	Vizinhança
i_c	Corrente de coletor
i_b	Corrente de base
k	Quantidade de centroides de classificação para o classificador *k*-Médias
k_i	Grau do nó i
$\bar{k}$	Grau médio
σ_k	Desvio padrão do grau
R	Resistor
R_c	Resistor de coletor
R_e	Resistor de emissor
T	Transistor de junção bipolar
μ_{ij}	*Matching index* da aresta que conecta os nós i e j
$\bar{\mu}$	*Matching index* médio
σ_μ	Desvio padrão do *matching index*
V_{cc}	Sinal continuo de alimentação positivo
V_{ee}	Sinal contínuo de alimentação negativo
v_i	Sinal alternado de entrada
v_o	Sinal alternado de saída

SUMÁRIO

1 INTRODUÇÃO

Este capítulo tem o objetivo de introduzir brevemente os assuntos abordados neste trabalho, de forma a apresentar alguns conceitos elementares de aplicação de grafos e dos circuitos amplificadores, de forma a contextualizar o tema proposto, além de apresentar a motivação, as hipóteses e os objetivos deste trabalho.

Grafos e redes complexas têm sido vastamente aplicados nas mais variadas áreas científicas e tecnológicas. (1) Estas estruturas possuem papeis fundamentais em pesquisas nas áreas de matemática e ciência da computação, porém muitos desenvolvimentos em redes complexas estão acontecendo em áreas como a sociologia, biologia e física. (2) Áreas como redes neurais, redes de tráfego, redes sociais e eletrônica também vêm-se aperfeiçoando cada vez mais pelo uso de recursos advindo dos grafos e redes complexas. (3)

Em um nicho específico da eletrônica, encontram-se os circuitos amplificadores, que possuem grande variedade com diversas características e arquiteturas, sendo muitos desses com funções similares ou complementares. Este tipo de circuitos eletrônicos tem como principal finalidade a amplificação do sinal de entrada, ou seja, permitem a intensificação, seja de tensão, corrente ou propriamente a potência do circuito. (4) Existem circuitos amplificadores originalmente criados para realizar operações matemáticas como, soma, subtração, multiplicação, diferenciação, integração, inversão de sinais, etc. Atualmente, estes dispositivos possuem vasta aplicabilidade em áreas como controle de sistemas, instrumentação, conversão de escala para adaptação de sinais para sistemas diversos, entre outras. (5)

Tomando-se como base que circuitos eletrônicos (digitais e analógicos) podem ser representados por grafos e que também podem apresentar padrões redes complexas do tipo pequeno mundo ou "*small world"* (6), e que o uso da teoria de grafos pode auxiliar na simplificação de circuitos eletrônicos analógicos mais complexos, otimizando a análise de forma significativa (7), o estudo de circuitos amplificadores com o auxílio da teoria de grafos pode revelar novas características para melhor se caracterizar e compreender os tipos destes circuitos, sendo que muitas vezes circuitos amplificadores são vistos como uma caixa preta universal e idealizada. (8)

Neste aspecto, o desenvolvimento de ferramentas e métodos com o objetivo de criar grafos baseados em circuitos eletrônicos que inclua uma análise baseada em métricas para grafos, apresentam-se como pontos chave neste tipo de pesquisa. Este trabalho se encaixa neste contexto, apresentando o desenvolvimento de um programa computacional que auxilia o desenvolvimento de grafos baseados em imagens de esquemáticos de circuitos eletrônicos

amplificadores, assim como apresentar um estudo preliminar dos grafos resultantes através de medidas de grafos e redes complexas, de forma a obter uma classificação a partir de um classificador não supervisionado.

A metodologia proposta incluiu o desenvolvimento do programa computacional para o auxílio no desenvolvimento dos grafos baseados nos circuitos eletrônicos amplificadores. Na sequência, deve-se aferir medidas específicas em cada grafo obtido e, por fim, gerar histogramas e organogramas com finalidade de identificar possíveis agrupamentos e novas características dos tipos de circuitos considerados, com o auxílio de um classificador não-supervisionado. A metodologia de pesquisa utilizada está resumidamente representada pela Figura 1.

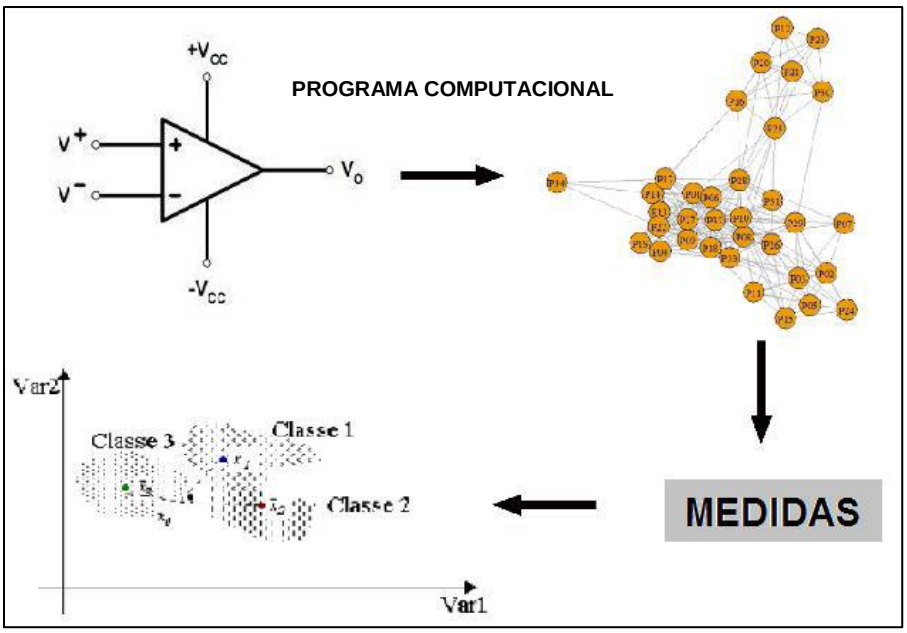


Figura 1 – Metodologia de pesquisa utilizada
Fonte: Elaborada pelo autor

1.1 Questões de pesquisa

Dentre as principais questões consideradas neste trabalho, menciona-se:

- É possível desenvolver um programa computacional para auxiliar a obtenção de grafos a partir de imagens de circuitos amplificadores?
- É possível classificar os circuitos amplificadores eletrônicos com funções similares a partir das medidas de grafos?
- Quais são as medidas mais relevantes para os grafos baseados em circuitos eletrônicos amplificadores?

- Quais são as principais características, relacionamentos e tipos de circuitos amplificadores eletrônicos?

1.2 Objetivos

O presente estudo possui o objetivo de desenvolver um programa computacional que auxilie na conversão de imagens de circuitos eletrônicos amplificadores em grafos, assim como respectiva aplicação preliminar para caracterizar os tipos de circuitos a partir de medidas dos respectivos grafos. Dentre os objetivos específicos deste estudo, apresentam-se:

- Analisar a viabilidade do uso de um programa computacional no processo de conversão de imagens e grafos;
- Validar e aplicar estes recursos computacionais para a obtenção de grafos representando diversos circuitos amplificadores analógicos;
- Identificar, dentre as medidas utilizadas, quais delas possuem maior relevância frente a circuitos eletrônicos amplificadores;
- Verificar as relações entre os circuitos considerados através de métodos estatísticos multivariados e reconhecimento de padrões.

1.3 Organização do texto

O presente trabalho está subdividido em cinco capítulos, sendo eles:

(1) Introdução: Apresenta uma breve contextualização dos principais assuntos abordados, incluindo as motivações, objetivos, metodologia e as principais questões relacionadas à pesquisa;

(2) Fundamentação teórica: Neste capitulo são apresentadas as bases teóricas dos assuntos relacionados à pesquisa desenvolvida, como: os principais circuitos amplificadores eletrônicos, os fundamentos de grafos e a teoria fundamental das métricas, classificador e das ferramentas em geral utilizadas ao longo da pesquisa;

(3) Materiais e métodos: Neste capítulo, apresenta-se e desenvolve-se as ferramentas utilizadas, sendo elas: a estrutura do programa computacional (interface e algoritmos fundamentais), as métricas e o classificador com base no referencial teórico abordado no capítulo de fundamentação teórica;

(4) Resultados e discussões: Este capítulo apresenta os resultados da parte computacional, tanto em questão da interface gráfica, como quanto seu respectivo funcionamento. Também apresenta os resultados e discussão dos grafos derivados dos circuitos amplificadores eletrônicos, com base nas características extraídas pelas métricas implementadas, a partir de gráficos e histogramas que revelam os inter-relacionamento entre os circuitos/grafos abordados;

(5) Conclusão: Faz-se uma recapitulação dos principais assuntos abordados e responde às questões da pesquisa. Enfatizam-se as principais contribuições e apresentam-se algumas sugestões de trabalhos futuros.

Ao fim do capitulo de conclusão, apresentam-se as referências deste trabalho seguidas por apêndices que apresentam os algoritmos de implementação do programa computacional (APÊNDICE A) e das métricas e do classificador desenvolvidos (APÊNDICE B). Também, ainda por meio de apêndices, apresentam-se as principais características dos circuitos amplificadores analisados (APÊNDICE C), assim como suas respectivas representações através de grafos (APÊNDICE D). No APÊNDICE E apresenta-se um exemplo de uma lista de adjacência resultante do programa computacional.

2 FUNDAMENTAÇÃO TEÓRICA

Este capitulo tem o objetivo de apresentar as bases teóricas relacionadas aos objetos de discussão e métodos empregados durante o trabalho.

2.1 Amplificadores analógicos

Amplificador é um termo genérico usado para descrever circuitos que amplificam seu sinal de entrada (9). Em geral, os amplificadores possuem a estrutura apresentada na Figura 2.

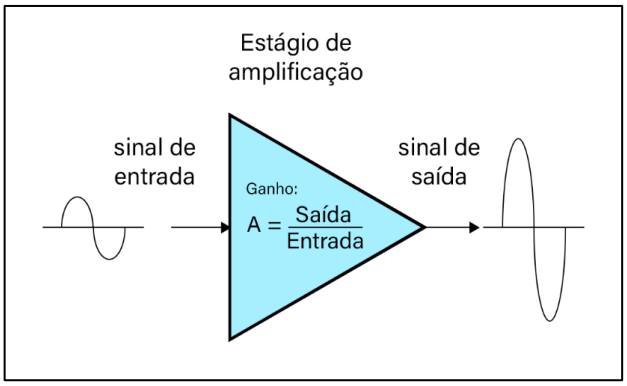


Figura 2 – Estrutura geral de um amplificador
Fonte: Elaborada pelo autor

Dentre os tipos de amplificadores mais conhecidos, pode-se citar os amplificadores lineares, como aqueles construídos a partir de estágios amplificadores a transistores de junção bipolar (10), e que apresentam diversos modelos de circuitos para as mais diversas aplicações. Para este trabalho, pode-se destacar os circuitos: amplificador emissor-comum por transistor; amplificador *push-pull*; amplificador diferencial e amplificador divisor de fase (*phase-splitter*).

2.1.1 Amplificador emissor comum por transistor de junção bipolar

Um amplificador emissor comum envolve um transistor de junção bipolar de estágio simples. Este amplificador recebe um sinal de entrada v_i no terminal de base e emite um sinal de saída v_o no coletor, possuindo o emissor comum para todos os terminais. (11) A Figura 3 apresenta o circuito básico de um amplificador emissor comum.

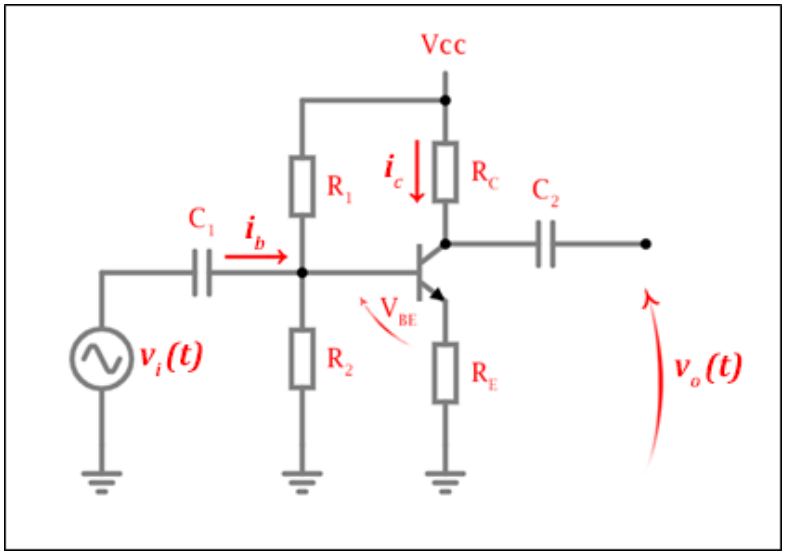


Figura 3 – Circuito amplificador emissor comum
Fonte: Elaborada pelo autor

O ganho de corrente β do circuito pode ser determinado pela razão da variação da corrente de coletor i_c pela corrente de base i_b (11) conforme mostra a Equação 2.1.

$$\beta = \frac{\Delta i_c}{\Delta i_b} \tag{2.1}$$

Os resistores R_1, R_2 e R_E são utilizados para polarizar e estabilizar o transistor em um ponto de operação Q conveniente. Os capacitores C_1 e C_2 são chamados de capacitores de acoplamento, que possuem função de isolar as componentes de corrente contínua do amplificador.

Esta topologia de amplificador possui inversão de fase (o que o caracteriza como amplificador inversor) e possui parâmetros de ganho de corrente, tensão e, consequentemente, potência elevados. (12)

2.1.2 Amplificador *push-pull*

Um amplificador *push-pull* é um amplificador de potência que contém um par de transistores responsáveis por alimentar (*push*) ou drenar (*pull*) corrente na carga. (13) A Figura 4 apresenta, de forma bastante simplificada, a topologia típica deste amplificador.

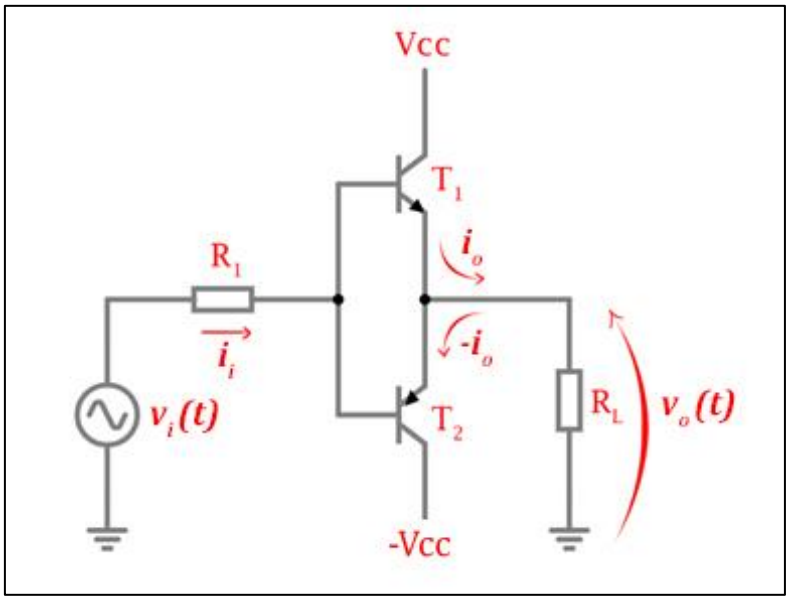


Figura 4 - Circuito amplificador *push-pull*
Fonte: Elaborada pelo autor

Nessa topologia, é possível perceber que o transistor de junção bipolar NPN T_1 é responsável por fornecer corrente de V_{cc} para a carga R_L no ciclo positivo do sinal de entrada v_i. Por sua vez, o transistor de junção bipolar PNP T_2 drena corrente da carga R_L para o terminal terra durante o ciclo negativo de v_i.

Este amplificador apresenta maior eficiência quando comparado ao amplificador emissor comum (Seção 2.1.1) visto que não há fluxo de corrente pelo circuito quando não há sinal presente em sua entrada, ou seja, a polarização em corrente contínua não consome potência.

A saída deste amplificador consiste na combinação entre as correntes de coletor dos dois transistores. Por outro lado, essa topologia causa uma distorção chamada *crossover*, causada pela diferença de polarização de chaveamento entre os dois transistores. (13)

2.1.3 Amplificador diferencial

Um amplificador diferencial é um tipo de amplificador eletrônico que amplifica a diferença entre dois sinais de entrada v_{i1} e v_{i2} e, em teoria, elimina qualquer sinal comum entre as duas entradas. A Figura 5 apresenta um exemplo de circuito de um amplificador diferencial com dois transistores de junção bipolar.

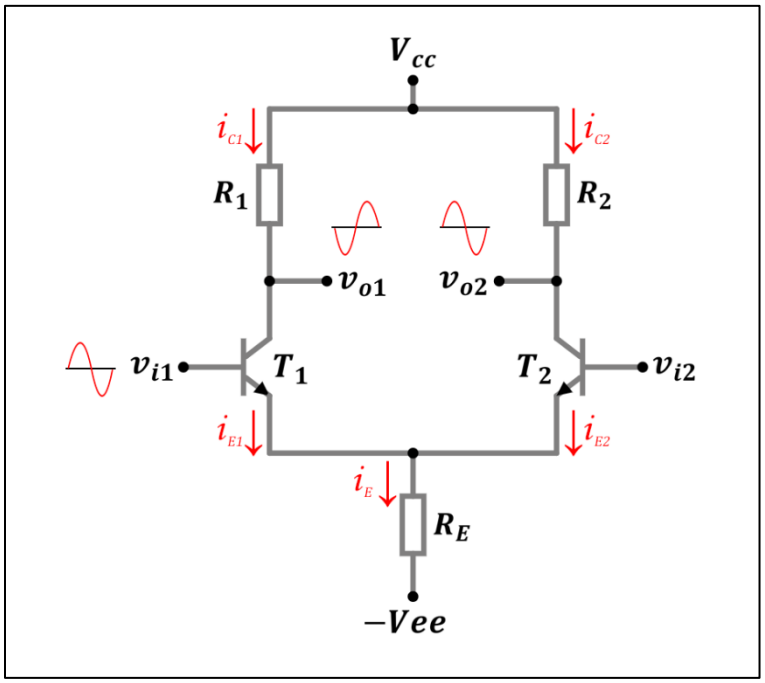


Figura 5 – Circuito amplificador diferencial com dois transistores
Fonte: Elaborada pelo autor

A partir da Figura 5, pode se perceber que, nesta arquitetura, o circuito possui dois transistores de junção bipolar do tipo NPN T_1 e T_2, possui alimentação simétrica (V_{cc} e V_{ee}), e utiliza três resistores, dois deles sendo resistores de coletor (R_1 e R_2) e um resistor de emissor comum entre ambos transistores. (14)

Este tipo de circuito tem a característica de amplificar a diferença dos sinais v_{i1} e v_{i2} na proporção de um certo ganho A_d. Sendo assim, pode-se definir a Equação 2.3, que relacionam os sinais de saída v_{o1} e v_{o2} frente aos sinais de entrada v_{i1} e v_{i2}.

$$v_{o1} - v_{o2} = A_d(v_{i2} - v_{i1}) \tag{2.3}$$

Considerando o sinal v_{i1} do tipo senoidal, enquanto v_{i1} aumenta, o transistor T_1 passa a conduzir, permitindo um fluxo de corrente i_{c1} pelo seu coletor. Isso faz com que aumente a queda de tensão sobre o resistor de coletor R_1, resultando na queda do sinal v_{o1}.

Conhecendo esse comportamento, pode-se observar que as variações no sinal de entrada v_{i1} são refletidas em v_{o2} e aparecem com defasagem de 180° em v_{o1}. (14)

Além deste circuito não possuir referencial ao terminal terra, dentre as características principais deste circuito, pode-se citar a rejeição ao modo comum, ou seja, este circuito anula (ou ameniza) os sinais em comum de ambos os sinais de entrada, o que o torna menos suscetível a ruídos.

2.1.4 Amplificador divisor de fase

O estágio divisor de fase utiliza dois transistores que recebem e amplificam separadamente o clico positivo e negativo da onda do sinal de entrada. (15) A Figura 6 mostra o estágio divisor de fase atuando na entrada do estágio *push-pull*.

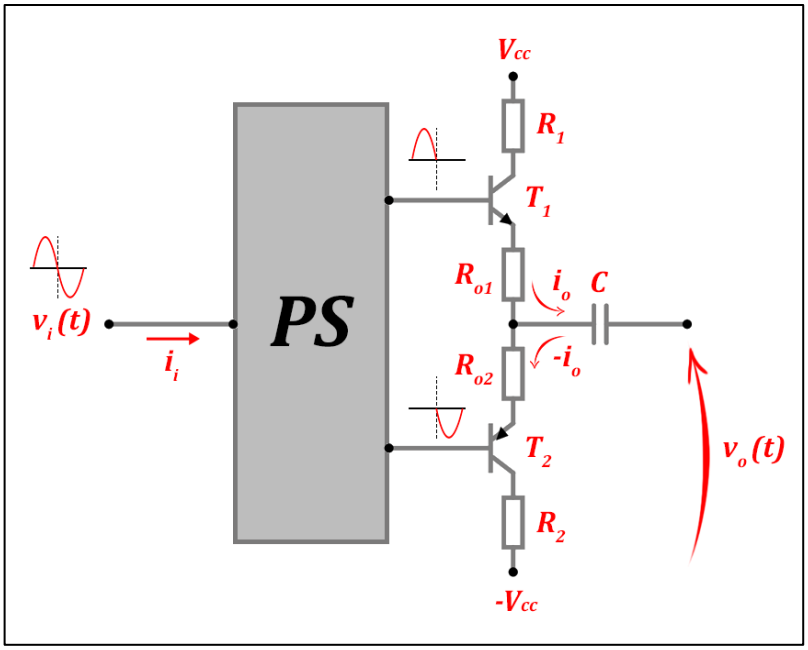
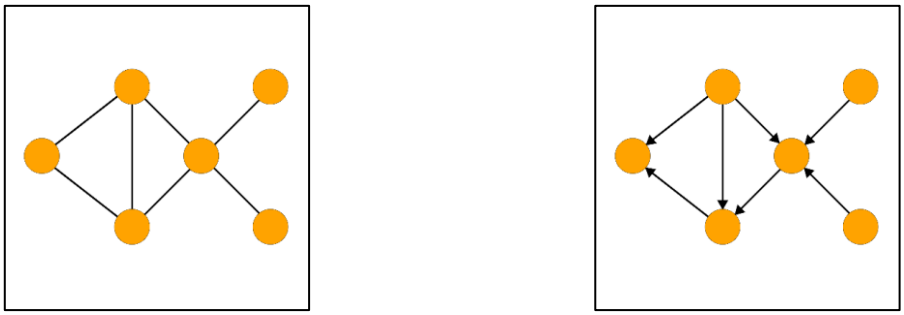


Figura 6 – Estágio divisor de fase atuando na entrada do estágio *push-pull*
Fonte: Elaborada pelo autor

O estágio divisor de fase separa os ciclos positivo e negativo, utilizando o primeiro destes ciclos para controlar o transistor T_1 do estágio push-pull, enquanto o ciclo negativo atua sobre o transistor T_2, controlando a parte negativa do sinal de saída.

2.2 Fundamentos de grafos

Um grafo G (também conhecido como rede (2)) é um conjunto de pontos V (conhecidos como nós ou vértices) e linhas E que ligam estes pontos (conhecidas como arestas ou ligações) quais podem ser direcionadas ou não, caracterizando assim o tipo de grafo (direcionado ou não-direcionado). Pode-se definir um grafo pelo conjunto de pares ordenados $G = (V, E)$ onde $E = \{\{x, y\}: x, y \in V\}$ (7). Quando corretamente representados, os grafos tendem a ser uma maneira prática de identificar relações entre objetos. (3) A Figura 7 mostra exemplos de grafo direcionado e não-direcionado.



(a) (b)

Figura 7 - Tipos de grafo. (a) Grafo não direcionado. (b) Grafo direcionado
Fonte: Elaborada pelo autor

Atualmente utilizados nas mais diversas áreas, os grafos vêm sendo uma ferramenta importante ao longo dos anos.

2.2.1 Contexto histórico do estudo dos grafos

De acordo com a literatura, considera-se a resolução de Euler do Problema das Pontes de Königsberg, publicada em 1736, como sendo a primeira publicação da teoria de grafos. Esse problema foi motivado na antiga cidade de Königsberg na Prússia oriental, qual era dividida pelo Rio Pregel. Esse rio delimitava uma ilha (conhecida como Kneiphof) e também se dividia em dois ramos em um ponto específico. Para os cidadãos dessa cidade viajar mais facilmente entre as partes dela, foram construídas sete pontes. O que diziam é que as pessoas da cidade costumavam se entreter em criar uma rota pela cidade que utilizaria apenas uma vez cada uma das sete pontes. Intrigado com problema, o matemático Leonhard Euler escreveu um artigo sobre esse problema e concluiu que não havia solução para o desafio ao perceber que só seria possível percorrer o caminho inteiro passando uma única vez em cada ponto se houvesse exatamente zero ou dois pontos de onde saísse um número ímpar de caminhos. A partir deste estudo, Euler desenvolveu o primeiro teorema da teoria de grafos, assim como vários conceitos básicos da teoria de grafos. (16)

Iniciando-se com este fato histórico, o interesse nos grafos e, posteriormente, nas redes complexas, foi se tornando cada vez maior e sendo aplicado em áreas cada vez mais diversas.

2.2.2 Aplicabilidade dos grafos

A teoria dos grafos é usada em uma extensa variedade de áreas de pesquisa como, por exemplo, engenharia, matemática, sociologia, economia, redes de computadores, administração

e marketing empresarial. (1) Problemas como planejamento de rotas para empresas de entrega ou coleta de lixo, por exemplo, podem ser modelados por caminhos formados ao percorrer uma aresta ao longo de um grafo. (17)

Pode-se apontar a importância dos grafos também no estudo de relações pessoais por meio das redes sociais, comunicações, economia, mercado financeiro, ciência da computação, internet, transporte, sistemas de transmissão de energia elétrica, redes biomoleculares, medicina, ecologia, neurociência, linguística, física, química, matemática, clima, segurança, contágio epidêmico e muitas outras áreas.

2.2.3 Redes complexas

As redes complexas, principal objeto da ciência de redes, vêm sendo vastamente utilizadas para representar, caracterizar e modelar diversos fenômenos, tanto no ponto de vista teórico quanto aplicado. (18) Além da particular efetividade das redes complexas na representação virtualmente qualquer sistema discreto, temos também a ênfase que esta abordagem coloca na importância das interconexões entres os respectivos elementos constituintes do sistema de interesse.

Conforme Costa (19), “Redes complexas são grafos que diferem substancialmente de um grafo regular ou de um grafo estatístico regular”. Em particular, tais redes possuem distribuições heterogêneas de várias de suas medidas topológicas, não podendo, portanto, serem abreviadamente descritas em termos dos respectivos valores médios.

Um ponto de particular interesse na pesquisa na área de ciência de redes refere-se ao possível relacionamento entre a estrutura de interconectividade de uma rede e as propriedades de dinâmicas implementadas sobre a mesma. O caso de circuitos eletrônicos representa um interessante exemplo deste tipo de pesquisa, pois as propriedades dinâmicas dos circuitos estarão em boa parte influenciadas pela interconexão entre os respectivos componentes.

2.2.4 Formas de representação dos grafos

Existem várias formas de se representar um grafo, cada forma difere-se no custo computacional (ex. armazenamento), facilidade de leitura, entre outras características. (20)

Dentre as várias formas disponíveis, são usadas neste estudo duas formas, sendo elas: matriz de adjacência (*adjacency matrix*) e lista de adjacência (*adjacency list* ou *edge list*).

2.2.4.1 Matriz de adjacência

Conforme Singh e Sharma (17), “Na matemática e na ciência da computação, uma matriz adjacência é um meio de representar quais vértices de um grafo são adjacentes a outros vértices”.

