UNIVERSIDADE SANTA CECÍLIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
MESTRADO EM ENGENHARIA MECÂNICA

WELLINGTON TULER MORAES

MODELO PARA AUXILIAR A IDENTIFICAÇÃO DE CÓDIGO NUMÉRICO UTILIZANDO A LÓGICA PARACONSISTENTE ANOTADA: ESTUDO DE CASO EM CONTÊINERES

SANTOS/SP
2016

WELLINGTON TULER MORAES

MODELO PARA AUXILIAR A IDENTIFICAÇÃO DE CÓDIGO NUMÉRICO UTILIZANDO A LÓGICA PARACONSISTENTE ANOTADA: ESTUDO DE CASO EM CONTÊINERES

Dissertação apresentada à Universidade Santa Cecília como parte dos requisitos para obtenção de título de Mestre no Programa de Pós-Graduação em Engenharia Mecânica, sob orientação do Prof. Dr. Maurício Conceição Mário e coorientação do Prof. Dr. João Inácio da Silva Filho.

SANTOS/SP

2016

Moraes, Wellington Tuler.
Modelo para auxiliar a identificação de código numérico utilizando a Lógica Paraconsistente Anotada: estudo de caso em contêineres. 2016.

Orientador: Maurício Conceição Mário
Coorientador: João Inácio da Silva Filho

Dissertação de Mestrado - Universidade Santa Cecília, Programa de Pós-Graduação em Engenharia Mecânica Mestrado Em Engenharia Mecânica, Santos,SP, 2016.

1. Contêiner. 2.Código Numérico. 3. Lógica Paraconsistente. 4. Lógica Paraconsistente Anotada.
I. Mário, Maurício Conceição, orient.
II. Da Silva Filho, João Inácio, coorient.
III. Título. Modelo para auxiliar a identificação de código numérico utilizando a Lógica Paraconsistente Anotada: estudo de caso em contêineres.

Elaborada pelo SIBi – Sistema Integrado de Bibliotecas - Unisanta

Dedico este trabalho à minha amada esposa, aos meus pais e à minha querida irmã, que jamais deixaram de me apoiar.

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Maurício Conceição Mário, pela orientação, dedicação, compreensão e contribuição para o desenvolvimento deste trabalho.

Ao meu coorientador, Prof. Dr. João Inácio da Silva Filho, por pacientemente ter me ensinado todos os fundamentos da Lógica Paraconsistente.

À minha querida prima Profa. Dra. Íris Morais Araújo, por todas as correções e observações realizadas em meu trabalho.

Aos meus colegas de mestrado e amigos Sérgio Figueiredo, que me conduziu a esse programa de mestrado, Enir Fonseca, Nestor, Joseffe e Vitor, que foram exemplos de companheirismo e muito me ensinaram.

À minha amada esposa Jaqueline de Almeida Tuler Moraes, que me deu todo o apoio nos momentos mais difíceis e turbulentos nesta difícil jornada. Não tenho dúvida que sem esse apoio eu não chegaria ao que posso chamar de nosso objetivo, concluir este mestrado.

A minha maior ambição na vida é o de transmitir aos outros o que sei (Frank Sinatra).

RESUMO

Esta dissertação descreve a construção de um modelo de *software* que servirá para identificação de códigos numéricos extraídos de imagens digitais de contêineres. As imagens foram obtidas no porto de Santos (SP). Para a tomada de decisão