Uma matriz de adjacência se trata de uma matriz $n\ x\ n$ (onde n é o número de vértices de um determinado grafo) em que cada posição ij na matriz representa uma conexão entre o vértice i e j. A Figura 8 apresenta um grafo e sua devida matriz de adjacência.

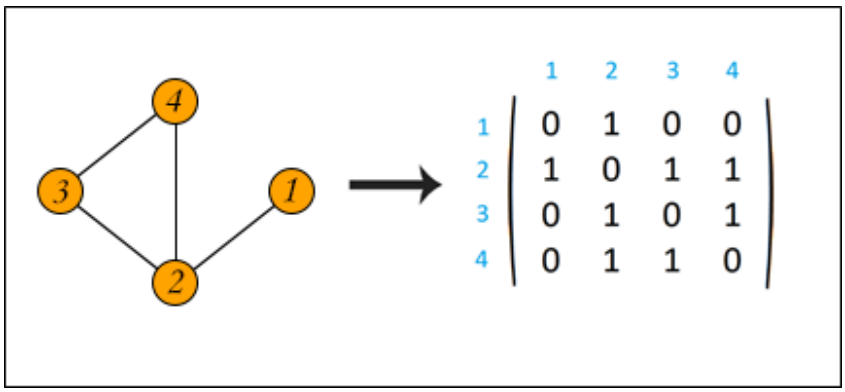


Figura 8 – Exemplo de grafo não-direcionado e sua respectiva matriz adjacência
Fonte: Elaborada pelo autor

Tendo como exemplo o grafo apresentado na Figura 8, pode se observar que o mesmo possui quatro vértices o que leva à uma matriz adjacência com dimensões 4 x 4. Ao observar as ligações, vê-se uma ligação entre os vértices 1 e 2, logo essa é representada na posição na linha 1 e coluna 2. Portanto, expandindo essa análise para todos os outros vértices do grafo, obtêm-se a devida matriz adjacência.

Se o grafo a ser representado for não-direcionado e com zeros na diagonal principal (sem auto ligações), a matriz de adjacência resultante se trata de uma matriz simétrica. (17)

Esta forma de representação, por se tratar de uma matriz numérica, demanda relativo baixo custo computacional de armazenamento e utilização quando comparado a outras formas de representação.

2.2.4.2 Lista de adjacência

Uma lista de adjacência define-se por uma lista onde cada elemento representa uma ligação (aresta) de um determinado grafo. Cada elemento na lista possui, obrigatoriamente, um vértice de saída (de onde está saindo a ligação) e um vértice de destino (onde termina a ligação).

Também pode-se incluir, além das informações obrigatórias, outras informações de acordo com a necessidade (17) como: nome da aresta, intensidade da ligação, atributos visuais como cor da aresta, entre outras. A Figura 9 mostra uma representação de um grafo em forma de lista de adjacência.

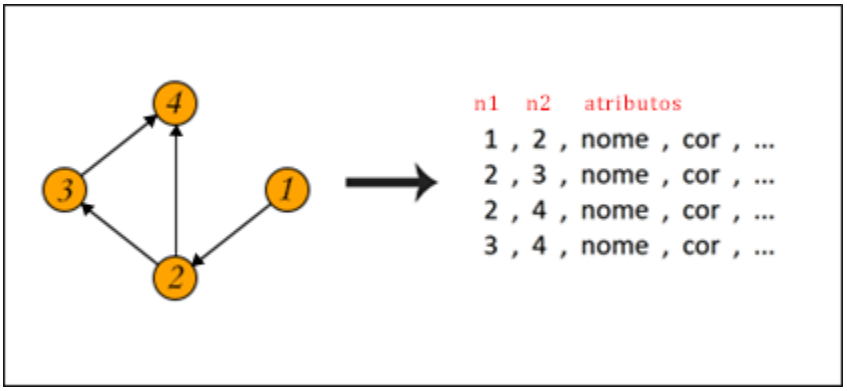


Figura 9 – Exemplo de grafo direcionado e sua respectiva lista de adjacência
Fonte: Elaborada pelo autor

2.2.5 Métricas em grafos

Quando se trata de caracterizar as propriedades topológicas de grafos, há uma variedade de medidas possíveis de serem extraídas considerando diferentes aspectos de cada grafo. Dentre esta variedade, e frente às características topológicas dos circuitos eletrônicos, destacam-se as seguintes medidas que estão aqui apresentadas.

2.2.5.1 Grau

Sendo uma das medidas mais comuns utilizadas em grafos, o grau de um vértice representa o número de arestas a ele conectadas (também conhecido na literatura como "conectividade") (2). Considerando dois vértices i e j, o grau de um vértice, consequentemente k_i, em um grafo não direcionado pode ser obtido através da Equação 2.4.

$$k_i = \sum_j a_{ij} = \sum_j a_{ji} \tag{2.4}$$

Considerando N a quantidade de vértices em um grafo, a partir do grau de todos os vértices, obtêm-se o grau médio ($\bar{k}$) do grafo e seu respectivo desvio padrão (σ_k) de acordo com a Equação 2.5 e Equação 2.6, respectivamente.

$$\bar{k} = \frac{1}{N}\sum_{i} k_i \tag{2.5}$$

$$\sigma_k = \sqrt{\frac{1}{N}\sum_{i}\left(k_i - \bar{k}\right)^2} \tag{2.6}$$

Em grafos não-direcionados, o grau médio baixo indica uma rede pouco conectada, podendo haver vários vértices isolados ou aglomerados com pequenas quantidades de conexões. (2)

2.2.5.2 Distância mínima

Em um grafo não direcionado e não ponderado, pode-se definir o comprimento do caminho entre os vértices i e j como sendo a quantidade de arestas presentes no caminho. Dentre os vários caminhos possíveis, destaca-se aquele que possui o menor comprimento, ou seja, aquele caminho que liga o vértice i ao vértice j com a menor quantidade de arestas possíveis. Este caminho especifico é chamado de caminho geodésico (ou caminho mais curto). O comprimento deste caminho geodésico é conhecido como a distância geodésica ou distância mínima d_{ij} entre os vértices i e j. (2)

A distância mínima média $\bar{d}$ e o desvio padrão σ_d de um grafo pode ser determinada, respectivamente, de acordo com a Equação 2.7 e Equação 2.8.

$$\bar{d} = \frac{1}{N(N-1)}\sum_{i \neq j} d_{ij} \tag{2.7}$$

$$\sigma_d = \sqrt{\frac{1}{N(N+1)}\sum_{i \neq j}\left(d_{ij} - \bar{d}\right)^2} \tag{2.8}$$

Com base na Equação 2.7, pode-se perceber que a função diverge para casos de vértices não conectados na rede. Devido a isto, entram no somatório apenas pares de vértices conectados (2).

2.2.5.3 Grau hierárquico

Diferente da maioria das métricas topológicas para caracterização de cada nó em redes complexas, as medidas hierárquicas não levam em consideração apenas sua vizinhança imediata, mas também as próximas vizinhanças de acordo com sucessivos níveis hierárquicos.

Isso fornece informações adicionais sobre as características topológicas da rede. (21) Dentre essas medidas hierárquicas, destaca-se o grau hierárquico.

Para um nó n_i e a sua respectiva quantidade de nós vizinhos na vizinhança h, tem-se o grau hierárquico $k_i(h)$. Percebe-se que para $h = 0$, a medida se trata exatamente da métrica de grau (como visto na Seção 2.2.5.1). Porém, para $h > 0$, o grau hierárquico é definido pela quantidade de arestas que ligam nós vizinhos aos nós pertencentes à vizinhança h, desconsiderando os nós do mesmo nível e anteriores. A Figura 10 mostra o exemplo de um grafo e seus respectivos graus hierárquicos.

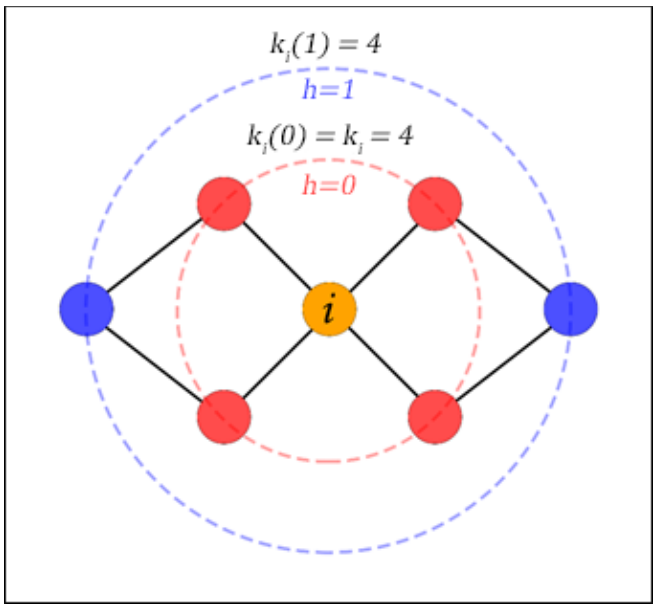


Figura 10 – Graus hierárquicos do nó i
Fonte: Elaborada pelo autor

A partir da observação da Figura 10, com base no nó i, é possível perceber que, para sua vizinhança imediata ($h = 0$), existem quatro ligações com os nós representados em cor vermelha, consequentemente, o grau hierárquico $k_i(0)$ é igual a quatro (valor idêntico ao grau tradicional k_i). Pela análise da próxima vizinhança ($h = 1$), percebe-se que existem quatro ligações que ligam os nós vermelhos aos dois nós azuis, portanto, o grau hierárquico $k_i(1)$ é igual a quatro.

2.2.5.4 *Matching index*

A métrica de *matching index* pode ser utilizada para mensurar a similaridade entre a conectividade de dois vértices conectados por uma aresta. Altos valores de *matching index* identifica uma aresta que conecta regiões similares na rede, caso contrário, indica uma ligação

que conecta regiões pouco similares. Para uma determinada aresta (i, j), o coeficiente de *matching index* μ_{ij} pode ser computado pela razão da quantidade de vértices vizinhos comuns aos vértices i e j (conexões para outro mesmo vértice k) e o número total de conexões de ambos os vértices (excluindo conexões entre i e j), desconsiderando-se os vizinhos comuns, conforme mostra a Equação 2.9. (2)

$$\mu_{ij} = \frac{\sum_{k \neq i,j} a_{ik} a_{jk}}{\sum_{k \neq j} a_{ik} + \sum_{k \neq i} a_{jk}} \tag{2.9}$$

Considerando-se *M* a quantidade de arestas de um grafo, pode-se representar o *matching index* médio $\bar{\mu}$ e desvio padrão σ_μ referente a todo grafo respectivamente conforme a Equação 2.10 e Equação 2.11.

$$\bar{\mu} = \frac{1}{M} \sum_{i \neq j} \mu_{ij} \tag{2.10}$$

$$\sigma_\mu = \sqrt{\frac{1}{M} \sum_{i \neq j} \left(\mu_{ij} - \bar{\mu}\right)^2} \tag{2.11}$$

2.2.5.5 Coeficiente de aglomeração

Em grafos, é comum se deparar com estruturas triangulares, e uma das medidas que caracterizam este tipo de estrutura é o coeficiente de aglomeração (*clustering coefficient*). Basicamente, essa medida leva em conta as conexões entre os vizinhos de um nó, que quando presentes, formam uma estrutura triangular entre os 3 vértices. Para um determinado nó i, uma das formas de se calcular seu coeficiente de aglomeração C_i é através da Equação 2.12, onde k_i representa a quantidade de vizinho do vértice i e l_i representa a quantidade de ligações entre esses vizinhos. (2)

$$C_i = \frac{2l_i}{k_i(k_i - 1)} \tag{2.12}$$

Em outras palavras, o coeficiente de aglomeração pode ser definido como uma taxa que relaciona quantos triângulos o vértice i participa frente ao número máximo de triângulos que seriam possíveis de acordo com a quantidade de vizinhos.

Visto que o coeficiente apresentado na Equação 2.12 é relativo a um determinado vértice, considerando N a quantidade de vértices em um grafo, uma das maneiras de se obter o coeficiente de aglomeração de todo o grafo é através da média aritmética dos coeficientes de aglomeração de cada vértice, conforme a Equação 2.13. (2)

$$\bar{C} = \frac{1}{N}\sum_{i} C_i \tag{2.13}$$

Também pode-se obter o desvio padrão σ_C através da Equação 2.14.

$$\sigma_C = \sqrt{\frac{1}{N}\sum_{i}(C_i - \bar{C})^2} \tag{2.14}$$

2.3 Grafos de circuitos eletrônicos

Um circuito eletrônico consiste em elementos interconectados com comportamento geralmente dependente de dois fatores, sendo eles: as características de cada elemento internamente conectado e a regra pela qual estes elementos são conectados. A partir deste segundo fator, pode-se observar uma relação entre circuitos elétricos e a teoria de grafos, onde um elemento elétrico de dois terminais pode ser representado por uma aresta. A partir dessa relação, podemos obter um grafo a partir de um circuito considerando os componentes elétricos como arestas e seus respectivos terminais como nós. (7)

Um exemplo de representação de um circuito eletrônico através de um grafo pode ser observado na Figura 11.

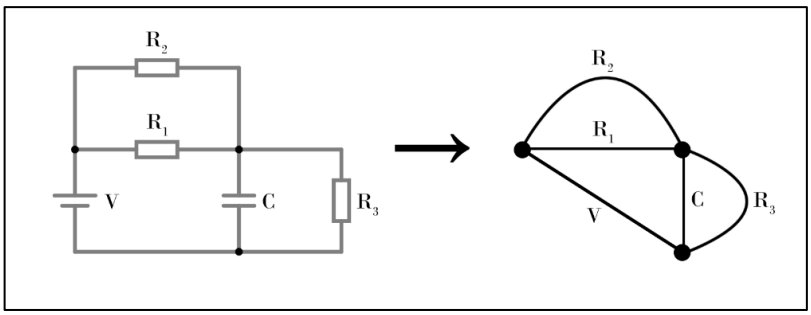


Figura 11 – Grafo a partir de um circuito eletrônico
Fonte: Elaborada pelo autor

A partir desse conceito, pode-se perceber que uma aresta é capaz de representar um componente de apenas dois terminais, portanto, para representação de um elemento com mais de dois terminais, necessita acrescentamos mais arestas para relacionar todos os terminais deste componente. Para exemplificação, podemos citar o componente eletrônico transistor de junção bipolar, cujo qual possui três terminais distintos, cujo qual é representado por três arestas que relacionam seus respectivos terminais, sendo elas: base-coletor (BC), base-emissor (BE) e coletor-emissor (CE). A Figura 12 mostra a representação em grafo de um transistor de junção bipolar do tipo NPN.

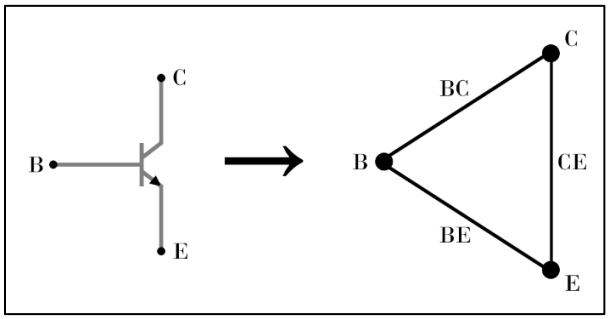


Figura 12 – Grafo a partir de um transistor de junção bipolar NPN
Fonte: Elaborada pelo autor

Com isso, pode-se então representar circuitos eletrônicos que possuem os mais diversos componentes utilizando estes conceitos apresentados nesta seção. Pode-se também, caso necessário, atribuir função sobre a direção da aresta (grafos direcionados) onde a direção pode indicar a polaridade do componente ou a direção do fluxo de corrente.

2.4 Classificador *k*-Médias

Um classificador *k*-médias é dito ser não supervisionado, baseado em um método iterativo que consiste em particionar um conjunto de n objetos em $k > 2$ grupos, conhecidos no inglês como *clusters*. (22) O algoritmo desse classificador por ser definido através de quatro passos conforme apresentado na Figura 13.

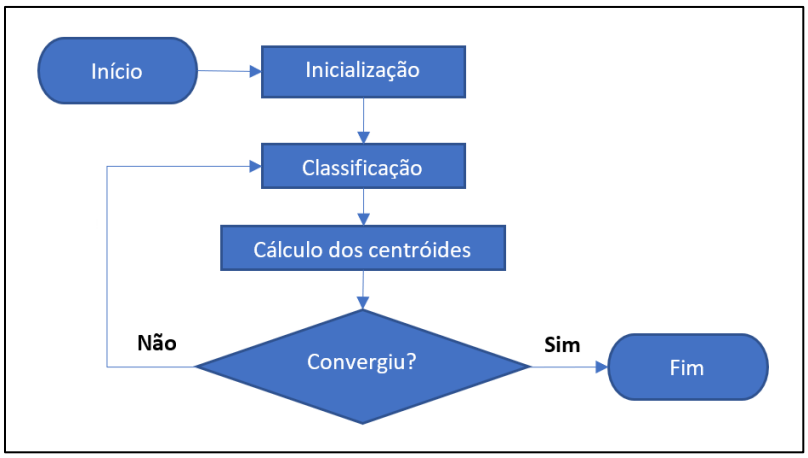


Figura 13 – Fluxograma do algoritmo padrão *k*-médias
Fonte: Elaborada pelo autor

De acordo com o fluxograma apresentado na Figura 13, o primeiro passo consiste na inicialização das variáveis, ou seja, realizar a leitura dos dados a serem classificados, assim como inicializar as posições dos centroides dos grupos de classificação. Após isso, inicia-se o

passo de classificação dos dados, onde obtém-se a distância Euclidiana de todos os objetos para todos os centroides e, então, atribuindo cada objeto ao grupo do centroide mais próximo. Ao fim da etapa de classificação, o algoritmo realiza o cálculo das novas posições dos centroides, onde a nova posição de cada centroide é obtida por meio da média aritmética das posições de cada objeto pertencente ao seu respectivo grupo. Por fim, o algoritmo verifica a convergência comparando se existe mudança entre a classificação anterior e a nova. Se existe diferença, o algoritmo retorna para o passo de classificação novamente, porém, se não existir diferença, significa que os dados que estão classificados não sofrerão mais mudanças e encerra o processo.

Dentre as vantagens do classificador K-médias, podemos citar: relativamente simples de implementar, garante convergência e facilidade de adaptação para novos exemplos, permitindo assim ser um classificador simples, porém eficiente.

Quanto as desvantagens, podemos citar o alto custo computacional para conjuntos de dados muito extensos, pelo fato de realizar o cálculo da distância Euclidiana de cada ponto x_i para cada k centroides em cada iteração do algoritmo. (22) Uma outra característica deste método é que diferentes agrupamentos podem ser obtidos dependendo das posições iniciais.

2.5 Análise de Componentes Principais (PCA)

Tipicamente, dados possuem um elevado número de observações. Cada observação possui uma certa quantidade de medidas, propriedades ou características, o que por sua vez, pode impossibilitar a visualização dessas observações, além de impor consideráveis limitações computacionais. Considerando a tendência de as medidas serem correlacionadas, é possível reduzir a dimensão dos dados por meio de técnicas de descorrelação a partir do método de Análise de Componentes Principais (PCA), permitindo a remoção de redundância. (23)

O PCA é uma transformação linear que utiliza autovetores ortogonais para gerar um espaço de menor dimensão, sendo assim, a essência das aplicações de PCA consiste na simplificação dos dados com mínima perda da dispersão geral, o que permite uma redução de dimensionalidade em que os dados serão representados. (24)

Os principais passos do PCA são: estimação da matriz de covariância dos dados, cálculo dos autovalores e autovetores associados, ordenação crescente dos autovalores (e autovetores), e uso dos primeiros autovetores como linha da matriz de transformação. (23) Observa-se que PCA implicitamente incorpora normalização por estandardização.

2.6 Interface de usuário

Uma interface de usuário – *user interface* (*UI*) – é um espaço onde os usuários interagem com uma máquina, seja ela um site, programa ou aplicativo. Esta pode incluir telas de exibição e periféricos como mouse e teclado, por exemplo. (25)

Entre os ramos derivados da interface de usuário, têm-se a interface gráfica de usuário – *graphical user interface* (*GUI*) – que se trata da interface onde o usuário interage com computadores ou *smartphones* por meio de ícones, menus e outros indicadores visuais ou gráficos. Essa tem, por objetivo, facilitar e ampliar o uso de tecnologias digitais. Com isso, as interfaces gráficas de usuário são um dos fatores mais importante que tornaram os computadores e tecnologias digitais mais acessíveis, facilitando a interação com estes sistemas. (26)

3 MATERIAIS E MÉTODOS

Neste capítulo, a metodologia, recursos e o planejamento experimental utilizados neste trabalho serão descritos. Além da descrição do desenvolvimento do programa computacional, abordamos também os principais métodos de análise dos grafos em questão.

3.1 Circuitos amplificadores eletrônicos

Com finalidade de obter uma certa variedade de circuitos amplificadores eletrônicos, optou-se por buscar diversos esquemas de amplificadores eletrônicos disponíveis na Internet e de livre acesso.

Considerando circuitos com funções de amplificação similares, foram obtidas 21 imagens de circuitos amplificadores que serão posteriormente representados por grafos correspondentes levando em conta a topologia e os componentes eletrônicos de cada circuito, de acordo com a metodologia apresentada na Seção 2.3 deste trabalho.

3.2 Programa computacional

Durante o processo inicial de obtenção dos grafos frente aos circuitos eletrônicos sem auxílio computacional (manualmente), notou-se que as várias etapas e complexidade dos circuitos contribuíam para maior probabilidade de erro humano. Portanto, para fins de auxiliar a obtenção dos grafos frente aos circuitos eletrônicos discutidos na Seção 3.1, propôs-se o desenvolvimento de um programa computacional que receba uma imagem de um esquemático eletrônico e permita a respectiva identificação de nós e arestas nomeados de acordo com a necessidade do usuário.

3.2.1 Interface proposta

Em vista de otimizar o uso do programa computacional, foi concebida uma interface simples e intuitiva, onde o usuário vê apenas quatro regiões, sendo elas: imagem do circuito eletrônico; criador de nós; criador de arestas; salvar e sair. Essas regiões estão dispostas no *layout* como mostra a Figura 14.

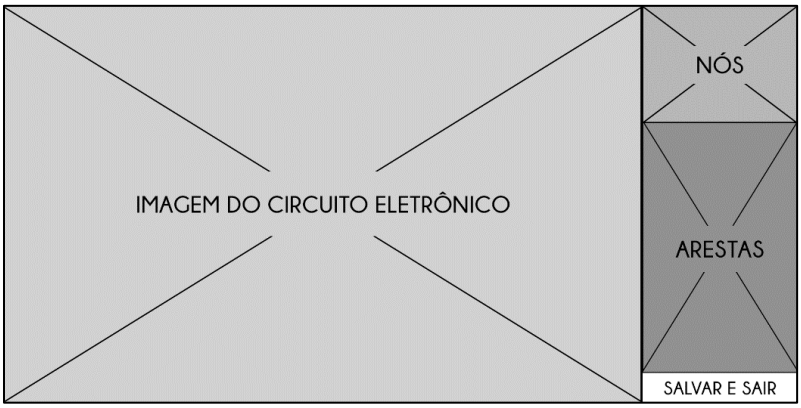


Figura 14 – Layout geral proposto para interface gráfica
Fonte: Elaborada pelo autor

Por meio da ferramenta "*guide"*, nativa do ambiente MATLAB®, é possível desenvolver todos *layouts* necessários para o programa.

3.2.1.1 Imagem do circuito eletrônico

Esta região é dedicada para exibir a imagem do circuito eletrônico e receber os cliques do *mouse* para cadastrar os nós respectivas coordenadas. Para isso, essa região deve mostrar inicialmente ao usuário um botão que permita o usuário importar a imagem desejada. Após isso, a imagem deve preencher o espaço dedicado respeitando as proporções da imagem. A Figura 15 apresenta o *layout* proposto para essa região.

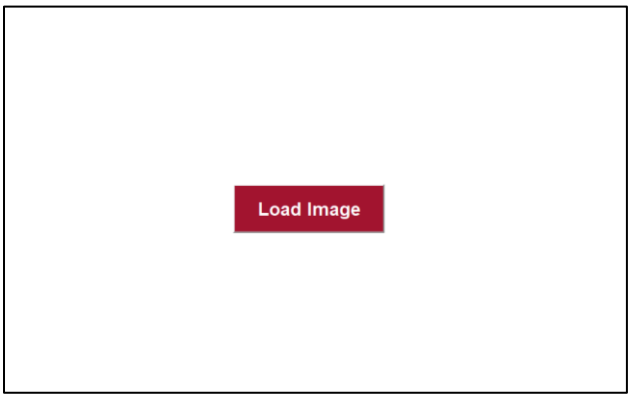


Figura 15 – Layout da região que exibe a imagem do circuito eletrônico
Fonte: Elaborada pelo autor

3.2.1.2 Nós

Para que se permita criar nós sobre uma imagem, é necessário que o *layout* da região de nós possua, ao menos, um botão para iniciar o processo de criação do nó e uma lista que mostre

os nós que já foram criados pelo usuário. Também é desejado um botão para remover o último nó criado. Pensando nisso, o *layout* resultante desta região está apresentado na Figura 16.

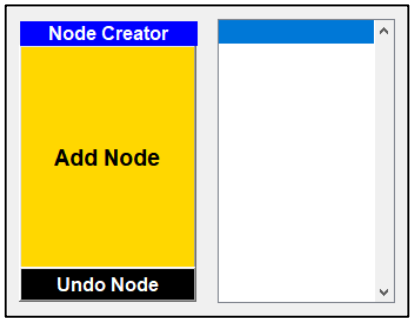


Figura 16 – Layout da região dos nós
Fonte: Elaborada pelo autor

3.2.1.3 Arestas

Para criar uma aresta, é necessário que já existam ao menos um vértice criado, portanto o *layout* proposto para essa região deve possuir dois campos que recebam as referências dos vértices fonte e de destino que receberão a ligação, assim como também um botão para criar essa aresta. Além destes, deve também possuir uma lista que apresente as arestas já criadas pelo usuário.

Para fins de melhorar a experiência do usuário, também é desejado um botão adicional para remover a última aresta criada. O *layout* proposto para essa região é representado pela Figura 17.

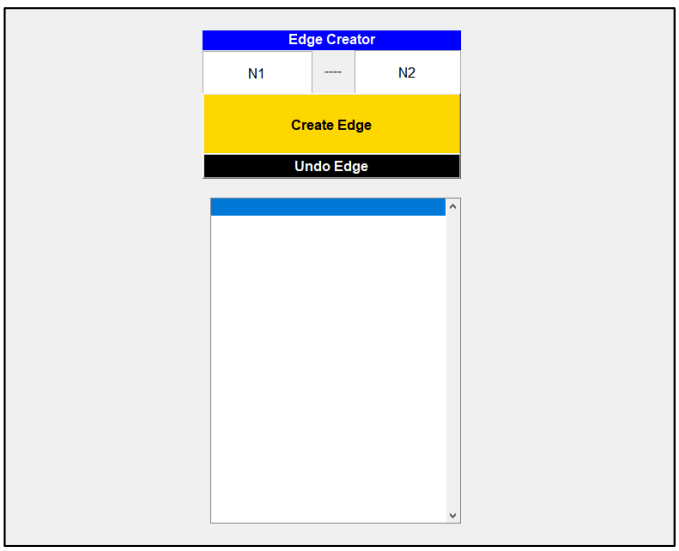


Figura 17 – Layout da região das arestas
Fonte: Elaborada pelo autor

3.2.1.4 Salvar e sair

Para esta região, é necessário apenas um botão para salvar e finalizar o processo. Sabendo disso, o *layout* proposto consiste em apenas este botão, como mostra a Figura 18.

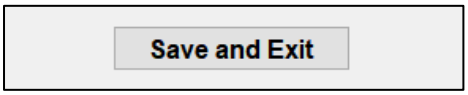


Figura 18 – Layout da região de salvar e sair
Fonte: Elaborada pelo autor

Por fim, a partir da união dos *layouts* apresentados, obtêm-se o *layout* final do programa, cujo qual é representado pela Figura 19.

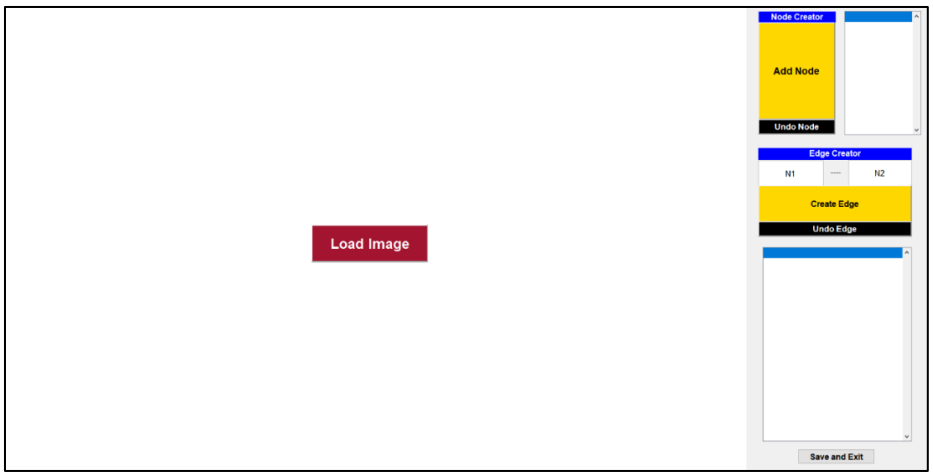


Figura 19 – Layout final do programa
Fonte: Elaborada pelo autor

Com o *layout* final definido, pode-se dar início à implementação das funcionalidades respectivas, definindo-se também a dinâmica geral do programa.

3.2.2 Importar e mostrar imagem

Existem várias formas de mostrar uma imagem ao usuário através do MATLAB®, porém, no caso especifico deste programa, além de simplesmente mostrar uma imagem que o usuário escolha, deve também permitir o usuário, quando solicitado, ter acesso a determinada coordenada dentro daquela imagem. Por isso, decidiu-se utilizar um plano cartesiano para exibir a imagem escolhida, pois permite o acesso a este tipo de informação.

A Figura 20 apresenta o fluxograma da função implementada que é chamada após o clique no botão "*Load Image*".

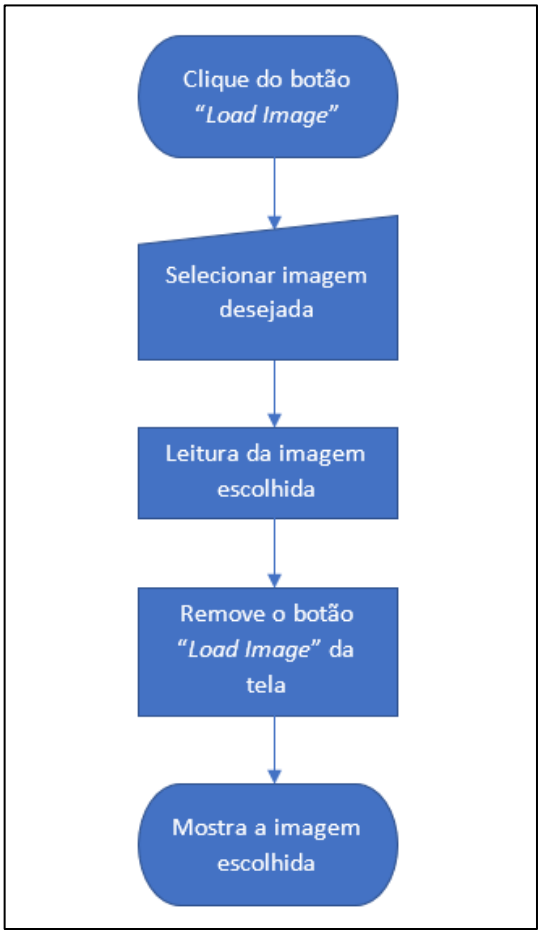


Figura 20 – Fluxograma da função para mostrar imagem
Fonte: Elaborada pelo autor

Inicialmente, após o clique do botão "*Load Image*", a respectiva função solicita ao usuário que selecione a imagem desejada. Após isso, o programa faz a leitura da imagem, desabilita a visibilidade do botão "*Load Image*" na tela da interface gráfica e, por fim, mostra a imagem na região delimitada no *layout* (Seção 3.2.1.1).

3.2.3 Cadastro de nós

Para cadastrar um nó, deve-se clicar no botão "*Add Node*" localizado na região denominada como "*Node Creator*" (Figura 16). Após isso, o programa permitirá um clique na posição desejada sob a imagem previamente importada. Com o clique realizado, o programa mostra na tela uma pequena janela que possui um campo de texto que permite o usuário nomear o nó que está sendo criado. A Figura 21 apresenta o *layout* dessa nova janela.

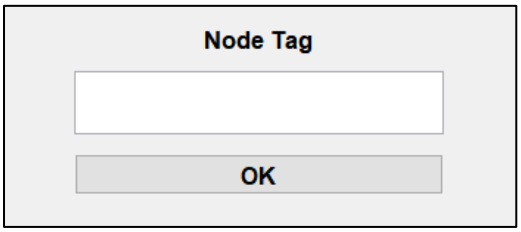


Figura 21 – Layout da janela de cadastro de novo nó
Fonte: Elaborada pelo autor

O fluxograma que representa o funcionamento da função de cadastro do nó pode ser observado na Figura 22.

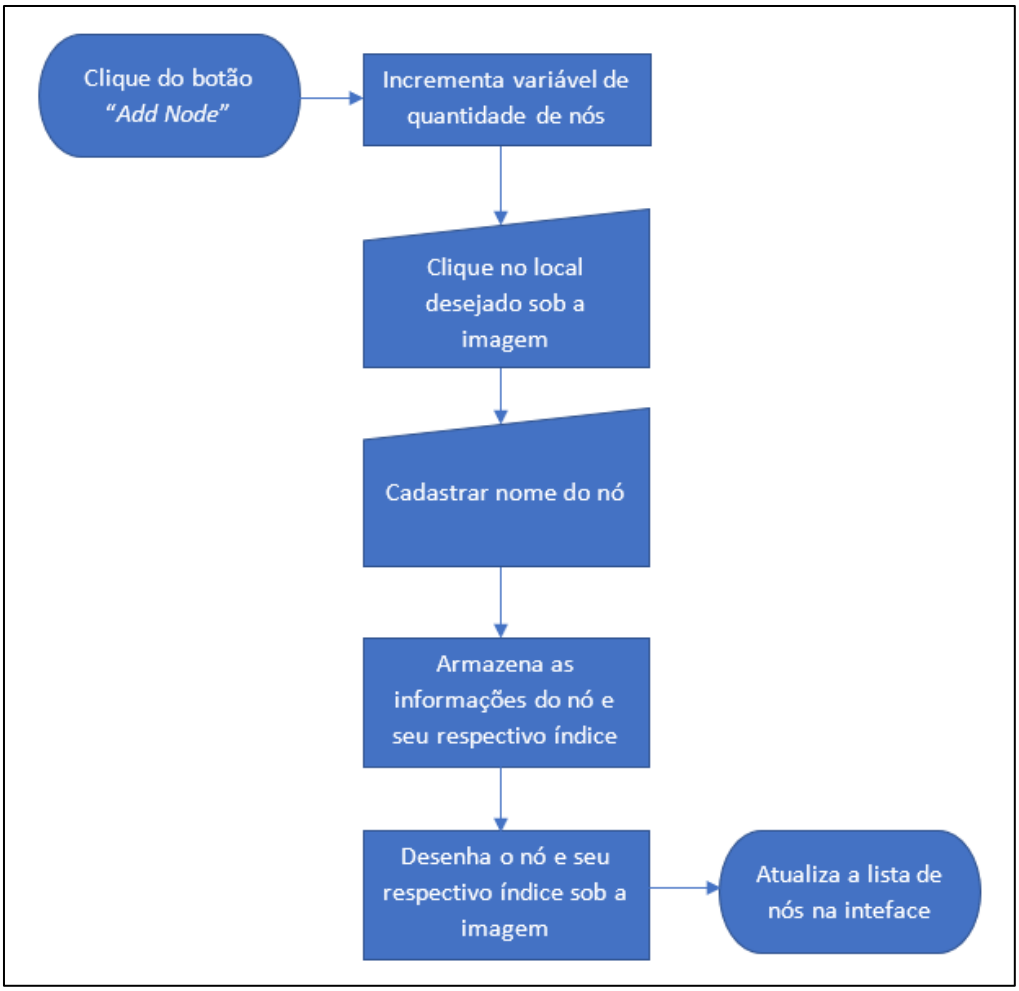


Figura 22 – Fluxograma da função de cadastro de novo nó
Fonte: Elaborada pelo autor

Após o clique no botão "*Add Node*" (Figura 16), o programa permite ao usuário um clique em um local desejado sob a imagem. Após isso, o programa aguarda o usuário completar o cadastro do nome do nó e então, ao término do cadastro, o programa atualiza a lista de nós e desenha sob a imagem o nó e seu respectivo índice de referência.

3.2.4 Cadastro de arestas

Após o cadastro de ao menos um nó, pode-se criar uma aresta. Para isso, o usuário deve preencher, nas caixas de texto localizadas na região das arestas (Figura 17), os campos responsáveis por receber *n1* e *n*2, que são os índices dos nós que serão ligados. Após, ao clicar no botão "*Create Edge*", o programa mostra na tela uma outra janela qual permite o usuário informar atributos sobre a aresta a ser criada. A Figura 23 apresenta o *layout* desta janela.

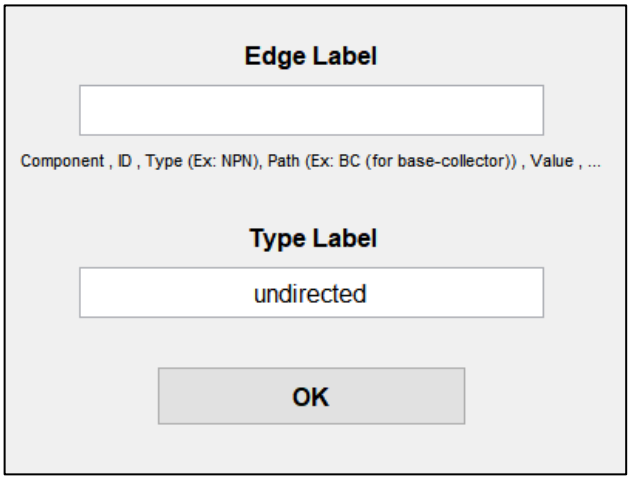


Figura 23 – Layout da janela de cadastro de nova aresta
Fonte: Elaborada pelo autor

Para o caso específico deste trabalho, o nome da aresta foi padronizado por uma lista separada por vírgula. Essa lista deve, obrigatoriamente, respeitar a seguinte ordem de informações: nome do componente eletrônico que a aresta representa e um índice numérico identificador daquele componente. Para componentes que necessitam de mais de uma aresta, pode-se preencher atributos extras como, no caso de um transistor por exemplo, qual o tipo do componente (FET, NPN ou PNP), qual o caminho dentro do componente a aresta representa, valores, etc.

Ainda na janela de cadastro da aresta, é também permitido escolher o tipo da aresta, sendo ela direcionada ou não-direcionada. Isso permite criar grafos mistos com arestas direcionadas e não-direcionadas de acordo com os diferentes componentes. A função responsável pelo cadastro da aresta está resumidamente representada pelo fluxograma apresentado na Figura 24.

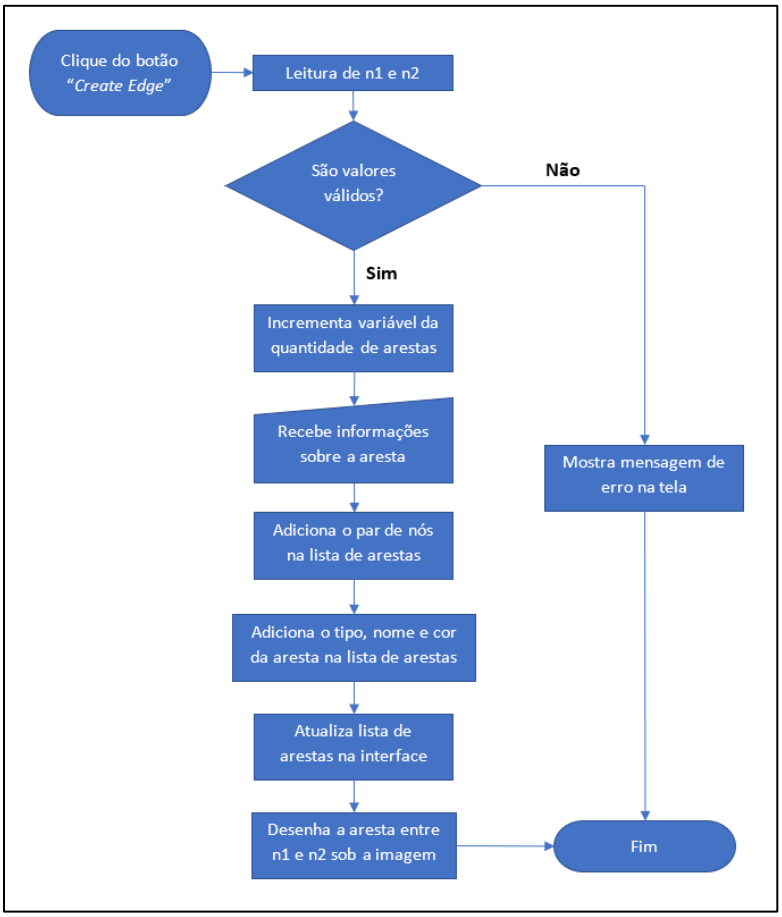


Figura 24 – Fluxograma da função de cadastro de arestas
Fonte: Elaborada pelo autor

Em vista de auxiliar a visualização do circuito eletrônico através de seu respectivo grafo, foram atribuídas cores específicas às arestas para a representação de cada determinado tipo de componente eletrônico. A Tabela 1 apresenta os tipos de componentes eletrônicos e sua respectiva cor.

Tabela 1 – Cores de arestas para representação dos tipos de componentes eletrônicos

Tipo	Cor	Representação	Código HEX
Resistor	Verde		#00ff00
Diodo	Cinza Médio		#7f7f7f
Capacitor	Vermelho		#ff0000
Indutor	Azul claro		#93b4cd
Fonte de corrente	Azul		#0000ff
TBJ (base-coletor)	Amarelo claro		#ffff95
TBJ (base-emissor)	Amarelo		#ffff00
TBJ (coletor-emissor)	Amarelo escuro		#b78b00
FET (*gate-drain*)	Marrom claro		#ff964b
FET (*gate-source*)	Marrom		#d2691e
FET (*drain-source*)	Marrom escuro		#b94b00
Outros	Preto		#000000

Fonte: Elaborada pelo autor

O tipo “outros” foi incluído com a finalidade de ampliar as áreas para aplicação do programa computacional, não sendo limitado apenas para imagens de circuitos eletrônicos. Sendo assim, caso a aresta não seja nomeada com o tipo de componente eletrônico já cadastrado no programa computacional, esta aresta será representada como “outros”.

3.2.5 Desfazer último nó ou aresta

Para o usuário, até o momento, o programa não possui funcionalidade de desfazer uma ação se necessário. Assim, foram incorporados alguns recursos que permitem o usuário desfazer ações.

Cada vez que um nó ou aresta é criado, vetores específicos recebem as informações como, por exemplo, coordenadas cartesianas, e as armazenam de forma sequencial. Desta forma, para desfazer a última ação, basta remover a última informação armazenada nesses vetores. Este processo pode ser observado pelos respectivos fluxogramas das funções dos botões “*Undo Node*” e “*Undo Edge*” como mostram as Figuras 25 e 26.

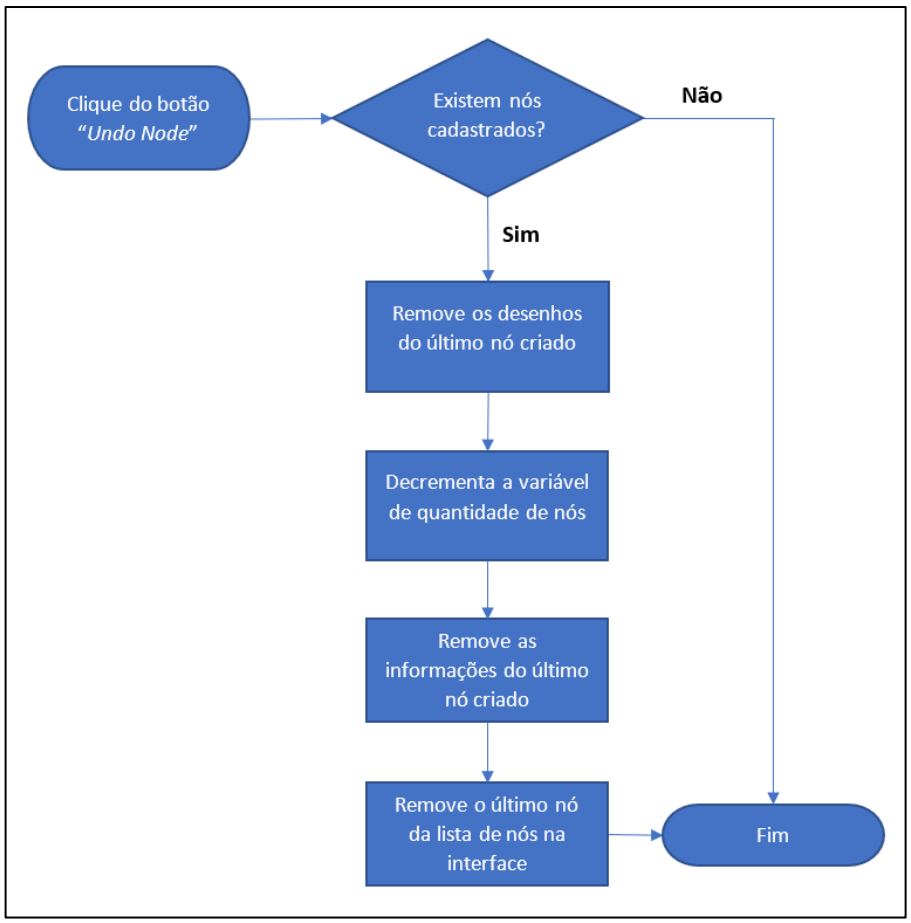


Figura 25 – Fluxograma da função para desfazer o último nó criado
Fonte: Elaborada pelo autor

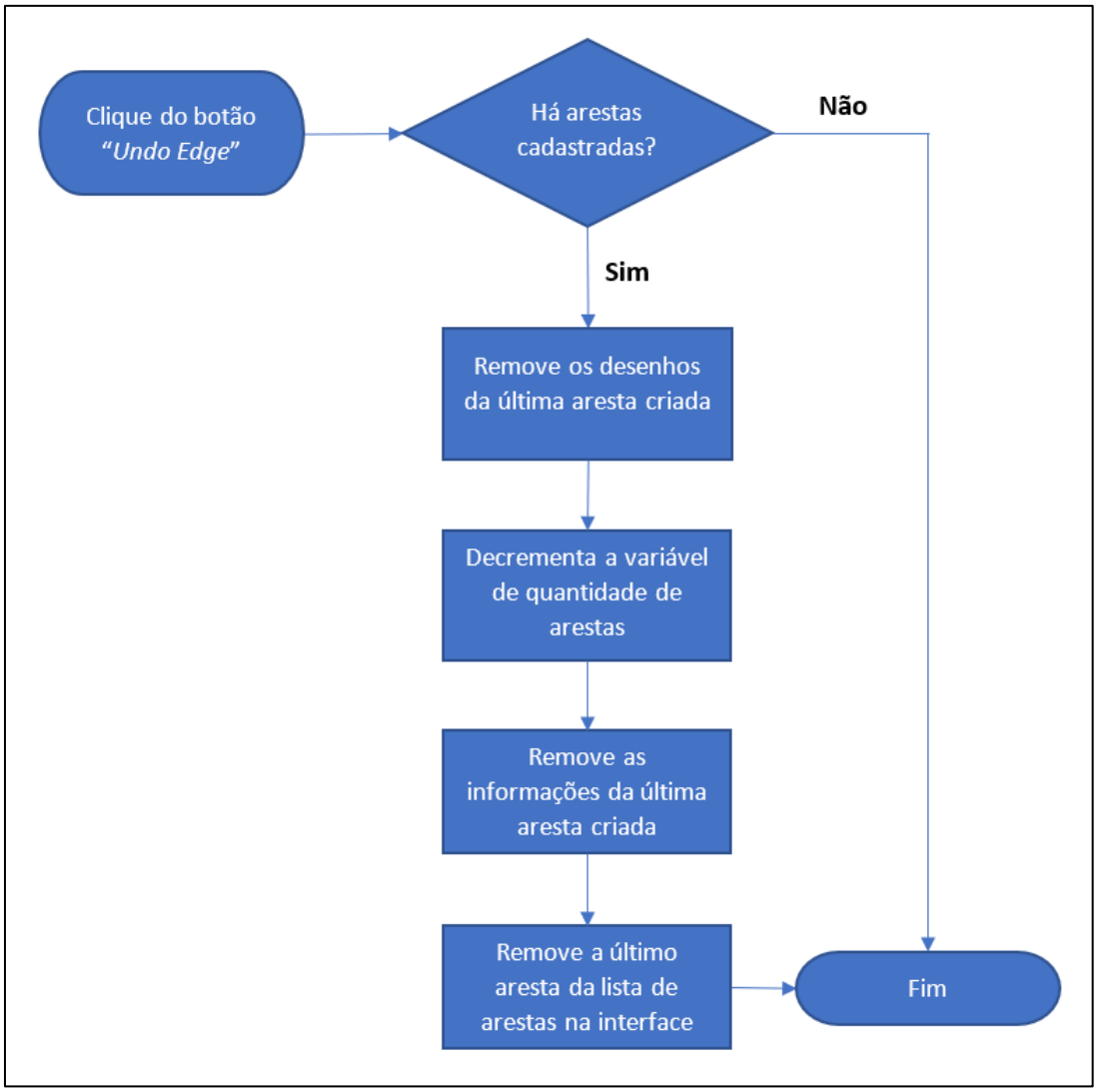


Figura 26 – Função para desfazer a última aresta criada
Fonte: Elaborada pelo autor

3.2.6 Salvar e sair

Ao término do cadastro dos nós e arestas, o usuário deve clicar no botão "*Save and exit*" para gerar os arquivos de representação do grafo. Este botão está vinculado à função representada pelo fluxograma na Figura 27, cuja qual formata e cria um documento separado por vírgula que contêm o grafo em forma de lista de adjacência, assim como também cria um arquivo padrão do MATLAB® (.m) contendo as variáveis "*edges*_list" e "*tag*_nodes" que armazenam respectivamente a lista de aresta e lista de nós. Este arquivo padrão será utilizado posteriormente para extrair medidas dos grafos dentro do ambiente do MATLAB®. Os arquivos são salvos em uma pasta com o nome da imagem importada, criada dentro da pasta chamada "Grafos".

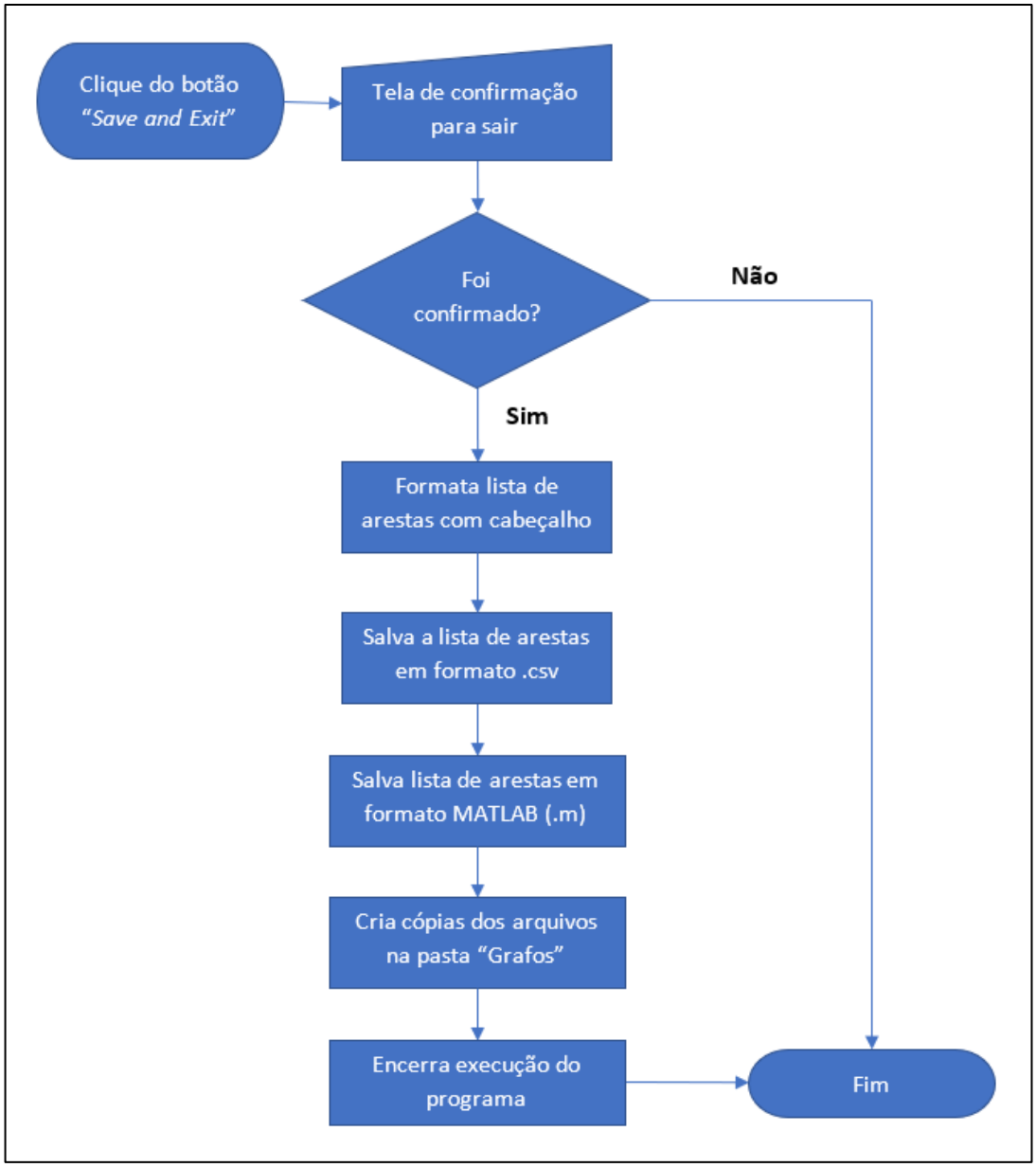


Figura 27 – Fluxograma da função do botão "*Save and exit*"
Fonte: Elaborada pelo autor

Por fim, têm-se implementadas todas as funções do programa. O código completo do programa está presente no APÊNDICE A deste trabalho.

3.3 Visualização dos grafos

Em vista de observar a capacidade de integração do programa com outras linguagens/plataformas, foi desenvolvido um algoritmo a partir da biblioteca "*iGraph*" (27) utilizando a linguagem de programação R (28), , para a leitura e visualização do grafo a partir

do arquivo gerado pelo programa desenvolvido anteriormente (Seção 3.2). A Figura 28 apresenta os passos executados por esse algoritmo.

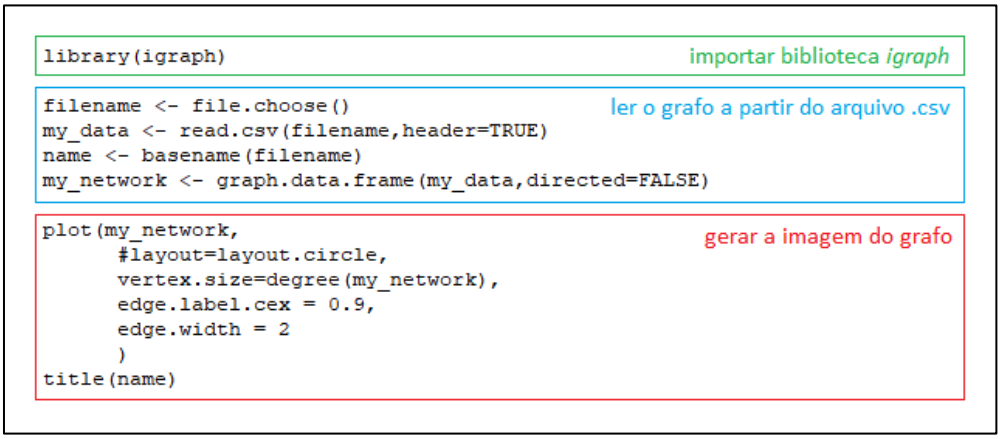


Figura 28 – Passos do algoritmo em R para leitura e visualização dos grafos.
Fonte: Elaborada pelo autor

De acordo com a Figura 28, pode-se perceber que são necessários apenas três passos para se obter a devida representação gráfica do grafo. Inicialmente se faz a importação da biblioteca de códigos "*iGraph*" e, então, faz-se a leitura do arquivo resultante do programa computacional e, por fim, obtêm-se a respectiva figura com a representação gráfica do grafo.

Para a renderização da imagem do grafo, foi utilizado, como exemplo, a medida de grau (Seção 2.2.5.1) para automatizar o tamanho dos nós.

3.4 Algoritmos das métricas

De forma a manter uma melhor compreensão e acesso às metodologias a serem empregadas, optou-se por implementar essas determinadas métricas.

Para fins de padronização, adotou-se que cada função de cada medida deve receber uma mesma informação padrão. No caso, optou-se pela matriz adjacência (Seção 2.2.4.1) do grafo desejado.

3.4.1 Grau

Com base na teoria apresentada na Seção 2.2.5.1, foi implementada a função "medidaGrau" (APÊNDICE B (a)), cuja qual recebe uma matriz adjacência de um grafo e retorna as medidas do grau médio e seu respectivo desvio padrão. O fluxograma referente a esta implementação pode ser observado na Figura 29.

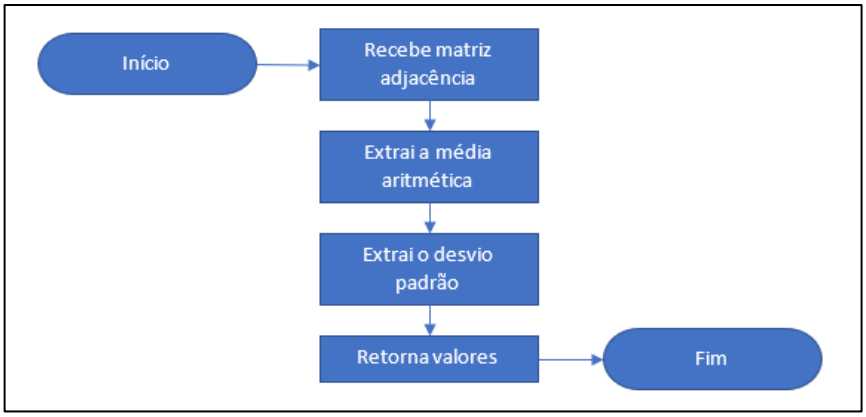


Figura 29 – Fluxograma da função “medidaGrau.”
Fonte: Elaborada pelo autor

3.4.2 Distância mínima

De acordo com a base teórica apresentada na Seção 2.2.5.2, a medida de distância mínima de um nó a todos os outros foi obtida a partir de um algoritmo de orientação de busca por largura *Breadth-first Search* (BFS) que recebe a matriz adjacência do grafo e um nó de referência, e retorna uma matriz com informações da distância relativa do nó de referência a todos os outros (representados por cada coluna). Este algoritmo também retorna (na segunda linha da matriz) qual é o nó antecessor (também chamado de nó pai) ao nó representado. A Figura 30 mostra essa matriz a partir de um grafo de exemplo.

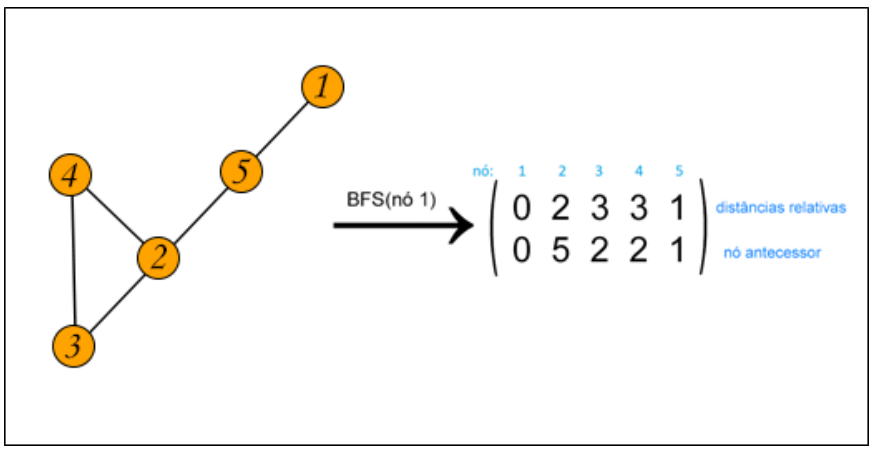


Figura 30 – Exemplo de matriz resultante a partir do algoritmo Busca em Largura
Fonte: Elaborada pelo autor

Com base no resultado desse algoritmo de busca em largura, é possível obter os valores de média e desvio padrão relativos às distancias mínimas para todo o grafo a partir da implementação da função “medidasDistancia” (APÊNDICE B (b)).

3.4.3 Grau hierárquico

Com base na teoria apresentada na Seção 2.2.5.3, realizou-se a implementação da função “grauHierarquicoVertice” (APÊNDICE B (c)), cuja qual recebe a matriz adjacência de um grafo e um valor numérico que representa um determinado nó e retorna um vetor que contêm todos os níveis de grau hierárquico que o determinado vértice possui, assim como sua devida vizinhança hierárquica. O processo para a obtenção desta medida está resumidamente representado pelo fluxograma contido na Figura 31.

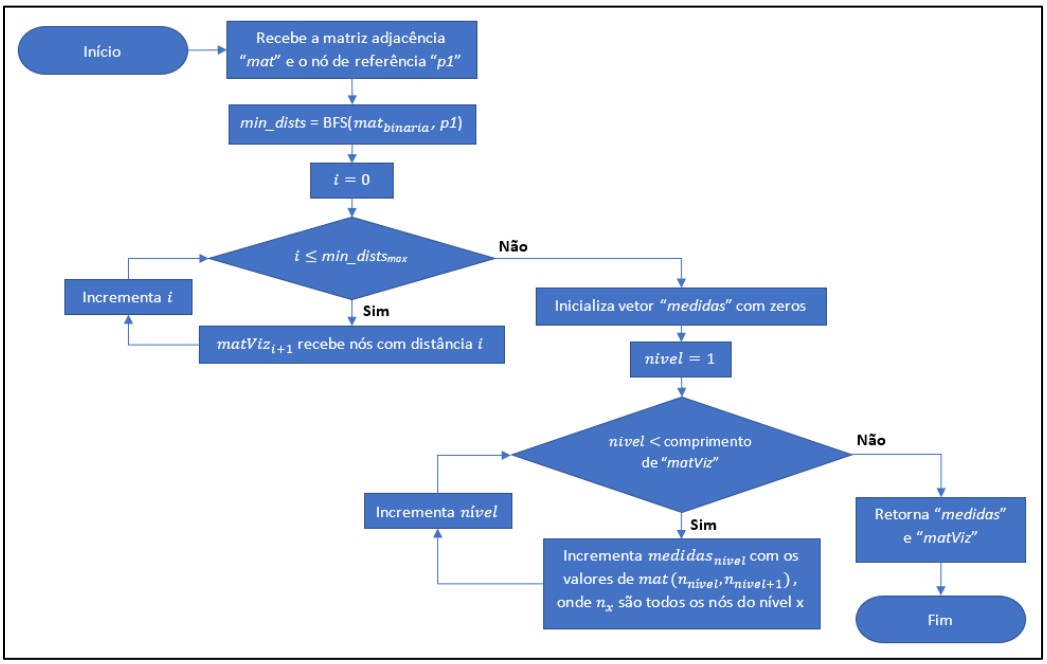


Figura 31 – Fluxograma da função “grauHierarquicoVertice.”
Fonte: Elaborada pelo autor

Através da Figura 31, percebe-se que a função inicialmente também utiliza-se do algoritmo de busca por largura, em que identifica as distâncias mínimas de todos os nós do grafo relativas ao nó “$p1$”. Com base nesta informação, organizam-se os nós na matriz “*matViz*” em níveis de acordo com suas distâncias relativas. Obtidos estes respectivos níveis, contabilizam-se e armazenam-se no número de arestas que ligam o nível atual com o seu respectivo antecessor (vetor “medidas”).

Como essa função apresenta valores relativos a um nó especifico, para a obtenção dos valores de todos os nós do grafo, é necessária a implementação da função “grauHierarquico” (APÊNDICE B (d)) para realizar e organizar as medidas em todos os nós. Tal função recebe a matriz adjacência de um grafo e retorna uma outra matriz em que suas linhas representam os níveis de hierarquia de cada nó (representado pelas colunas).

A partir da matriz obtida por essa nova função, é possível extrair medidas como média e desvio padrão de cada nível hierárquico percorrendo todas as colunas de uma mesma linha (nível desejado). Observa-se que a primeira linha desta matriz deve coincidir com as medidas do grau (Seção 2.2.5.1) de cada vértice do grafo.

3.4.4 *Matching index*

Conforme a teoria apresentada na Seção 2.2.5.4, implementou-se a função "matIndex" (APÊNDICE B (e)) que recebe uma matriz adjacência e retorna as métricas de valor médio e desvio padrão para um vetor de *matching index*. A Figura 32 apresenta o fluxograma que representa essa função.

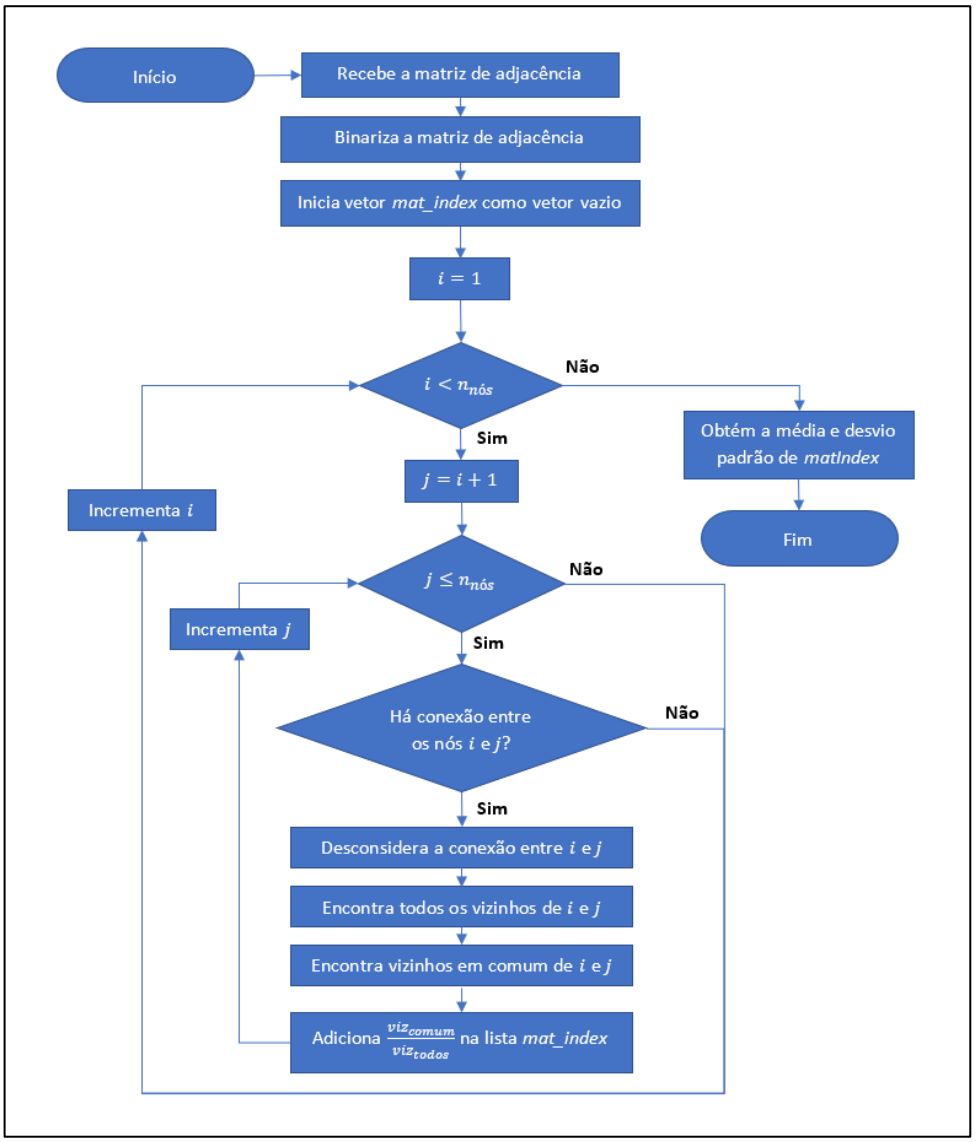


Figura 32 – Fluxograma da função para obtenção do *matching index.*
Fonte: Elaborada pelo autor

De acordo com a Figura 32, a função incialmente realiza a binarização de matriz de adjacência, desprezando a quantidade de ligações entre pares (visto que importa apenas a informação se há ou não conexão entre dois nós). Após isso, o algoritmo percorre todos os pares de nós e, para aqueles que existem conexões entre si, é feito o cálculo do índice de *matching index* com base na quantidade de nós vizinhos em comum em razão de todos os nós vizinhos ao par de nós analisado. Após a análise de todos os pares de nós e obtida o vetor "*mat_index*", obtém-se a respectiva média e o desvio padrão.

3.4.5 Coeficiente de aglomeração

Com base na base teórica apresentada na Seção 2.4.5.5, foi implementada a função "clusteringCoeficient" (APÊNDICE B (f)) que recebe uma matriz adjacência e retorna as métricas de valor médio e desvio padrão do coeficiente de aglomeração de todos os nós do grafo. O processo da função está representado pelo fluxograma apresentado na Figura 33.

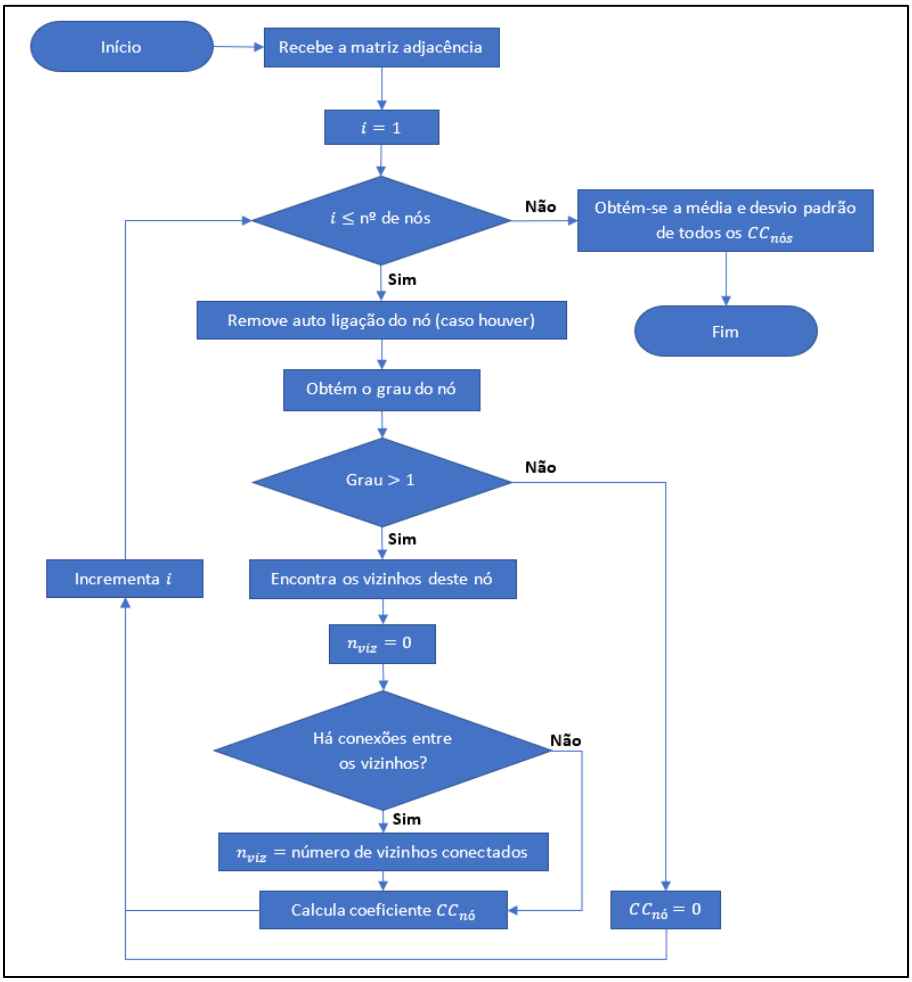


Figura 33 – Fluxograma da função para obtenção do coeficiente de aglomeração.
Fonte: Elaborada pelo autor

Pela análise da Figura 33, percebe-se que a função inicialmente percorre cada nó existente desconsiderando auto ligações e verificando se determinado nó possui mais de dois nós vizinhos (grau > 1). Caso sim, e se existirem conexões entre os vizinhos deste nó, é obtido seu respectivo coeficiente de aglomeração. Para nós com grau <= 1, atribui-se o coeficiente de aglomeração com valor zero. Após a análise de todos os nós do grafo e obtidos seus respectivos coeficientes de aglomeração, obtém-se a respectiva média e seu desvio padrão.

3.5 Algoritmo de classificação *k*-Médias

Para a classificação dos grafos, foi implementou-se a função "*k_means*" (APÊNDICE B (g)) de acordo com a base teórica apresentada na Seção 2.4. Esta função recebe uma matriz de dados, onde as linhas representam os atributos de cada ponto (representado pelas colunas), e um valor numérico "*k*" que representa a quantidade de centroides para a classificação. Após todo o processo, a função retorna as coordenadas finais dos centroides, assim como os grupos dos dados já classificados. Essa função está representada pelo fluxograma da Figura 34.

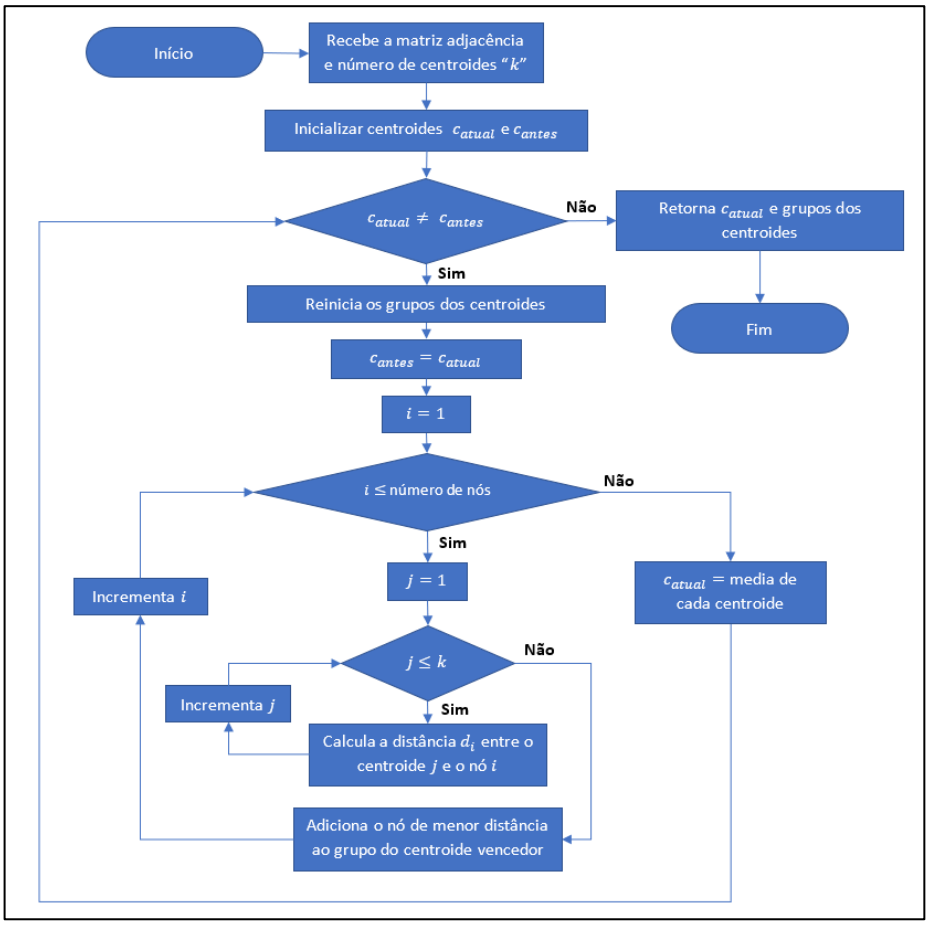


Figura 34 – Fluxograma do algoritmo de classificação *k*-Médias
Fonte: Elaborada pelo autor

De acordo com a Figura 34, o algoritmo inicializa (de maneira randômica) as coordenadas atuais e anteriores dos *k* centróides. Após isso, o algoritmo inicia um processo iterativo em que, sempre quando houver diferença entre as coordenadas atuais a sua respectiva antecessora, o algoritmo mensura a distância Euclidiana entre as coordenadas de cada dado e as coordenadas dos *k* centroides. Cada determinado dado é classificado com base no centroide mais próximo. Após percorrer todos os dados, o algoritmo obtém a nova posição dos centroides com base na média das coordenadas de todos os dados atribuídos à cada centroide. Quando não houver mais diferença significativa na posição dos centroides entre uma iteração à outra, o algoritmo se encerra e fornece os grupos de dados atribuídos a cada centroide.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta de forma sistêmica e discute os resultados do programa computacional, dos grafos e dos dados obtidos pela aplicação dos métodos anteriormente apresentados neste trabalho.

4.1 Resultados do programa computacional

Pela aplicação dos métodos citados no capítulo anterior, pôde-se obter com exatidão o *layout* proposto e as devidas funcionalidades do programa. Dentre essas funcionalidades, a Figura 35 apresenta um exemplo de importação de uma imagem para a área de trabalho do programa computacional.

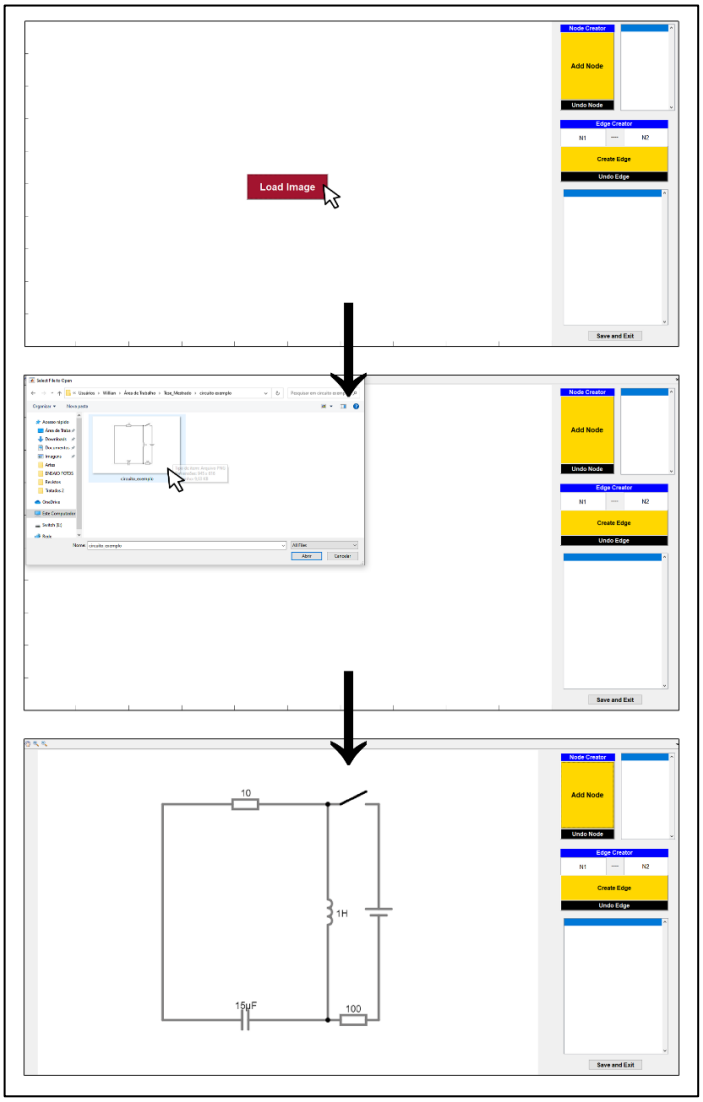


Figura 35 – Importação de imagem dentro do programa computacional
Fonte: Elaborada pelo autor

A partir da observação da Figura 35, pode-se perceber que são necessários apenas dois passos para importar uma imagem para dentro do programa computacional, assim como definido no Capítulo 3 deste trabalho.

Os resultados do programa para as etapas de criar e desfazer nós (ou vértices) sobre a imagem anteriormente importada podem ser observados na Figura 36.

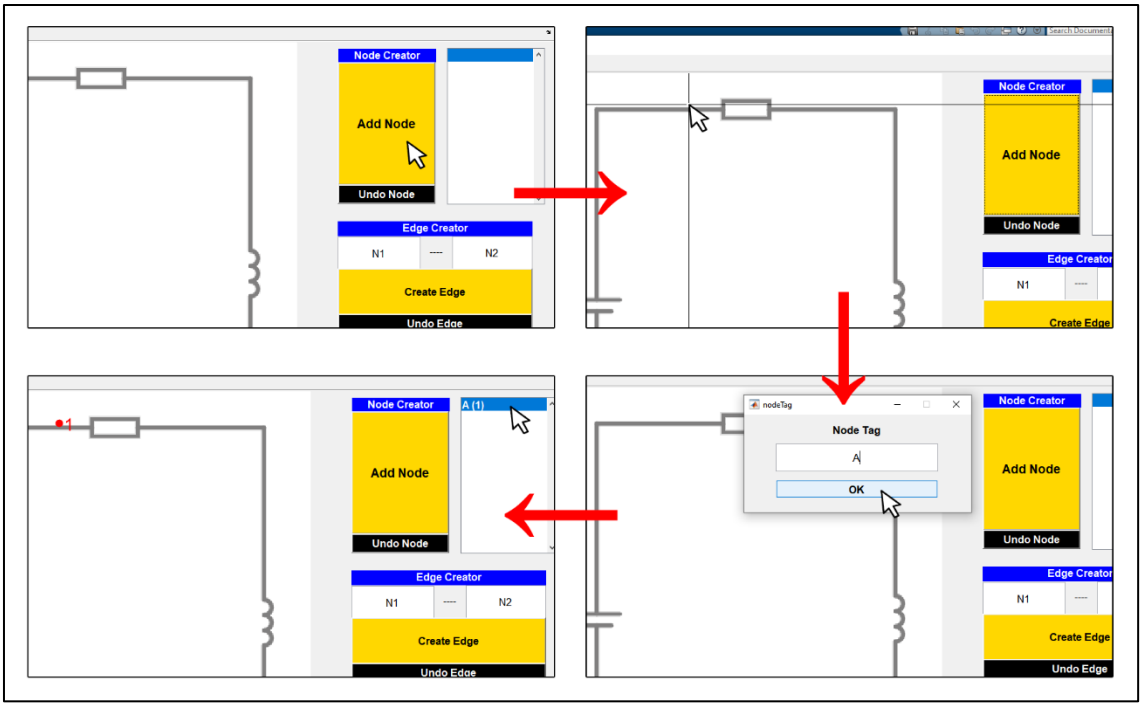


Figura 36 – Passos para criar nó sobre uma imagem previamente importada
Fonte: Elaborada pelo autor

Sempre que um nó é criado sobre a imagem, além do nome do nó definido pelo o usuário, o programa atribui a este um índice numérico de referência para o auxílio do usuário para o cadastro de arestas.

Com base na metodologia desenvolvida na Seção 3.2.4, foi possível realizar o cadastro de arestas entre dois pontos sobre a imagem previamente importada. Os resultados deste cadastro podem ser observados na Figura 37.

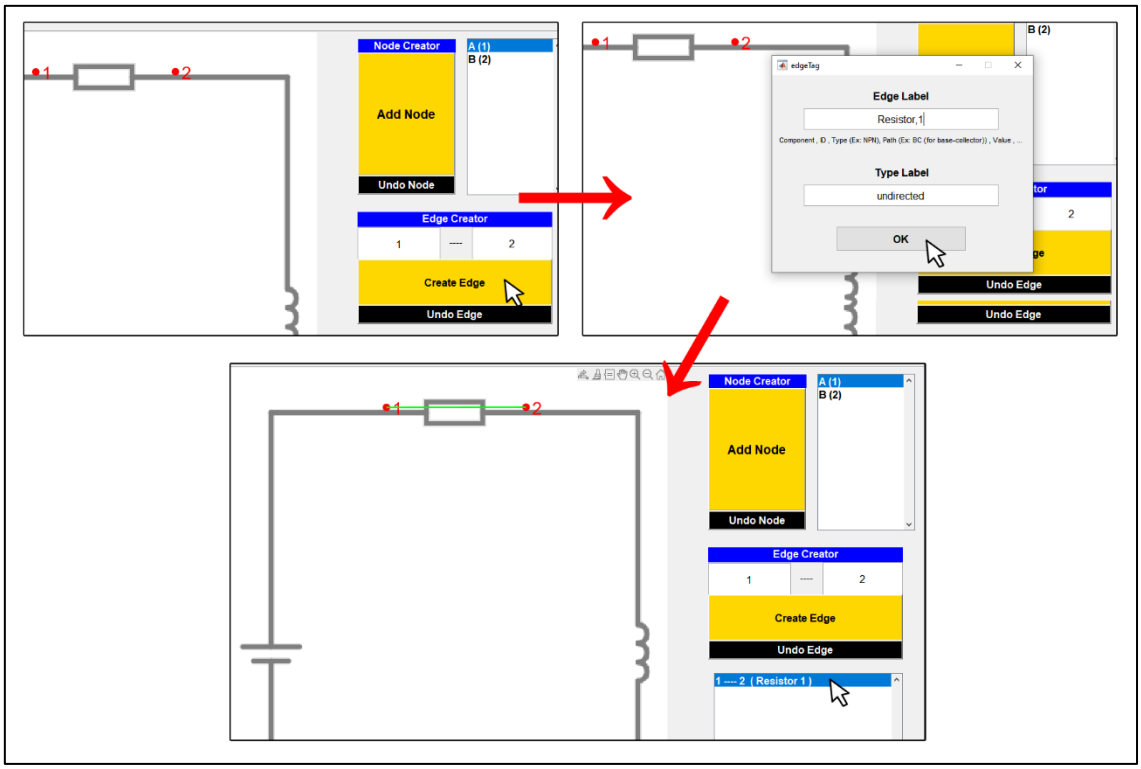


Figura 37 – Passos para criar aresta sobre uma imagem previamente importada
Fonte: Elaborada pelo autor

Ainda pela observação da Figura 37, é possível perceber que o programa efetua corretamente a colorização da aresta de acordo com o tipo do componente informado conforme a Tabela 1. Posteriormente, também foi observado o correto funcionamento das funções "desfazer nó" e "desfazer aresta" de acordo com a metodologia apresentada na Seção 3.2.5 deste trabalho.

Ao término do cadastro dos nós e arestas desejados, ao clicar no botão "*Save and Exit*", o programa efetuou corretamente os passos definidos na Seção 3.2.6, armazenando as informações pertinentes do grafo no arquivo em forma de lista separada por virgula (.csv) e o em seu respectivo arquivo padrão MATLAB® (.m).

4.1.1 Integração com outras linguagens e bibliotecas externas

Com base no método apresentado na Seção 3.2 deste trabalho, foram obtidas as representações visuais dos grafos a partir da linguagem de programação R e utilizando a biblioteca externa *iGraph* (27), como mostra a Figura 38.

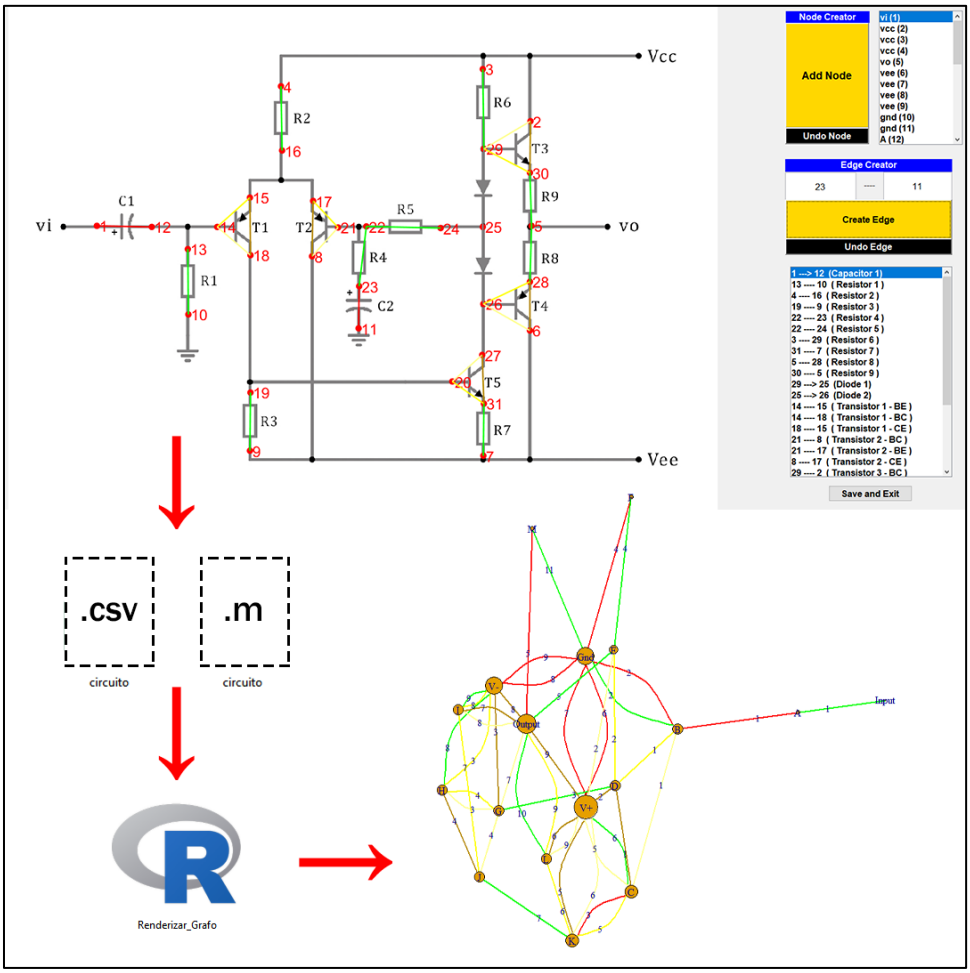


Figura 38 – Passos para obter grafo a partir de uma imagem
Fonte: Elaborada pelo autor

A partir da observação da Figura 38, percebe-se que foi possível a representação gráfica do grafo a partir da lista separada por vírgula (arquivo de extensão “.csv”) (APÊNDICE E) gerado pelo programa computacional desenvolvido, visto que também respeitou a distribuição de cores das arestas conforme pré-estabelecida na Tabela 1. Desta forma, é possível perceber a compatibilidade dos dados gerados a partir do programa computacional desenvolvido com outras bibliotecas e linguagens de programação.

Com o objetivo de simplificar a representação gráfica dos grafos, optou-se por filtrar as informações de arestas cadastradas e exibir apenas os índices dos componentes cadastrados nas arestas, visto que a cor já representa o tipo de cada componente eletrônico. Desta forma, é possível obter grafos mais objetivos, o que auxilia na compreensão geral do circuito representado. Essa otimização pode ser observada na Figura 39.

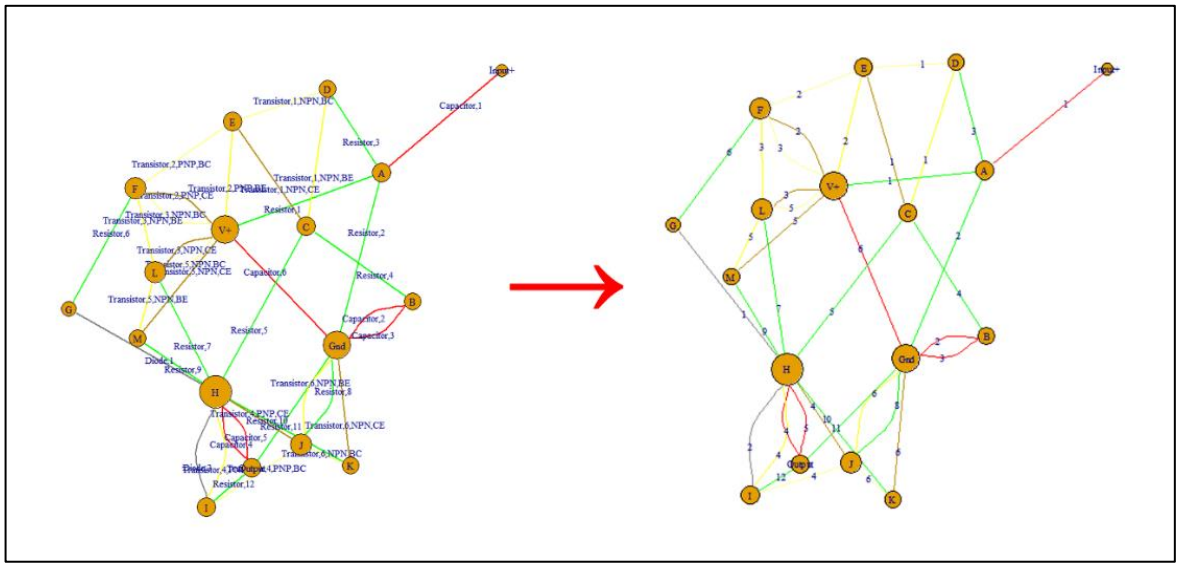


Figura 39 – Simplificação visual das informações das arestas sobre o grafo
Fonte: Elaborada pelo autor

Todas as figuras de grafos produzidas neste trabalho estão contidas no APÊNDICE D.

4.2 Representação gráfica das métricas dos grafos

Com base nas métricas desenvolvidas na Seção 3.4 e com a inclusão da métrica de número de arestas de cada grafo, obteve-se um banco com um total de quinze métricas às quais foram atribuídos índices numéricos de identificação, conforme mostra a Tabela 2.

Tabela 2 - Banco de medidas

Métrica	Variações	ID
Número de arestas	Valor absoluto	1
Grau	Média	2
Grau	Desvio padrão	3
Grau Hierárquico 1	Média	4
Grau Hierárquico 1	Desvio padrão	5
Grau Hierárquico 2	Média	6
Grau Hierárquico 2	Desvio padrão	7
Grau Hierárquico 3	Média	8
Grau Hierárquico 3	Desvio padrão	9
Distância mínima	Média	10
Distância mínima	Desvio padrão	11
Coeficiente de aglomeração	Média	12
Coeficiente de aglomeração	Desvio padrão	13
Matching index	Média	14
Matching index	Desvio padrão	15

Fonte: Elaborada pelo autor

Em vista de identificar possíveis relações, foram desenvolvidos vários gráficos de dispersão que relacionam cada par de métricas. Dentre os gráficos de dispersão resultantes, pôde-se observar determinados pares de medidas com altos índices de correlação, o que podem caracterizar medidas redundantes. A Figura 40 apresenta os gráficos de dispersão dos pares de medidas que apresentam os maiores valores de correlação ($C_{corr} > 0.9$).

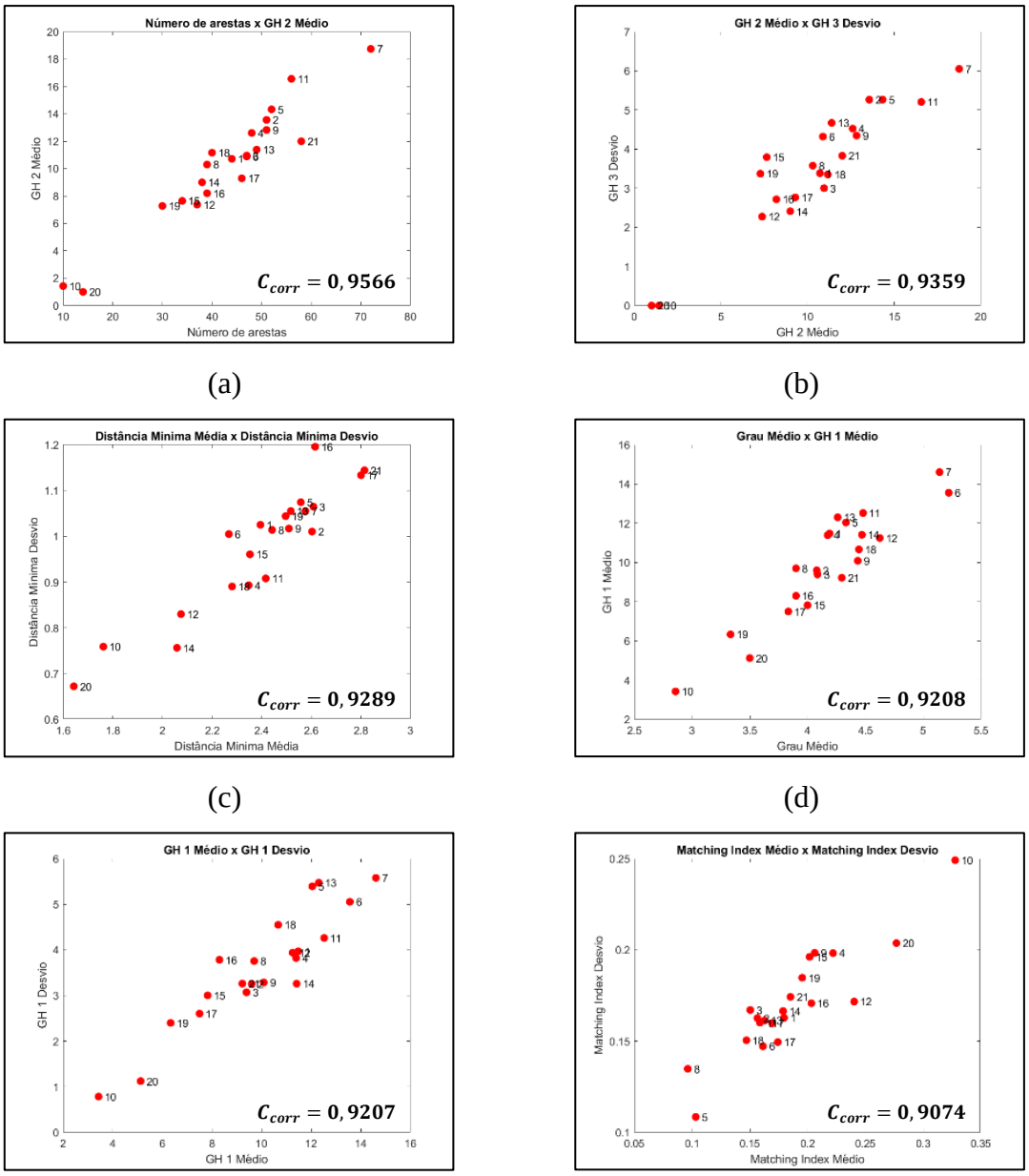


Figura 40 – Medidas com altos índices de correlação. (a) Número de arestas e grau hierárquico 2 médio (C_{corr}=0,9566). (b) Grau hierárquico 2 médio e desvio padrão do grau hierárquico 3 (C_{corr}=0,9359). (c) Distância mínima média e desvio padrão da distância mínima (C_{corr}=0,9289). (d) Grau médio e grau hierárquico 1 médio (C_{corr}=0,9208). (e) Grau hierárquico 1 médio e desvio padrão do grau hierárquico 1 (C_{corr}=0,9207). (f) *Matching index* médio e desvio padrão do *matching index* (C_{corr}=0,9074).

Fonte: Elaborada pelo autor

Outro comportamento que pode ser observado é a correlação negativa entre alguns pares de medidas com coeficientes de correlação $C_{corr} \approx -0{,}7$. A Figura 41 apresenta os gráficos de dispersão referentes a esses pares de medidas.

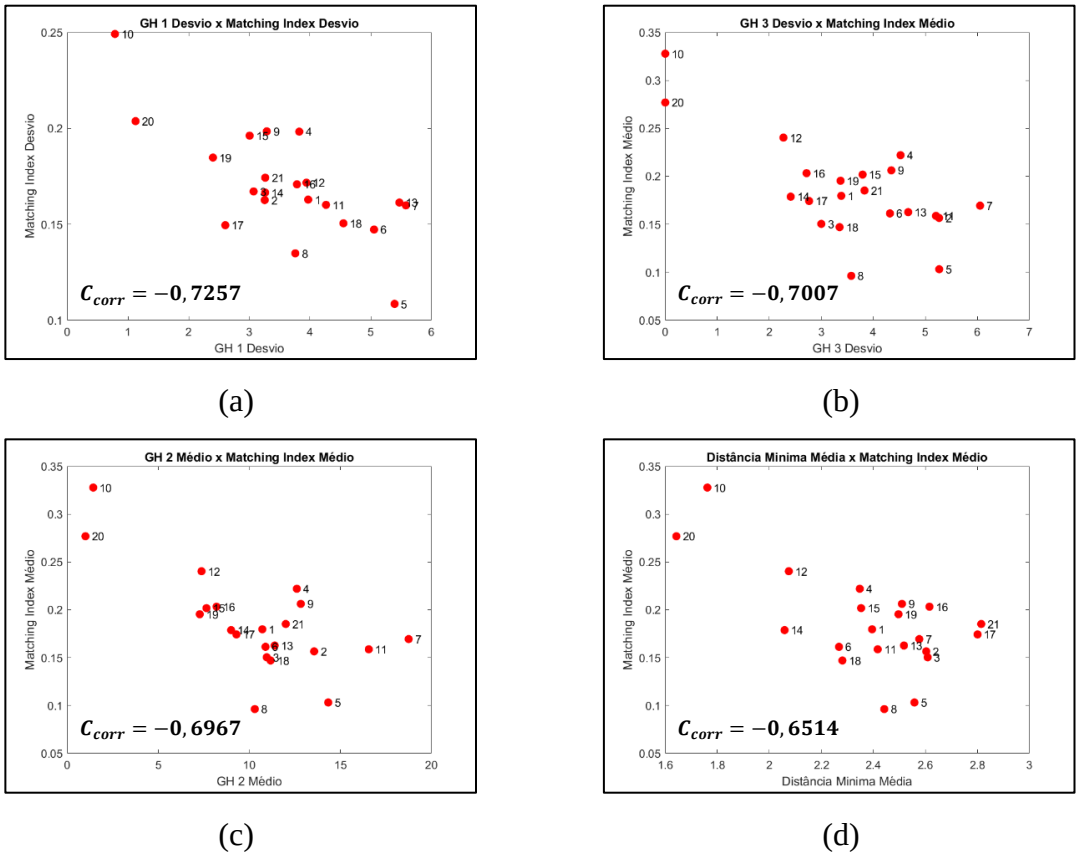


Figura 41 – Medidas que apresentam correlação negativa. (a) Desvio padrão do grau hierárquico 1 e desvio padrão do *matching* index (C_{corr}=-0,7257). (b) Desvio padrão do grau hierárquico 3 e *matching index* médio (C_{corr}=-0,7007). (c) Grau hierárquico 2 médio e *matching index* médio (C_{corr}=-0,6967). (d) Distância mínima média e *matching index* médio (C_{corr}=-0,6514)
Fonte: Elaborada pelo autor

Ao observar as métricas de grau hierárquico e distância mínima, percebe-se que ambas têm relação direta com o tamanho (complexidade) do circuito eletrônico. Portanto, essa correlação negativa entre essas medidas em relação à métrica de *matching index* condiz com a teoria, visto que o coeficiente de *matching index* mensura, de certa forma, a similaridade local entre as regiões dentro de um grafo (Seção 2.2.5.4), e quanto maior um grafo, existem mais possibilidades de as regiões adjacentes diferenciarem-se umas das outras.

Histogramas também foram obtidos com a finalidade de observar o comportamento geral de cada medida frente aos grafos. Dentre esses, foi possível observar que, para a métrica de distância mínima média, todos os grafos apresentaram valores dentro do intervalo de 1.5 a 3, conforme mostra a Figura 42.

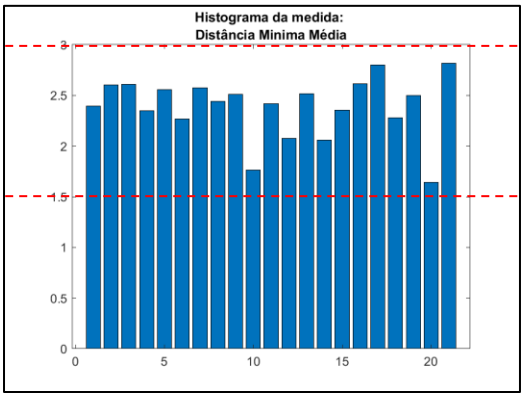
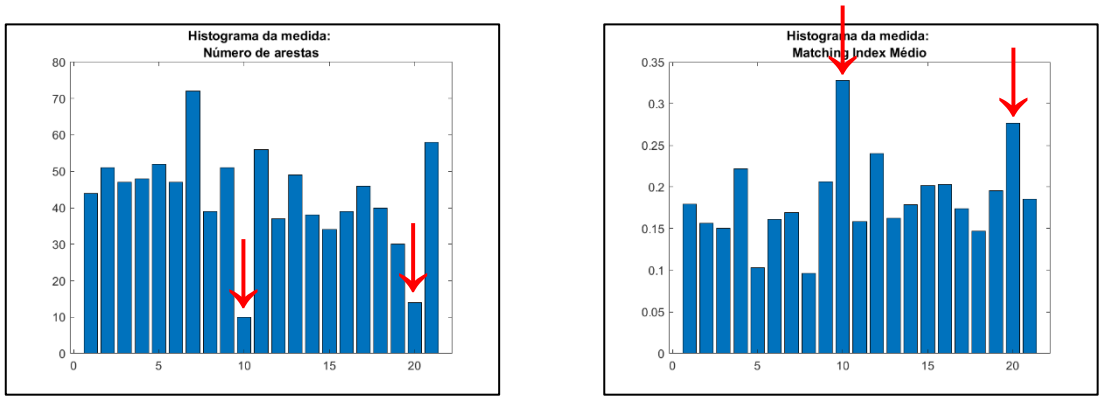


Figura 42 – Histograma da métrica Distância Mínima Média
Fonte: Elaborada pelo autor

Também foi observado que circuitos eletrônicos menores, ou seja, com menos componentes, apresentam valores maiores de *matching index*. Através da Figura 43, pode-se observar esse comportamento ao perceber que os grafos 10 e 20, que apresentam as menores quantidades de arestas, são aqueles que apresentam maiores valores de *matching index* médio.



(a) (b)

Figura 43 – Relação entre número de arestas e *matching index* médio. (a) Indicação dos grafos 10 e 20 que possuem os valores mais baixos no histograma da métrica do número de arestas. (b) Indicação dos grafos 10 e 20 que possuem os maiores valores no histograma da métrica *matching index* médio.
Fonte: Elaborada pelo autor

Tal resultado é justificável, pois como circuitos eletrônicos amplificadores possuem todos seus componentes conectados de alguma forma, quanto menor a quantidade de arestas (componentes/complexidade), maior a probabilidade de dois nós possuírem nós vizinhos em comum e, conforme apresentado na Equação 2.9, quanto maior a quantidade de vizinhos em comum, maior é o valor do coeficiente de *matching index*.

Com o objetivo de identificar possíveis agrupamentos de dados, optou-se por normalizar os dados e, posteriormente, classificá-los através do classificador *k*-Médias tendo como referência todas as quinze métricas. Devido à impossibilidade de representar dados em quinze

dimensões, utilizou-se o método PCA (Seção 2.5) para permitir a visualização aproximada (projeção) dos dados em duas dimensões. O resultado obtido através desse método pode ser observado na Figura 44.

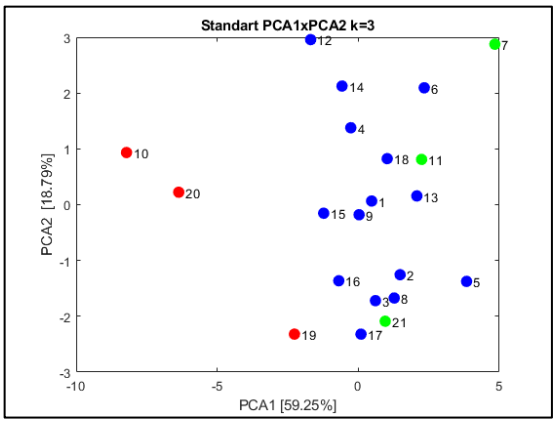


Figura 44 – Dados com visualização PCA
Fonte: Elaborada pelo autor

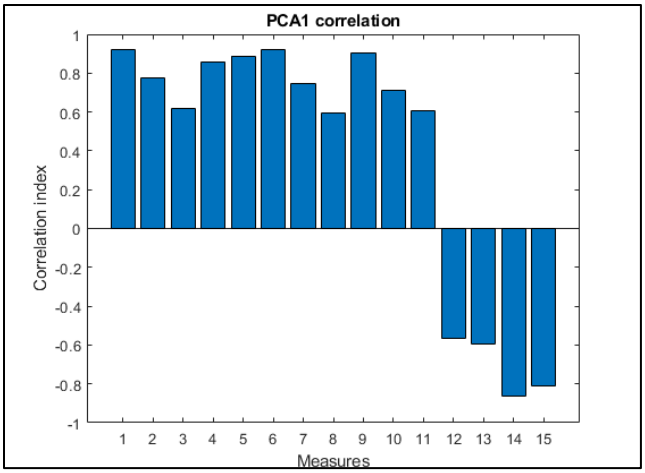
Ao observar o resultado apresentado na Figura 44, pode-se perceber, que através do método PCA, obteve-se uma variação de 59,25% ao longo do eixo PCA1 e 18,79% ao longo do eixo PCA2, portanto, mesmo após a redução de 15 para 2 dimensões, ainda sim foi possível representar aproximadamente 80% da variação dos dados.

Em busca de entender quais métricas são mais predominantes nestes eixos, extraiu-se o peso de cada métrica em cada eixo sobre o vetor de projeção. A Tabela 3 apresenta estes pesos em relação aos dois eixos PCA e a Figura 45 apresenta seus respectivos histogramas.

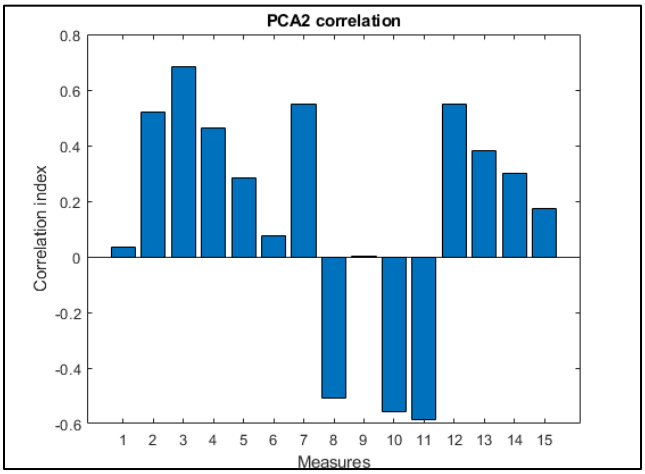
Tabela 3 - Índices de correlação das métricas em relação aos eixos PCA

Métrica [ID]	Correlação (PCA1)	Correlação (PCA2)
1	0,9188	0,0359
2	0,7754	0,5187
3	0,6192	0,6826
4	0,8598	0,4623
5	0,8892	0,2850
6	0,9223	0,0765
7	0,7447	0,5508
8	0,5957	-0,5079
9	0,9036	0,0034
10	0,7136	-0,5557
11	0,6066	-0,5855
12	-0,5676	0,5510
13	-0,5950	0,3799
14	-0,8609	0,3019
15	-0,8111	0,1721

Fonte: Elaborada pelo autor



(a)



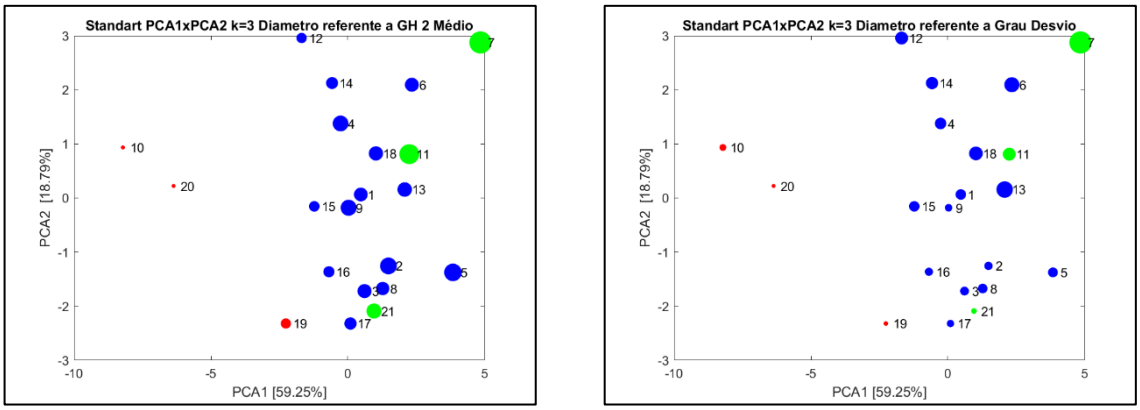
(b)

Figura 45 – Histogramas dos índices de correlação de cada métrica em relação aos eixos PCA. (a) Histograma referente aos índices de correlação de cada métrica em relação ao eixo PCA1. (b) Histograma referente aos índices de correlação de cada métrica em relação ao eixo PCA2
Fonte: Elaborada pelo autor

De acordo com a Tabela 3 e a Figura 45, destacam-se as métricas 6 e 3 com as maiores correlações respectivamente aos eixos PCA1 e PCA2. Portanto, com base na Tabela 2, o grau hierárquico 2 médio é a métrica de maior correlação ao eixo PCA1 e o desvio padrão do grau é a métrica de maior correlação ao eixo PCA2.

Conhecendo o grau de redundância da medida do grau hierárquico (Figura 40 (a)) e observando as métricas que apresentam coeficiente de correlação em relação ao eixo PCA1 superiores a 0.9, é possível perceber que ambas as medidas estão diretamente relacionadas com o tamanho dos circuitos, em outras palavras, pode-se dizer que o eixo PCA1 representa a complexidade dos circuitos representados.

Com a finalidade de representar essa relação no gráfico PCA, optou-se por variar o diâmetro de cada ponto de acordo com o valor de cada métrica. A Figura 46 mostra os gráficos PCA com o diâmetro dos pontos variável de acordo com as métricas: grau hierárquico 2 médio e grau desvio padrão.



(a) (b)

Figura 46 – Dados em visualização PCA com pontos de diâmetro variável de acordo com as medidas. (a) Visualização PCA com pontos de diâmetro variável de acordo com a métrica: grau hierárquico 2 médio. (b) Visualização PCA com pontos de diâmetro variável de acordo com a métrica: grau desvio padrão

Fonte: Elaborada pelo autor

A partir da Figura 45 (a), pode-se observar que o diâmetro dos pontos aumenta para valores maiores em PCA1, assim como também pela análise da Figura 45 (b), percebe-se que para maiores valores em PCA2, os pontos tendem a exibir um diâmetro maior.

Em relação à classificação pelo método *k*-Médias, não foi possível identificar uma separação apreciável dos dados através da projeção em duas dimensões.

No APÊNDICE C encontram-se tabelas com as principais informações relativas aos circuitos/grafos abordados, assim como os valores necessários para a obtenção dos gráficos e histogramas correntemente apresentados.

5 CONCLUSÕES

Este capítulo apresenta as considerações finais sobre o trabalho desenvolvido, assim como perspectivas de desenvolvimentos futuros.

5.1 Considerações finais

Tendo em base a crescente relevância do estudo de diversas áreas por meio de grafos, o uso de ferramentas que, de certa forma, simplifiquem a obtenção do grafo de determinado sistema ou processo se faz cada vez mais necessário, visto a complexidade relativa de cada objeto de estudo. De certa forma, no ramo da eletrônica, circuitos amplificadores eletrônicos tendem a apresentar diversas topologias, porém muitos com funções similares e, com base nas imagens dos esquemáticos desses circuitos, é possível realizar diversas análises por meio de grafos respectivos.

Neste trabalho, objetivou-se desenvolver um programa computacional para o auxílio na obtenção de grafos a partir de imagens dos esquemáticos de circuitos amplificadores eletrônicos, além de fornecer um estudo preliminar desses grafos com base em medidas topológicas de grafos e redes complexas. Para isso, tendo como base o ambiente de programação MATLAB®, foi projetado e desenvolvido um programa computacional com uma interface simples e intuitiva que permite o usuário importar uma imagem para uma área de trabalho pré-definida e, então, criar nós e arestas sobrepostos que, por fim, são representados em forma de uma lista de adjacência. Tal forma de representação, por sua generalidade e simplicidade, permite uma interface efetiva com outros módulos e programas de análise considerados no trabalho.

Com o objetivo de simplificar a visualização e interpretação dos grafos a partir dos circuitos amplificadores eletrônicos, partiu-se da metodologia do uso de cores nas arestas para representar cada tipo de componente eletrônico, o que tornou possível visualização efetiva da topologia do circuito por meio de seu respectivo grafo. Desta forma, comprovou-se que é possível desenvolver um programa computacional para o auxílio na obtenção de grafos a partir de imagens de circuitos amplificadores, podendo também ser válido para o auxílio na obtenção de um grafo baseado em qualquer tipo de sistema representado por uma imagem.

Para que se fosse possível analisar as características predominantes em cada grafo, basicamente seis métricas topológicas de grafos foram selecionadas e implementadas. Elas são:

- Número de arestas;

- Grau;
- Grau hierárquico;
- Distância mínima;
- Coeficiente de aglomeração;
- *Matching index*.

Pela utilização de três níveis hierárquicos do grau hierárquico e por meio de variações como média e desvio padrão, foram obtidas um total de quinze métricas utilizadas para obtenção de histogramas e organogramas, assim como classificação através do classificador não-supervisionado k-Médias e com visualização bidimensional no espaço PCA.

Diversos resultados interessantes foram obtidos. Dentre eles, foi observado que grafos que representam circuitos eletrônicos, tendem a apresentar distâncias mínimas médias próximas (intervalo entre 1,5 a 3). Também foi possível constatar que, para este tipo de circuito eletrônico, existe uma correlação negativa entre medidas diretamente relacionadas com o tamanho/complexidade do grafo quando relacionadas à medida de *matching index*, visto que quanto menor o circuito eletrônico e consequentemente seu grafo, maior é a probabilidade de os nós vizinhos possuírem conexões em comum, obtendo-se assim valores superiores de *matching index*.

Visando identificar as métricas mais relevantes desses grafos, foi mensurada a correlação de cada métrica com os dois eixos do espaço bidimensional PCA. Desta forma, constatou-se que as métricas que mensuram, de certa forma, a complexidade do circuito (especialmente o tamanho do grafo), foram as que apresentaram maior relevância com o primeiro eixo PCA, seguido pela medida de grau que apresentou uma correlação mais significativa ao segundo eixo PCA. Em relação a classificação dos grafos, não foi observada uma separação satisfatória dos dados pela projeção no espaço bidimensional PCA. Dentre as principais contribuições deste trabalho, enfatizamos:

- Desenvolvimento de um programa computacional que auxilia a obtenção de um grafo a partir de uma imagem.
- Apresenta implementações para o ambiente de programação MATLAB® de diversas métricas de grafos e do classificador não-supervisionado k-Médias.
- Revela as métricas mais relevantes para grafos baseados em circuitos amplificadores eletrônicos.

- Identificação de características típicas nos circuitos estudados, assim como seu inter-relacionamento e implicações.

5.2 Trabalhos futuros

Duas principais linhas de desenvolvimentos futuros podem ser delineadas a partir dos conceitos e resultados apresentados neste trabalho.

A primeira linha relaciona-se à complementação do *software* de representação e análise dos grafos, incluindo extensão para outros ambientes de programação e incorporação de medidas adicionais para caracterização das propriedades dos grafos obtidos, incluindo interfaces para bases de dados com sistemas de *query*.

A segunda linha de desenvolvimento refere-se ao estudo de outros tipos de amplificadores ou circuitos eletrônicos, incluindo-se aí circuitos integrados em vários níveis de integração, assim como circuitos digitais além dos circuitos analógicos correntemente abordados. Além disso, seria interessante complementar a análise dos circuitos considerados através do uso de medidas topológicas adicionais como outros métodos de reconhecimento de padrões. Uma outra perspectiva interessante seria obter a simulação da dinâmica dos circuitos e respectivo relacionamento com as respectivas características topológicas.

REFERÊNCIAS

1 COSTA, L. da F. *et al* Analyzing and modeling real world phenomena with complex networks a survey of applications. **Advances in Physics**, v. 60, n. 3, p. 329-412, 2011.

2 COSTA, L. da F. *et al.* Characterization of complex networks: a survey of measurements. **Advances in Physics**, v. 56, n. 1, p. 167-242, 2007.

3 TOSCANO, L; STELLA, S; MILOTTI, E. Using graph theory for automated electric circuit solving. **European Journal of Physics**, v. 36, n. 3, p. 035015, 2015.

4 EJIOFOR, O. S; SILVER, A. C. Design and construction of A 300 watt audio amplifier. **International Journal of Engineering and Management Research**, v. 5, n. 6, p. 63-67, 2015.

5 KHADKA, S. K. **Operational amplifier circuits and dynamics**. 2015. Disponível em: https://www.researchgate.net/publication/305730473_Operational_Amplifier_Circuits_and_Dynamics. Acesso em: 15 mar. 2022

6 CANCHO, R. F. JANSSEN, C. SOLÉ R. V. Topology of technology graphs small world patterns in electronic circuits. **Physical Review E**, v. 64, n. 4, p. 046119, 2001.

7 BORUAH, C.; GOGOI, K.; CHUTIA, C. Analysis of some electrical circuits with the help of graph theory using network equilibrium equations. **International Journal of Innovative Research in Science, Engineering and Technology**, v. 6, n. 1, p. 944-953, 2017.

8 LIMA, K. U. A. **Implementação e análise de um circuito amplificador aplicado na aquisição de sinais eletrofisiológicos de um inseto**. 2016. 104p. Trabalho de Conclusão de Curso (Engenharia Elétrica) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

9 ELETRONICS TUTORIALS. **Introduction to the amplifier**. 2021. Disponível em: https://www.electronics-tutorials.ws/amplifier/amp_1.html#:~:text=An%20amplifier%20is%20an%20electronic,version%20of%20its%20input%20signal. Acesso em: 23 de mar. 2022.

10 OLIVEIRA, T. Como funcionam os amplificadores de som? **Eletrônica geral**, 2021. Disponível em: https://eltgeral.com.br/como-funcionam-os-amplificadores-de-som/. Acesso em: 23 de mar. de 2022.

11 ELPROCUS. **Commom emitter amplifier circuit working & its characteristics**. Disponível em: https://www.elprocus.com/common-emitter-amplifier-circuit-working/. Acesso em: 23 mar. 2022.

12 CADERNO DE LABORATÓRIO. **Emissor comum sem desacoplamento**. Disponível em: https://cadernodelaboratorio.com.br/emissor-comum-sem-desacoplamento/. Acesso em: 23 de mar. de 2022.

13 ELPROCUS. **What is a push-pull amplifier**: circuit diagram and its working principle. Disponível em: https://www.elprocus.com/what-is-a-push-pull-amplifier-circuit-diagram-and-its-working-principle/. Acesso em: 25 abr. 2022.

14 ELECTRICAL4U. **Differencial amplifiers**: Gain, OP Amp & BJT circuit. 2021. Disponível em: https://www.electrical4u.com/differential-amplifier/. Acesso em: 2 jun. 2022.

15 CLASS B AMPLIFIER. **The Amazing world of electronics**. 2021. Disponível em: https://elamazing.com/2021/05/24/class-b-amplifier/. Acesso em: 12 jun. 2022.

16 BIGGS, N. L.; LLOYD, E. K.; WILSON, R. J. **Graph theory 1736-1936**. Oxford: Clarendon Press, 1976.

17 SINGH, H.; SHARMA, R. Role of adjacency matrix & adjacency list in graph theory. **International Journal of Computers & Technology**, v. 3, n. 1, p. 179-183, 2012.

18 COSTA, L. da F. Coincidence complex networks. **Journal of Physics**: complexity, v. 3, n. 1, p. 015012, 2022.

19 COSTA, L. da F. **What is a complex network? (CDT-2)**. 2018. Disponível em: https://www.researchgate.net/publication/324312765_What_is_a_Complex_Network_CDT-2. Acesso em: 11 mar. 2022.

20 CAJIC, D. **Edge lists visually explained. level up coding 2020**. Disponível em: https://level up.gitconnected.com/edge-lists-visually-explained-51336ae662a8. Acesso em: 11 mar. 2022.

21 COSTA, L. da F.; SILVA, F. N. Hierarchical characterization of complex networks. **Journal of Statistical Physics**, v. 125, p. 845-876, 2018. DOI: 10.1007/s10955-006-9130-y.

22 PÉREZ-ORTEGA, J. *et al*. The K-means algorithm evolution. In: SUD, K.; ERDOGMUS, P.; KADRY, S. (ed.). **Introduction to data science and machine learning**. Londres: Intechopen, 2020. DOI: 10.5772/intechopen.77469.

23 COSTA, L. da F. **A compact guide to PCA (CDT-47)**. 2020. Disponível em: https://www.researchgate.net/publication/346656784_A_Compact_Guide_to_PCA_CDT-47. Acesso em: 3 maio 2022.

24 GEWERS, F. L. *et al*. Principal component analysis: a natural approach to data exploration. **ACM Computing Surveys**, v. 54, n. 4, p. 1-34, 2021.

25 HASSUIKE, L. **O que é interface do usuário (UI):** definição, elementos e dicas. 2021. Disponível em: https://snov.io/glossary/br/interface-do-usuario-ui/#:~:text=reten%C3%A7%C3%A3o%20de%20usu%C3%A1rios.-,O%20que%%2020%C3%A9%20uma%20interface%20de%20usu%C3%A1rio%3F,de%20uma%20%C3%A1rea%20de%20trabalho. Acesso em: 23 abr. 2022.

26 STOLTZFUS. J. **Whats does graphical user interface (GUI) mean?**. 2021. Disponível em: https://www.techopedia.com/definition/5435/graphical-user-interface-gui. Acesso em: 25 abr. 2022.

27 CSARDI, G; NEPUSZ, T. The igraph software package for complex network research. **InterJournal, Complex Systems**, v. 1695, n. 5, p. 1-9, 2006. Disponível em: http://static1.squarespace.com/static/5b68a4e4a2772c2a206180a1/t/5cd1e3cbb208fc26c99de080/1557259212150/c1602a3c126ba822d0bc4293371c.pdf. Acesso em: 14 jun. 2022.

28 R CORE TEAM. R: A language and environment for statistical computing. **R Foundation for Statistical Computing**, Vienna, Austria, 2003. Disponível em: http://www.R-project.org/. Acesso em 29 set. 2022.

APÊNDICE A – Código MATLAB® do programa computacional

```
function varargout = cadastroGrafo(varargin)
%CADASTROGRAFO M-file for cadastroGrafo.fig
%      CADASTROGRAFO, by itself, creates a new CADASTROGRAFO or raises the existing
%      singleton*.
%
%      H = CADASTROGRAFO returns the handle to a new CADASTROGRAFO or the handle
to
%      the existing singleton*.
%
%      CADASTROGRAFO('Property','Value',...) creates a new CADASTROGRAFO using the
%      given property value pairs. Unrecognized properties are passed via
%      varargin to cadastroGrafo_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%                                          CADASTROGRAFO('CALLBACK')          and
CADASTROGRAFO('CALLBACK',hObject,...) call the
%      local function named CALLBACK in CADASTROGRAFO.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help cadastroGrafo

% Last Modified by GUIDE v2.5 19-Mar-2019 12:42:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn', @cadastroGrafo_OpeningFcn, ...
                'gui_OutputFcn',  @cadastroGrafo_OutputFcn, ...
                'gui_LayoutFcn',  [], ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
  gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before cadastroGrafo is made visible.
function cadastroGrafo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for cadastroGrafo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
clear all
clc
global i_nodes i_edges
i_nodes = 0;
```

```
i_edges = 0;



% UIWAIT makes cadastroGrafo wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = cadastroGrafo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in add_node.
function add_node_Callback(hObject, eventdata, handles)
% hObject    handle to add_node (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global i_nodes c_nodes tag_nodes plot_nodes nodes_listbox
i_nodes = i_nodes+1;
c_nodes(i_nodes,:) = ginput(1);
uiwait(nodeTag);
nodes_listbox{i_nodes,1} = [tag_nodes{i_nodes,1},' (',num2str(i_nodes),')'];
plot_nodes{i_nodes,1} = plot(c_nodes(i_nodes,1),c_nodes(i_nodes,2),'.r','markersize',30);
plot_nodes{i_nodes,2} = text(c_nodes(i_nodes,1)+5,c_nodes(i_nodes,2),num2str(i_nodes), ...
    'FontSize',20,'Color',[1 0 0],'HorizontalAlignment','left');
set(handles.listbox_node,'String',nodes_listbox,'FontSize',12);
```

```
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in loadImage_button.
function loadImage_button_Callback(hObject, eventdata, handles)
% hObject    handle to loadImage_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global File_Name
[File_Name, Path_Name] = uigetfile('*.jpg');
```

```
img = imread([Path_Name,File_Name]);
set(handles.loadImage_button,'Visible','off');
imshow(img,'Parent',handles.axes1);
hold on


% --- Executes on button press in undoEdge_button.
function undoEdge_button_Callback(hObject, eventdata, handles) %#ok<INUSL>
% hObject    handle to undoEdge_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global edges_list i_edges list_listbox plot_edges %#ok<GVMIS>
if i_edges %Verifica se não está zerado o número de edges
   set(plot_edges{i_edges},'Visible','off')
   i_edges = i_edges-1;
   edges_list = edges_list(1:end-1,:);
   plot_edges = plot_edges(1:end-1,:);
   list_listbox = list_listbox(1:end-1,:);
   set(handles.listbox1,'String',list_listbox,'FontSize',12);
end


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function edge_n1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edge_n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function edge_n2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edge_n2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in undoNode_button.
function undoNode_button_Callback(hObject, eventdata, handles) %#ok<INUSL>
% hObject    handle to undoNode_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global i_nodes c_nodes plot_nodes tag_nodes nodes_listbox %#ok<GVMIS>
if i_nodes %Verifica se não está zerado o número de nós
   set(plot_nodes{i_nodes, 1},'Visible','off')
   set(plot_nodes{i_nodes,2},'Visible','off');
   i_nodes = i_nodes-1;
   c_nodes = c_nodes(1:end-1,:);
   plot_nodes = plot_nodes(1:end-1,:);
   tag_nodes = tag_nodes(1:end-1);
   nodes_listbox = nodes_listbox(1:end-1);
   set(handles.listbox_node,'String',nodes_listbox,'FontSize',12);
end


% --- Executes on button press in makeEdge_button.
function makeEdge_button_Callback(hObject, eventdata, handles)
% hObject    handle to makeEdge_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global tag_nodes edges_list i_edges c_nodes list_listbox edge_label type_label plot_edges
i_nodes
n1 = get(handles.edge_n1,'String');
n2 = get(handles.edge_n2,'String');
if str2double(n1)>=0 && str2double(n1)<=i_nodes && str2double(n2)>=0 &&
str2double(n2)<=i_nodes
   i_edges = i_edges+1;
   edges_list{i_edges,1} = tag_nodes{str2double(n1)};
   edges_list{i_edges,2} = tag_nodes{str2double(n2)};
   uiwait(edgeTag);
   edges_list{i_edges,3} = type_label;
   edges_list{i_edges,4} = edge_label;
   edge_split = strsplit(edge_label,',');
   edge_component = edge_split{1};
   switch edge_component
```

```
    case 'Current Source'
        edge_color = '#0000ff'; %blue
        edge_color_plot = [0 0 1];
    case 'Resistor'
        edge_color = '#00ff00'; %green
        edge_color_plot = [0 1 0];
    case 'Transistor'
        path_transistor = edge_split{4};
        if strcmp(path_transistor(1:2),'BC') || strcmp(edge_split{4},'CB')
            edge_color = '#ffff95'; %brighter yellow
            edge_color_plot = [1 1 149/255];
        elseif strcmp(path_transistor(1:2),'BE') || strcmp(edge_split{4},'EB')
            edge_color = '#ffff00'; %pure yellow
            edge_color_plot = [1 1 0];
        elseif strcmp(path_transistor(1:2),'CE') || strcmp(edge_split{4},'EC')
            edge_color = '#b78b00'; %darker yellow
            edge_color_plot = [183/255 139/255 0];
        else
            edge_color = '#ffffff'; %white
            edge_color_plot = [0 0 0];
        end
    case 'Mosfet'
        path_transistor = edge_split{4};
        if strcmp(path_transistor(1:2),'GD') || strcmp(edge_split{4},'DG')
            edge_color = '#ff964b'; %brighter chocolate
            edge_color_plot = [210/255 105/255 30/255];
        elseif strcmp(path_transistor(1:2),'GS') || strcmp(edge_split{4},'SG')
             edge_color = '#d2691e'; %chocolate
            edge_color_plot = [1 150/255 75/255];
        elseif strcmp(path_transistor(1:2),'DS') || strcmp(edge_split{4},'SD')
            edge_color = '#b94b00'; %darker chocolate
            edge_color_plot = [185/255 75/255 0];
        else
            edge_color = '#111111'; %white
```

```
            edge_color_plot = [1 1 1];
        end
    case 'Capacitor'
        edge_color = '#ff0000'; %red
        edge_color_plot = [1 0 0];
    case 'Diode'
        edge_color = '#7f7f7f'; %medium gray
        edge_color_plot = [0.5 0.5 0.5];
    case 'Inductor'
        edge_color = '#93b4cd'; %soft blue
        edge_color_plot = [147/255 180/255 205/255];
    otherwise
        edge_color = '#ffffff'; %white
        edge_color_plot = [1 1 1];
end
if strcmp(edge_component,'Transistor') || strcmp(edge_component,'Mosfet');
    edge_listbox = [edge_split{1} ' ' edge_split{2} ' - ' path_transistor];
else
    edge_listbox = [edge_split{1} ' ' edge_split{2}];
end
edges_list(i_edges,5) = {edge_color};
if strcmp(type_label,'undirected')
    list_listbox{i_edges,1} = [n1,' ---- ',n2,'  ( ',edge_listbox,' )'];
else
    list_listbox{i_edges,1} = [n1,' ---> ',n2,'  (',edge_listbox,')'];
end
set(handles.listbox1,'String',list_listbox,'FontSize',12);
plot_edges{i_edges,1}                                                                    =
plot([c_nodes(str2double(n1),1),c_nodes(str2double(n2),1)],[c_nodes(str2double(n1),2),c_no
des(str2double(n2),2)],'LineWidth',2,'Color',edge_color_plot);
else
    set(handles.popup_text,'Visible','on');
    pause(2);
    set(handles.popup_text,'Visible','off');
```

```
end



function edge_n1_Callback(hObject, eventdata, handles)
% hObject    handle to edge_n1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edge_n1 as text
%        str2double(get(hObject,'String')) returns contents of edge_n1 as a double



function edge_n2_Callback(hObject, eventdata, handles)
% hObject    handle to edge_n2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edge_n2 as text
%        str2double(get(hObject,'String')) returns contents of edge_n2 as a double


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes on button press in finish_button.
```

```
function finish_button_Callback(hObject, eventdata, handles) %#ok<INUSD>
% hObject    handle to finish_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global edges_list finish_ok File_Name tag_nodes %#ok<GVMIS>
uiwait(finishPopup)
if finish_ok
    nome_imagem = File_Name(1:end-4);
    xls_list = edges_list;
    xls_list(2:end+1,:) = xls_list;
    xls_list(1,:) = {'source','target','type','label','color'};
    writecell(xls_list, [nome_imagem '.csv'])
    save([nome_imagem '.mat'],'edges_list','tag_nodes');
    % Cria pasta e faz copia para esta
    try
        mkdir('Grafos\', nome_imagem);
        writecell(xls_list, ['Grafos\' nome_imagem '\' nome_imagem '.csv'])
        save(['Grafos\' nome_imagem '\' nome_imagem '.mat'],'edges_list','tag_nodes');
    catch
        disp('Erro ao salvar na pasta Grafos\');
    end
    close(cadastroGrafo)
end


% --- Executes on selection change in listbox_node.
function listbox_node_Callback(hObject, eventdata, handles)
% hObject    handle to listbox_node (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns listbox_node contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox_node
```

```
% --- Executes during object creation, after setting all properties.
function listbox_node_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox_node (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

APÊNDICE B – Código MATLAB® das métricas utilizadas

(a) Implementação da função “medidaGrau”

```
function [medio, desvio] = medidaGrau(mat)
% HELP ---------------------------------------------------
% [medio, desvio] = medidaGrau(mat)
% Returns average degree and std about the graph matrix in mat
% -------------------------------------------------------------
medio = sum(sum(mat))/length(mat);
desvio = std(sum(mat));
end
```

(b) Implementação da função “medidasDistância”

```
function [media, dp] = medidasDistancia(mat)
% [media, dp] = medidasDistancia(mat)
% media = média dos caminhos médios de todos os nós do grafo
% dp = desvio padrão dos caminhos médios dos nós do grafo
   matDist = [];
   for n = 1:length(mat)
      distancias = BFS(mat,n);
      aux = distancias(1,:);
      aux = aux(aux ~= 0); % Remove self distance
      aux = aux(aux ~= Inf); % Remove unconnected vertices
      matDist = horzcat(matDist, aux);
   end
   vet = reshape(matDist',1,[]);
   media = mean(vet); % média
   dp = std(vet); % desvio padrão
end
```

(c) Implementação da função "grauHierarquicoVertice"

```
function [gh,matViz] = grauHierarquicoVertice(matrix, p1)
% HELP ----------------------------------------------
% [gh,matViz] = grauHierarquicoVertice(matrix, p1)
% gh: grau hierarquico do nó "p1".
% matrix: matriz do grafo
% matViz: representa as vizinhanças hierarquicas de "p1"
% ------------------------------------------------------
   % Obtem as distâncias mínimas do ponto p1 até todos os outros
   distancias = BFS(matrix > 0,p1); % Passa a matrix binarizada, para que
                                    % o grau da ligação não afete as distancias
   % Obtem a vizinhança hierarquica ( matViz{1} = p1 )
   for i = 0:max(distancias(1,:))
      matViz{i+1,:} = find(distancias(1,:) == i);
   end
   % Calcula o número de conexões que liga uma vizinhança hierarquica à sua proxima
   gh = zeros(1,length(matViz)-1);
   for nivel = 1:length(matViz)-1
      for j = matViz{nivel}
         for i = matViz{nivel+1}
            gh(nivel) = gh(nivel) + matrix(i,j);
         end
      end
   end
end
```

(d) Implementação da função "grauHierarquico"

```
function gh_mat = grauHierarquico(mat)
% Help ----------------------------------------------------------------
% gh_mat = grauHierarquico(mat)
% gh_mat é uma matriz cuja suas colunas reresentam cada nó, e as linhas
% seus respectivos níveis de grau hierárquico.
```

```
% | p1_g1   p2_g1   p3_g1 |
% | p1_g2   p2_g2   p3_g2 |
% | p1_g3   p2_g3   p3_g3 |
% ------------------------------------------------------------------
   for i = 1:length(mat)
      gh{i} = grauHierarquicoVertice(mat, i);
      tam_vet(i) = length(gh{i});
   end
   max_tam = max(tam_vet);
   for i = 1:length(gh)
      vet = gh{i};
      vet(max_tam+1) = 0;
      gh_mat(i,:) = vet;
   end
   gh_mat = gh_mat';
end
```

(e) Implementação da função “matIndex”

```
function [media, desvio, mat_index] = matIndex(mat)
mat = mat ~= 0; % Binarizando
tam = length(mat);
mat = mat .* ~eye(tam); % Eliminando auto-ligações
mat_index = [];
% mat_index = [];
for i = 1:tam-1
   for j = i+1:tam
      if mat(i,j) ~= 0 % Somente se houver conexão entre o par
         mat_copy = mat;
         mat_copy(i,j) = 0; % Ignorando ligação entre os pares de nós
         mat_copy(j,i) = 0; % \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
         a_viz = find(mat_copy(i,:) == 1);
         b_viz = find(mat_copy(j,:) == 1);
         comum_viz = intersect(a_viz, b_viz);
```

```
            for aux_i = a_viz
                b_viz = b_viz(b_viz ~= aux_i);
            end
            total_viz = length(a_viz) + length(b_viz);
            if total_viz ~= 0
                mat_index(end+1) = length(comum_viz)/total_viz;
            end
        end
    end
end
media = mean(mat_index);
desvio = std(mat_index);
end
```

(f) Implementação da função "clusteringCoeficient"

```
function [media, desvio] = clusteringCoeficient(mat)
    mat = (mat ~= 0); % Matriz Unitária (o que importa é a conexão ou não)
    tam = length(mat);
    for i = 1:tam
        vizinhos = [];
        mat(i,i) = 0; % Desconsidera auto-ligações
        grau = sum(mat(i,:)); % Grau do nó
        if grau > 1 % Se existir mais de uma conexão:
            vizinhos = find(mat(i,:) == 1); % Encontra os vizinhos
            tam_viz = length(vizinhos);
            nV = 0; % Número de ligações entre vizinhos
            for j = 1:tam_viz-1
                for k = j+1:tam_viz
                    if mat(vizinhos(j),vizinhos(k)) == 1 % Se existir ligação entre esses vizinhos:
                        nV = nV+1; % Incrementa o número de ligações entre vizinhos
                    end
                end
            end
```

```
        cc(i) = 2*nV/(grau*(grau - 1)); % Calcula o coeficiente de alomeração do nó i
    else
        cc(i) = 0;
    end
  end
  media = mean(cc);
  desvio = std(cc);
end
```

(g) Implementação da função “k_means”

```
function [new_centroides,grupos_centroides] = k_means(X,k)
  centroides = X(:,2:k+1); %forçar ser diferente de new_centroide para entrar no loop
  new_centroides = X(:,1:k);
  while ~isequal(new_centroides, centroides)
    grupos_centroides(1:k) = {[]}; % reinicia os grupos centróides
    centroides = new_centroides;
    for i = 1:size(X,2)
      for j=1:k
        D(j) = sqrt(sum((centroides(:,j)-X(:,i)).^2));
      end
      min_index = find(D==min(D));
      % Para casos onde der a mesma distancia para dois ou mais pontos (pega o primeiro
ponto)
      min_index = min_index(1);
      grupos_centroides{min_index} = horzcat(grupos_centroides{min_index},i);
    end
    for a = 1:k
      % Média para os novos centróides
      new_centroides(:,a)                                                    =
sum(X(:,grupos_centroides{a}),2)/length(grupos_centroides{a});
    end
  end
end
```

(h) Função “BFS”

```
function distancias_minimas = BFS(matriz,p1)
        % distancias_minimas = BFS(matriz,p1)
        % distancias_minimas = [distancias_relativas;pai];
           fila = [];
           tam = size(matriz,1);
           for i = 1:tam
              cor(i) = 'B';
              d(i) = inf;
              pai(i) = -1;
           end
           cor(p1) = 'C';
           d(p1) = 0;
           pai(p1) = 0;
           fila(end+1) = p1;
           while ~isempty(find(fila ~= 0, 1))
              u = fila(1);
              fila = fila(2:end);
              if adjacente(matriz,u) ~= 0
                 for i = adjacente(matriz,u)
                    if cor(i) == 'B'
                       cor(i) = 'C';
                       d(i) = d(u)+1;
                       pai(i) = u;
                       fila(end+1) = i;
                    end
                 end
              end
              cor(u) = 'P';
           end
           distancias_minimas = [d;pai];
end
```

APÊNDICE C – Características dos circuitos amplificadores analisados

Tabela C1 – Breve descrição topológica dos circuitos amplificadores analisados

ID	Descrição
1	Estagio diferencial de entrada, divisor de fase básico e saída *push-pull*
2	Estagio diferencial de entrada, divisor de fase e saída *push-pull*
3	Estagio diferencial de entrada, divisor de fase básico e dois estágios de saída *push-pull*
4	Estagio diferencial de entrada, divisor de fase básico e dois estágios de saída *push-pull*
5	Estagio diferencial de entrada, divisor de fase diferencial e saída *push-pull*
6	Estagio diferencial de entrada, divisor de fase e dois estágios de saída *push-pull*
7	Estagio diferencial de entrada, divisor de fase e três estágios de saída *push-pull*
8	Entrada simples, divisor de fase básico e estágio de saída *push-pull*
9	Estagio diferencial de entrada, divisor de fase e dois estágios de saída *push-pull*
10	Emissor comum em cascata de dois estágios
11	Estagio diferencial de entrada, divisor de fase e saída *push-pull*
12	Entrada direta, divisor de fase e dois estágios de saída *push-pull*
13	Estagio diferencial de entrada, divisor de fase básico e dois estágios de saída *push-pull*
14	Entrada direta, divisor de fase e dois estágios de saída *push-pull*
15	Entrada direta, divisor de fase e dois estágios de saída *push-pull*
16	Entrada diferencial, divisor de fase e saída *push-pull* em cascata
17	Entrada diferencial, divisor de fase e saída *push-pull* em cascata
18	Entrada diferencial, divisor de fase e saída *push-pull* em cascata
19	Entrada diferencial, divisor de fase básico e saída *push-pull*
20	Dois estágios emissor comum
21	Entrada diferencial com fonte de corrente, divisor de fase e dois estágios de saída *push-pull*

Fonte: Elaborada pelo autor

Tabela C2 – Quantidade de componentes eletrônicos presentes nos circuitos amplificadores analisados

ID	Capacitor	Diodo	Indutor	MOSFET	Resistor	Transistor	Total
1	7	1	1	2	17	4	32
2	6	1	0	2	17	7	33
3	8	3	0	0	15	7	33
4	7	0	0	0	20	7	34
5	12	1	1	2	17	5	38
6	9	0	0	0	11	9	29
7	12	3	0	0	21	12	48
8	9	3	0	0	14	4	30
9	5	1	1	0	20	8	35
10	2	0	0	0	2	2	6
11	11	0	0	2	18	7	38
12	5	0	0	0	14	6	25
13	8	3	0	0	17	7	35
14	6	2	0	0	12	6	26
15	2	2	0	0	12	6	22
16	5	4	0	0	9	8	26
17	5	5	0	0	12	8	30
18	8	2	0	0	9	7	26
19	3	2	0	0	10	5	20
20	3	0	0	0	5	2	10
21	10	5	0	0	19	8	42

Fonte: Elaborada pelo autor

Tabela C3 – Valores das medidas baseadas nos grafos dos circuitos amplificadores analisados (métricas representadas pelos respectivos índices definidos na Tabela 2)

ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15
1	44	4,2	2,1	11,5	4,0	10,7	3,3	4,0	3,4	2,4	1,0	0,3	0,2	0,2	0,2
2	51	4,1	1,8	9,6	3,3	13,6	3,7	8,0	5,3	2,6	1,0	0,2	0,2	0,2	0,2
3	47	4,1	1,9	9,4	3,1	11,0	2,8	7,0	3,0	2,6	1,1	0,2	0,2	0,2	0,2
4	48	4,2	2,2	11,4	3,8	12,6	3,3	4,0	4,5	2,3	0,9	0,4	0,3	0,2	0,2
5	52	4,3	2,0	12,0	5,4	14,3	4,0	6,1	5,3	2,6	1,1	0,2	0,1	0,1	0,1
6	47	5,2	2,6	13,6	5,1	10,9	5,0	2,2	4,3	2,3	1,0	0,3	0,2	0,2	0,1
7	72	5,1	3,4	14,6	5,6	18,8	6,3	7,6	6,1	2,6	1,1	0,3	0,3	0,2	0,2
8	38	3,9	2,0	9,7	3,8	10,3	3,3	3,1	3,6	2,4	1,0	0,1	0,2	0,1	0,1
9	51	4,4	1,8	10,1	3,3	12,8	2,8	6,6	4,3	2,5	1,0	0,3	0,3	0,2	0,2
10	10	2,9	1,7	3,4	0,8	1,4	1,4	0	0	1,8	0,8	0,4	0,4	0,3	0,2
11	56	4,5	2,4	12,5	4,3	16,6	3,9	4,6	5,2	2,4	0,9	0,3	0,2	0,2	0,2
12	37	4,6	2,4	11,3	3,9	7,4	3,4	1,7	2,3	2,1	0,8	0,5	0,3	0,2	0,2
13	49	4,3	2,8	12,3	5,5	11,4	2,8	6,1	4,7	2,5	1,1	0,3	0,2	0,2	0,2
14	38	4,5	2,3	11,4	3,3	9,0	4,7	0,8	2,4	2,1	0,8	0,3	0,2	0,2	0,2
15	34	4,0	2,1	7,8	3,0	7,6	3,2	4,5	3,8	2,4	1,0	0,3	0,3	0,2	0,2
16	39	3,9	1,8	8,3	3,8	8,2	2,4	4,7	2,7	2,6	1,2	0,3	0,3	0,2	0,2
17	46	3,8	1,7	7,5	2,6	9,3	3,1	8,8	2,8	2,8	1,1	0,3	0,2	0,2	0,1
18	40	4,4	2,5	10,7	4,6	11,2	4,0	3,1	3,4	2,3	0,9	0,2	0,2	0,1	0,2
19	30	3,3	1,4	6,3	2,4	7,3	1,6	3,8	3,4	2,5	1,0	0,3	0,3	0,2	0,2
20	14	3,5	1,3	5,1	1,1	1,0	1,7	0	0	1,6	0	0	0,2	0,3	0,2
21	58	4,3	1,5	9,2	3,3	12,0	2,6	10,9	3,8	2,8	1,1	0,3	0,3	0,2	0,2

Fonte: Elaborada pelo autor

APÊNDICE D – Grafos dos circuitos amplificadores eletrônicos

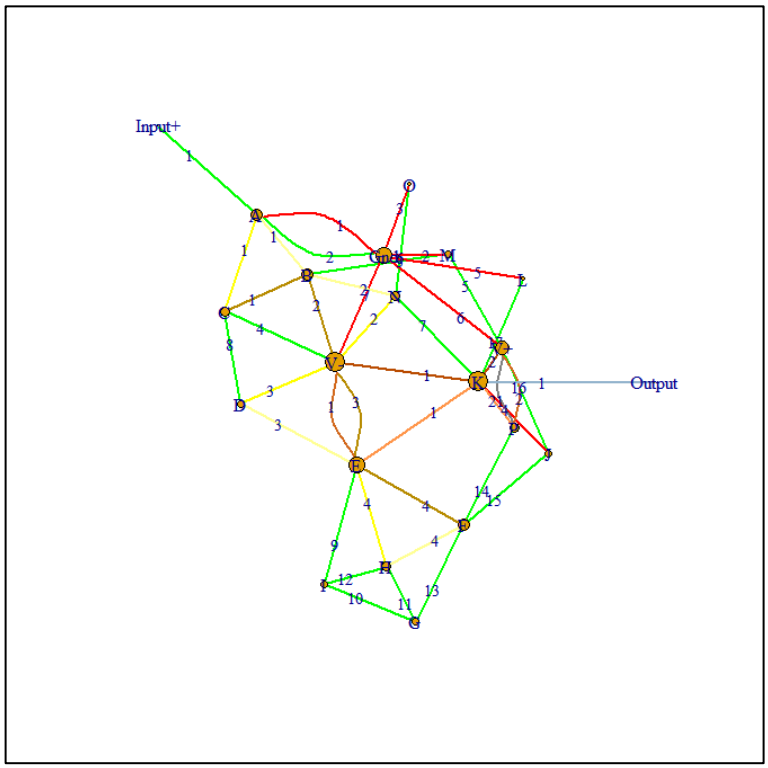


Figura D-1 – Grafo 1
Fonte: Elaborada pelo autor

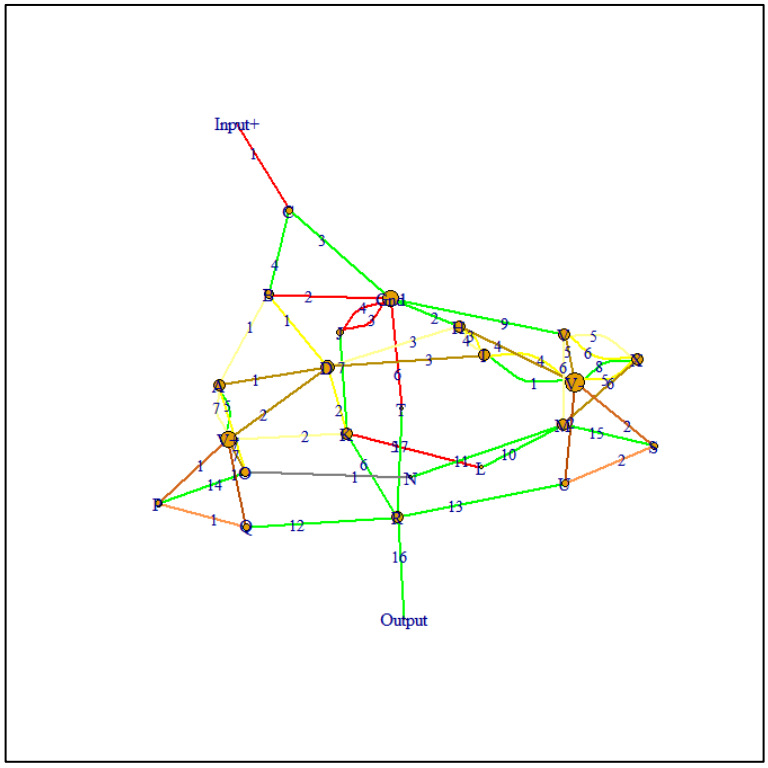


Figura D-2 – Grafo 2
Fonte: Elaborada pelo autor

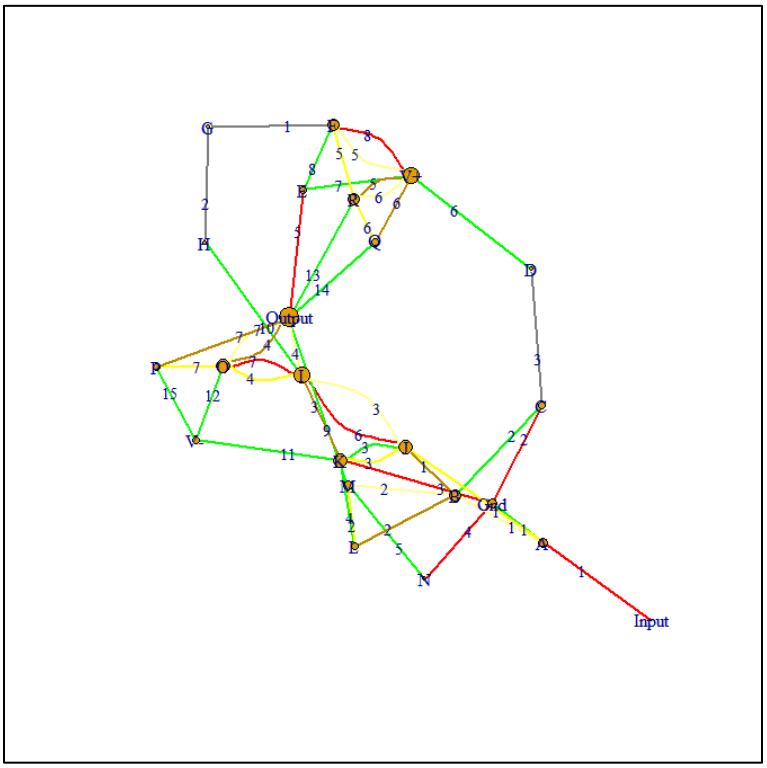


Figura D-3 – Grafo 3
Fonte: Elaborada pelo autor

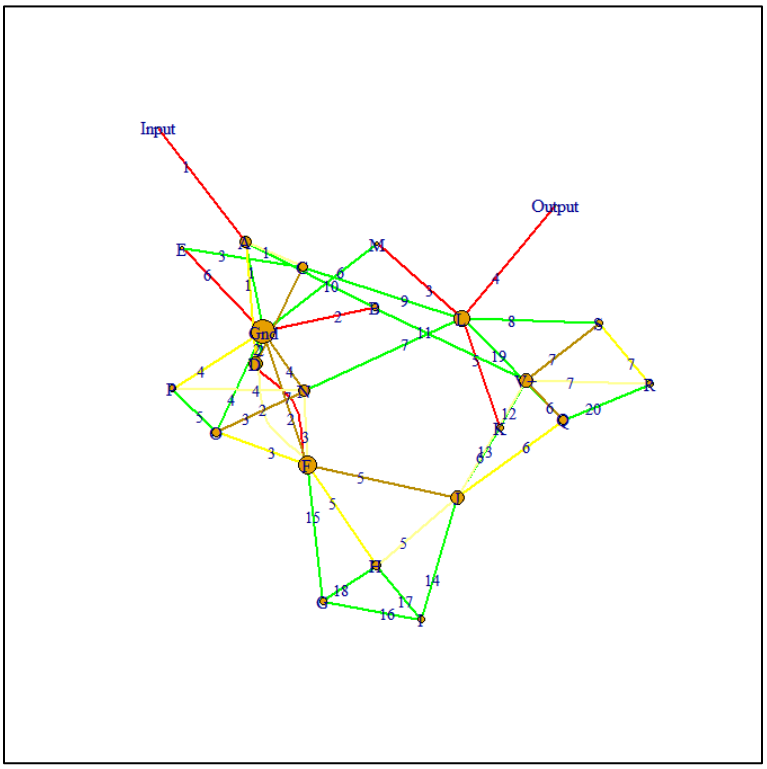


Figura D-4 – Grafo 4
Fonte: Elaborada pelo autor

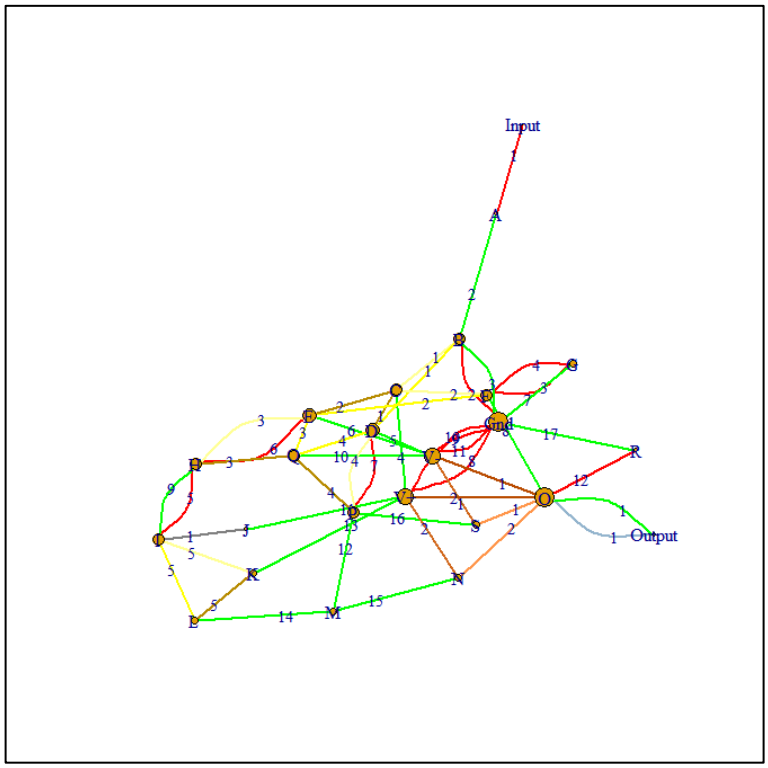


Figura D-5 – Grafo 5
Fonte: Elaborada pelo autor

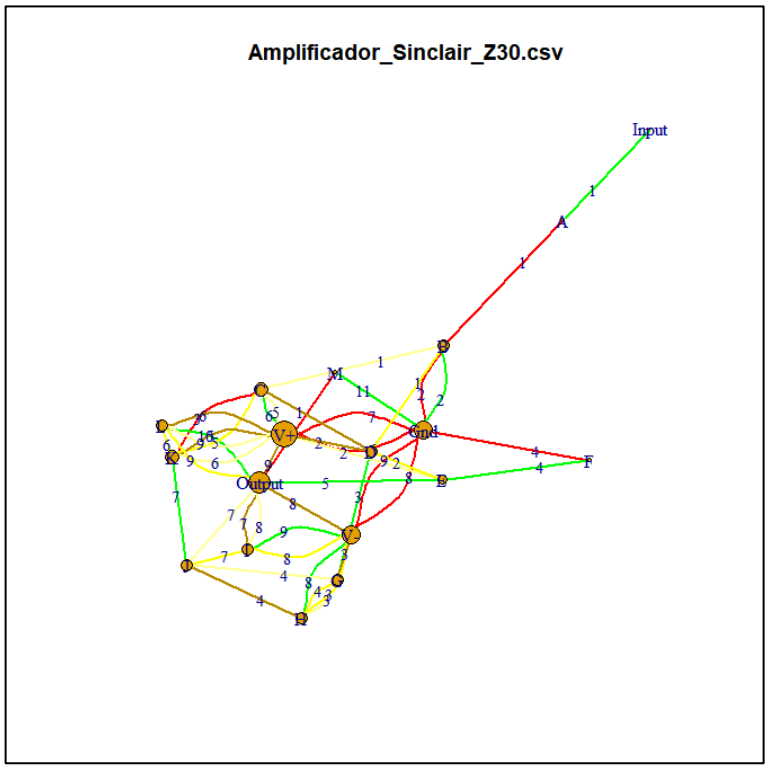


Figura D-6 – Grafo 6
Fonte: Elaborada pelo autor

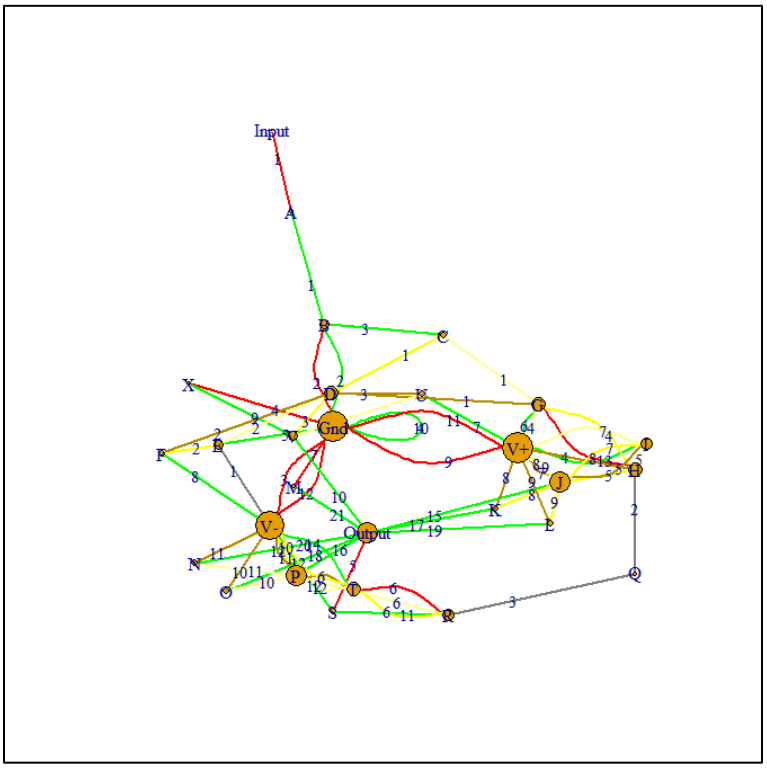


Figura D-7 – Grafo 7
Fonte: Elaborada pelo autor

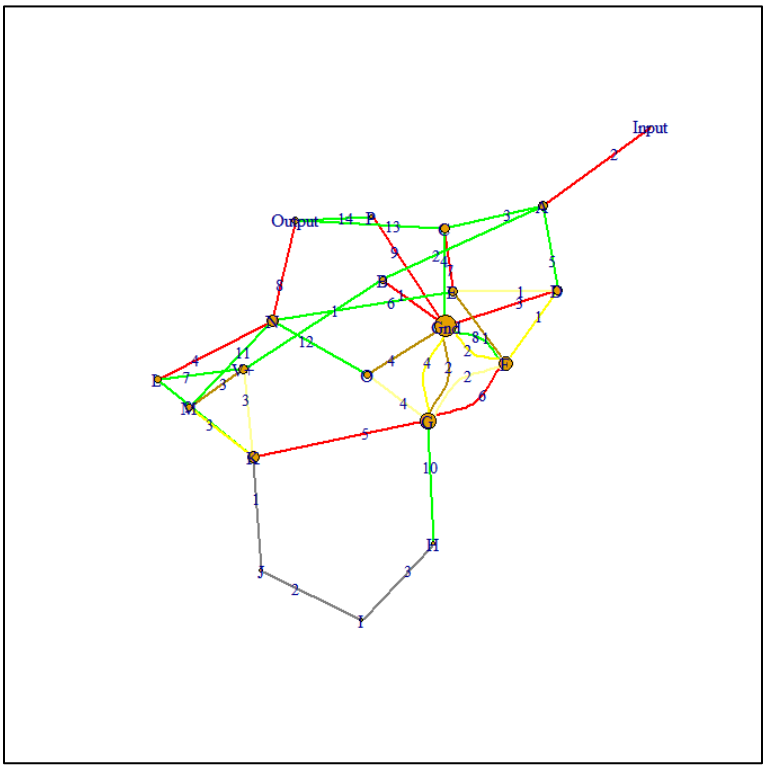


Figura D-8 – Grafo 8
Fonte: Elaborada pelo autor

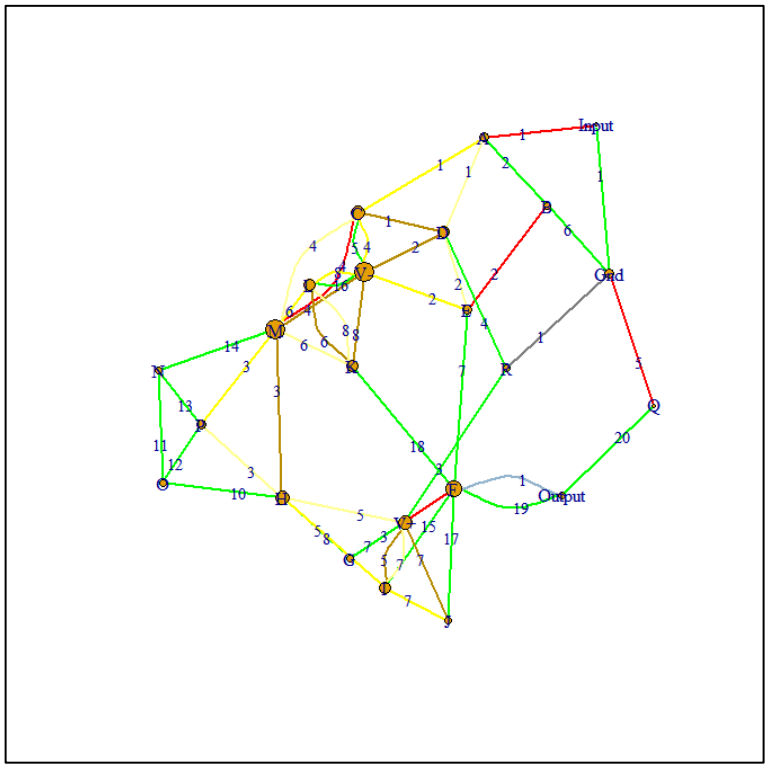


Figura D-9 – Grafo 9
Fonte: Elaborada pelo autor

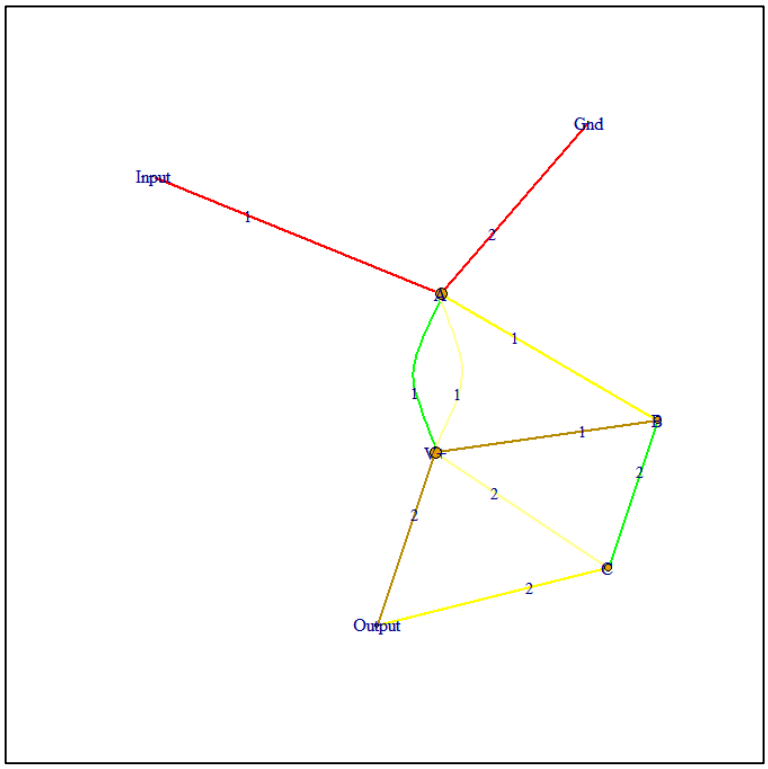


Figura D-10 – Grafo 10
Fonte: Elaborada pelo autor

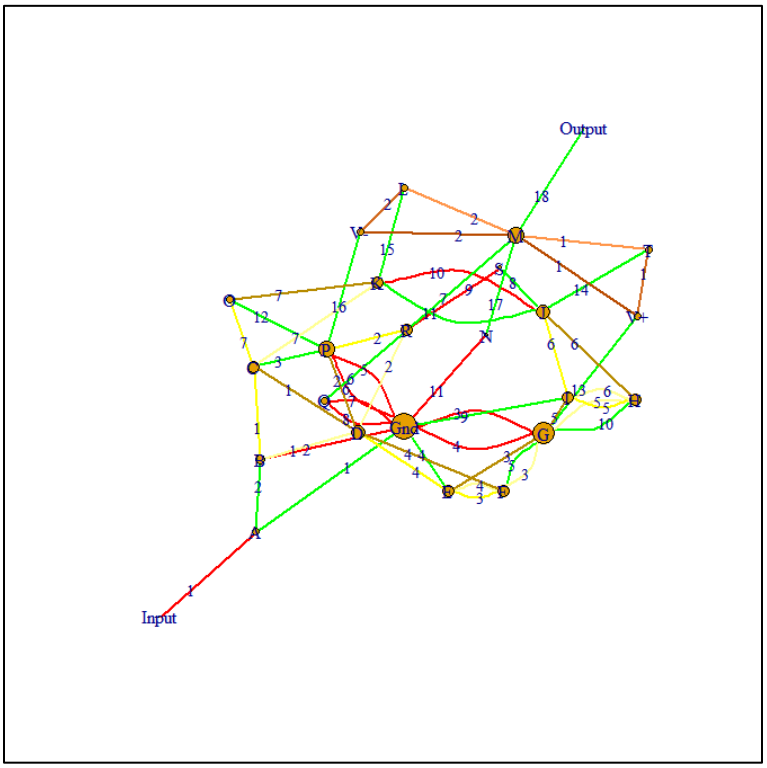


Figura D-11 – Grafo 11
Fonte: Elaborada pelo autor

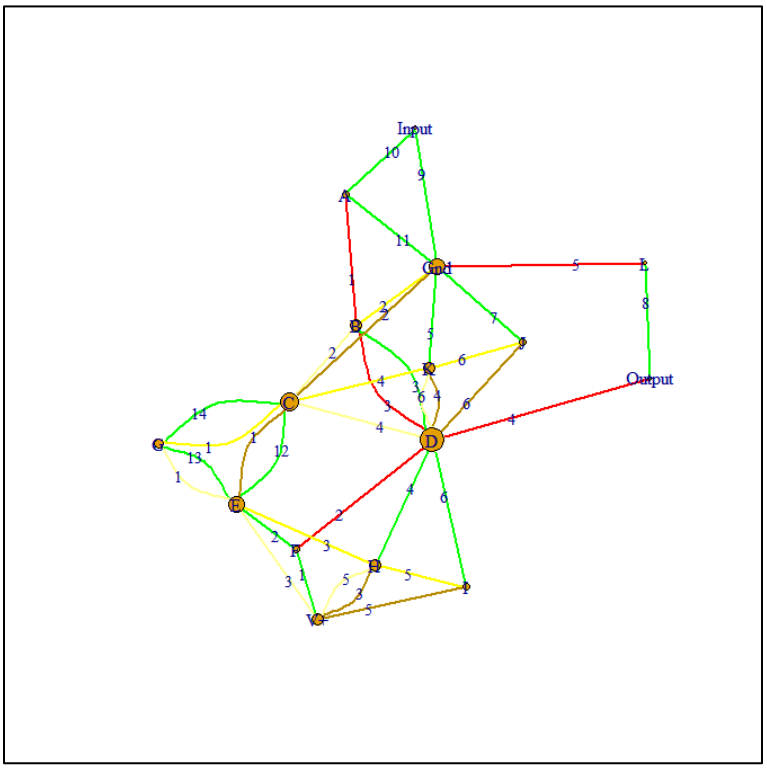


Figura D-12 – Grafo 12
Fonte: Elaborada pelo autor

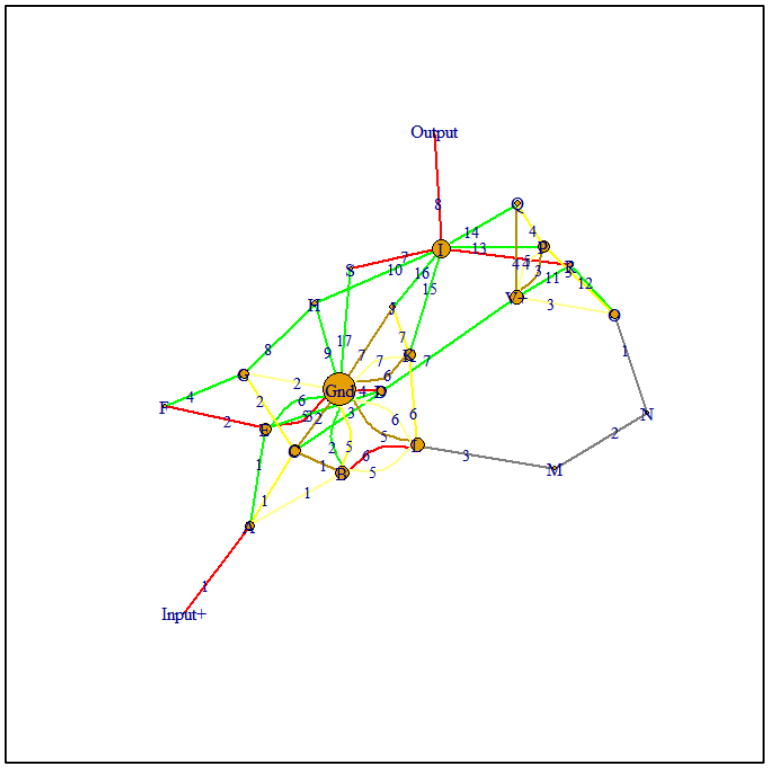


Figura D-13 – Grafo 13
Fonte: Elaborada pelo autor

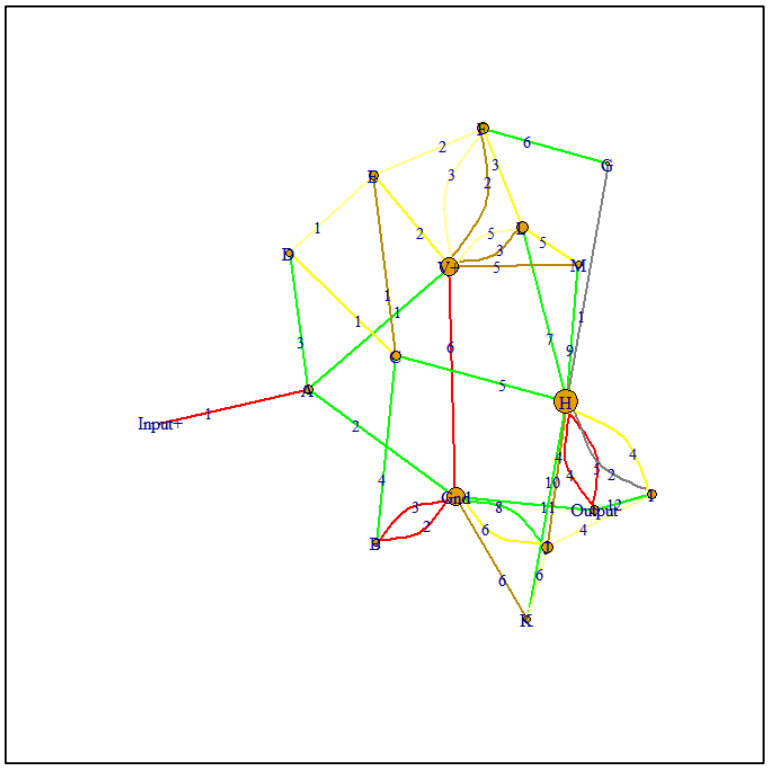


Figura D-14 – Grafo 14
Fonte: Elaborada pelo autor

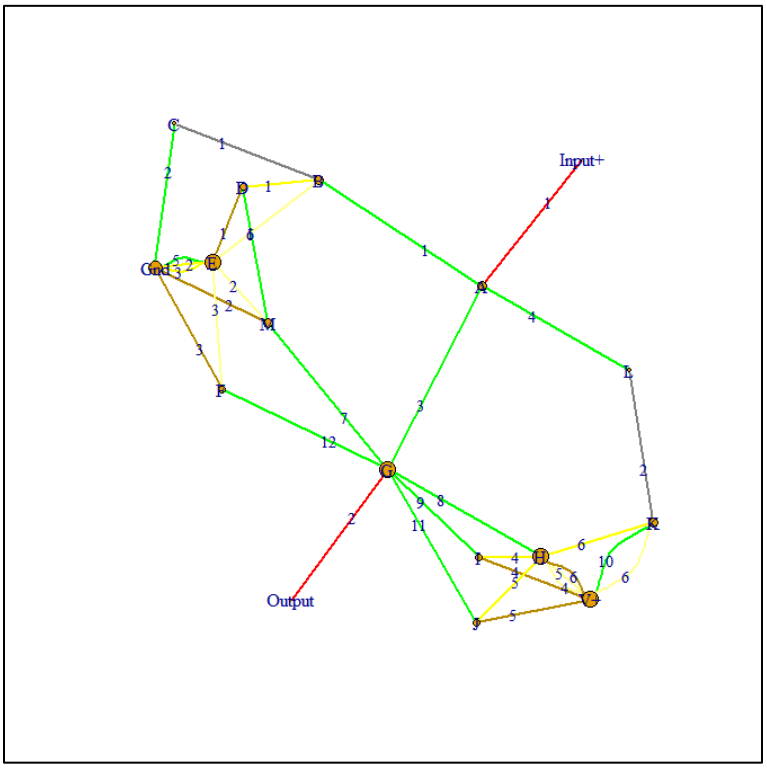


Figura D-15 – Grafo 15
Fonte: Elaborada pelo autor

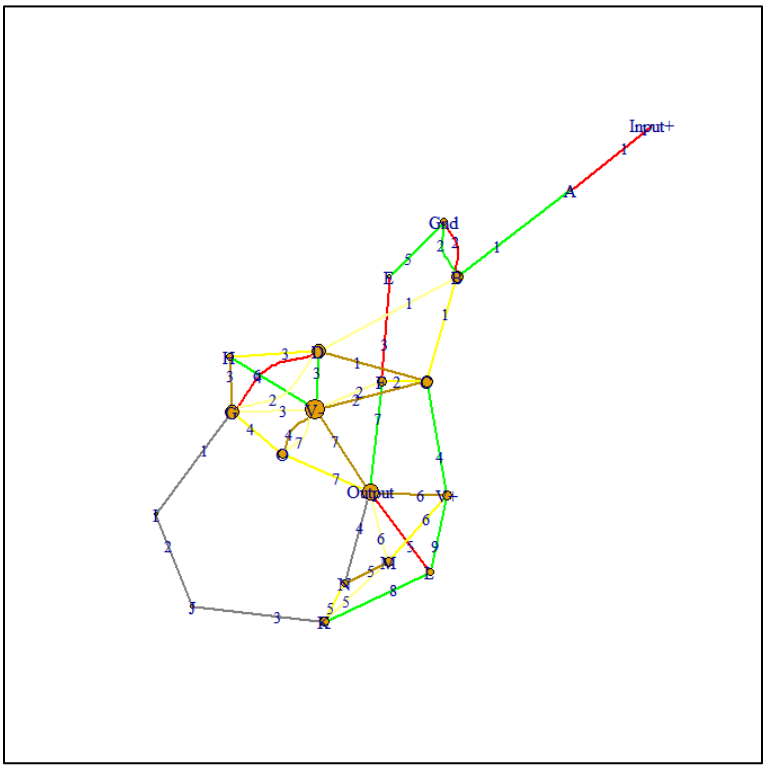


Figura D-16 – Grafo 16
Fonte: Elaborada pelo autor

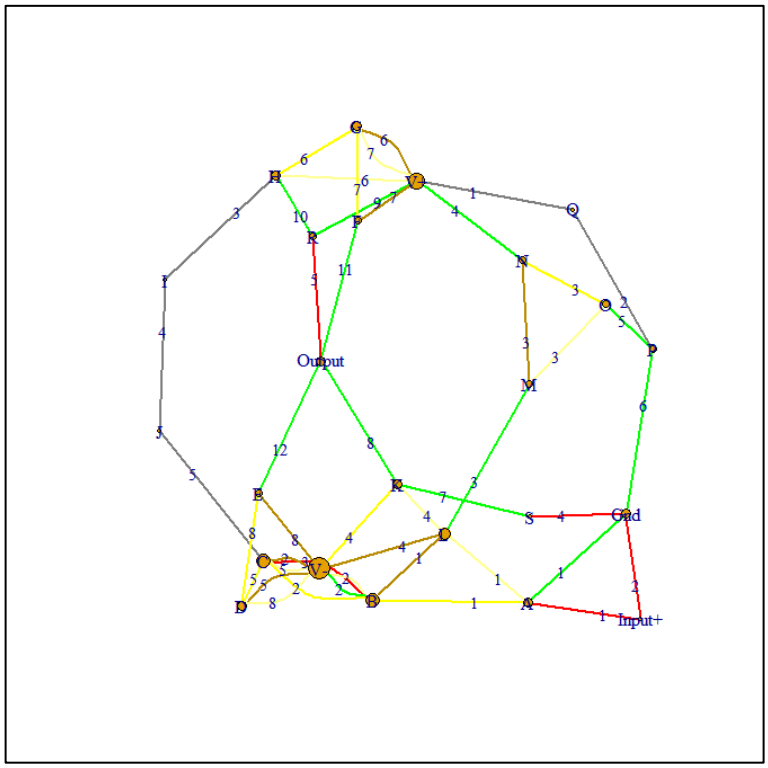


Figura D-17 – Grafo 17
Fonte: Elaborada pelo autor

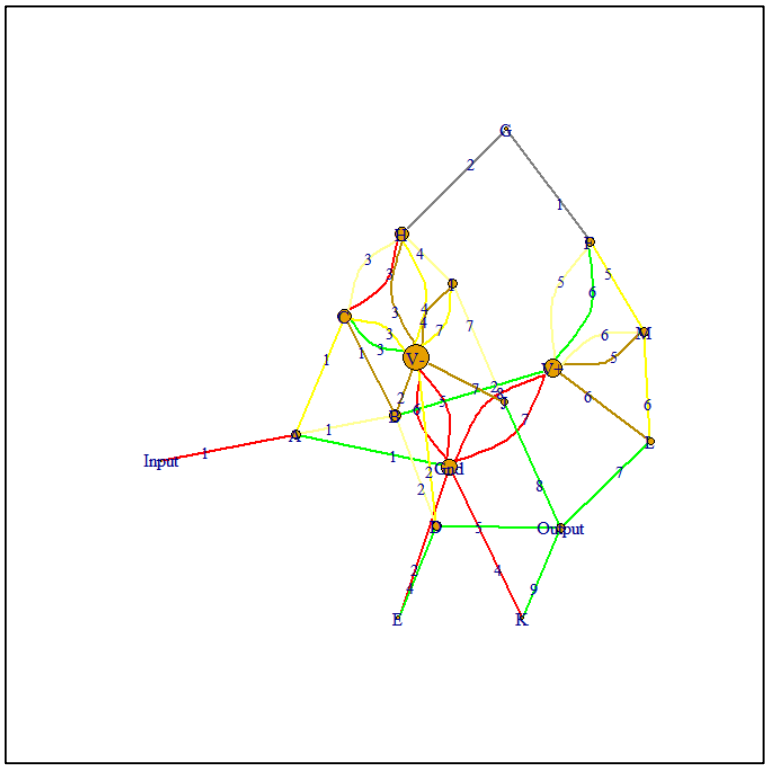


Figura D-18 – Grafo 18
Fonte: Elaborada pelo autor

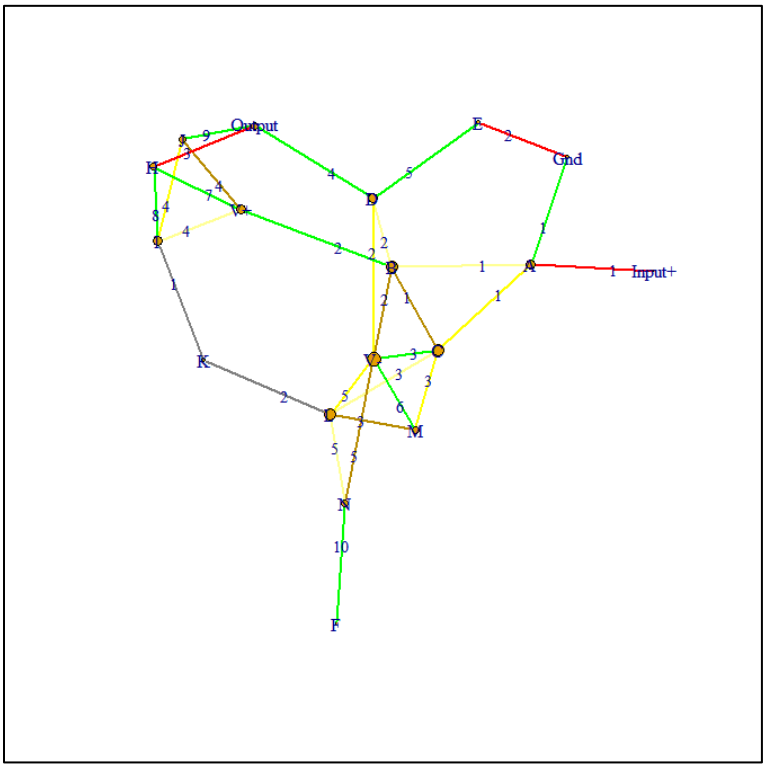


Figura D-19 – Grafo 19
Fonte: Elaborada pelo autor

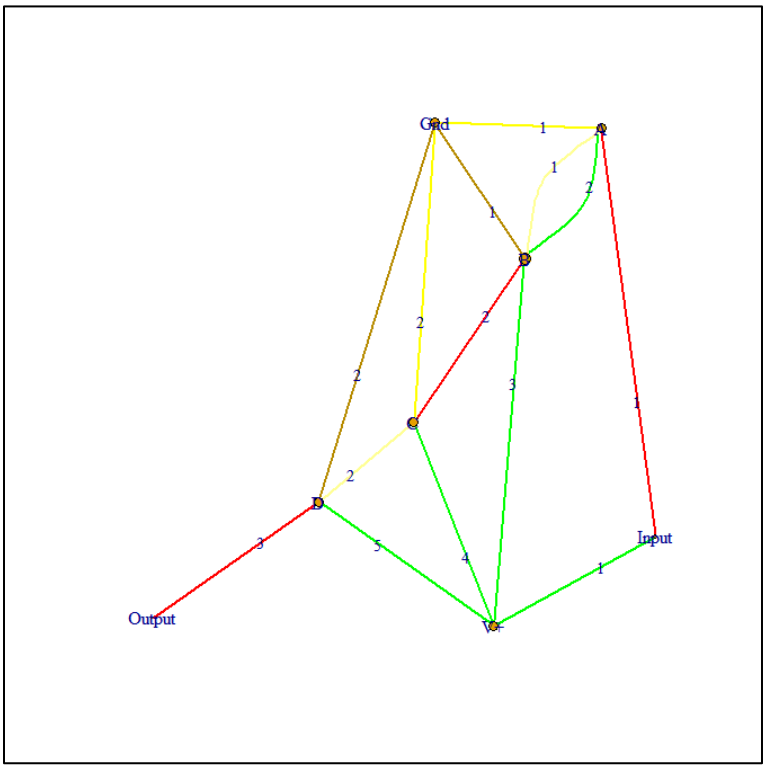


Figura D-20 – Grafo 20
Fonte: Elaborada pelo autor

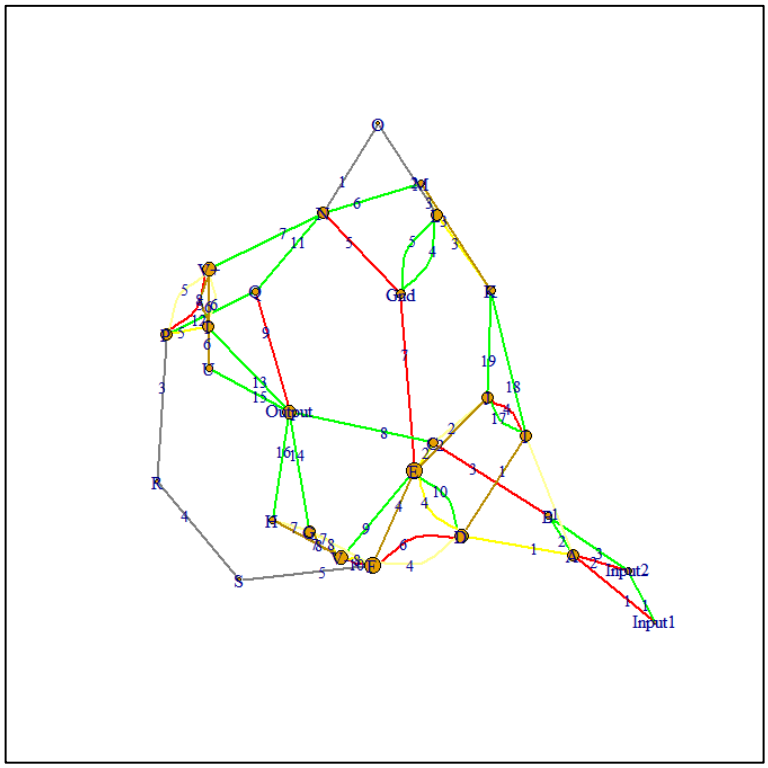


Figura D-21 – Grafo 21
Fonte: Elaborada pelo autor

APÊNDICE E – Lista de adjacência em uma lista separada por virgula (.csv)

"source,""target"",""type"",""label"",""color"""
"vi,""A"",""directed"",""1"",""#ff0000"""
"A,""gnd"",""undirected"",""1"",""#00ff00"""
"vcc,""B"",""undirected"",""2"",""#00ff00"""
"C,""vee"",""undirected"",""3"",""#00ff00"""
"D,""E"",""undirected"","",4"",""#00ff00"""
"D,""F"",""undirected"",""5"",""#00ff00"""
"vcc,""I"",""undirected"",""6"",""#00ff00"""
"L,""vee"",""undirected"",""7"",""#00ff00"""
"vo,""H"",""undirected"",""8"",""#00ff00"""
"J,""vo"",""undirected"",""9"",""#00ff00"""
"I,""F"",""directed"",""1"",""#7f7f7f"""
"F,""G"",""directed"","",2"",""#7f7f7f"""
"A,""B"",""undirected"",""1"",""#ffff00"""
"A,""C"",""undirected"",""1"",""#ffff95"""
"C,""B"",""undirected"",""1"",""#b78b00"""
"D,""vee"",""undirected"",""2"",""#ffff95"""
"D,""B"",""undirected"",""2"",""#ffff00"""
"vee,""B"",""undirected"",""2"",""#b78b00"""
"I,""vcc"",""undirected"",""3"",""#ffff95"""
"I,""J"",""undirected"",""3"",""#ffff00"""
"vcc,""J"",""undirected"",""3"",""#b78b00"""
"G,""vee"",""undirected"",""4"",""#ffff95"""
"G,""H"",""undirected"",""4"",""#ffff00"""
"vee,""H"",""undirected"",""4"",""#b78b00"""
"C,""G"",""undirected"",""5"",""#ffff95"""
"C,""L"",""undirected"",""5"",""#ffff00"""
"G,""L"",""undirected"",""5"",""#b78b00"""
"E,""gnd"",""directed"",""2"",""#ff0000"""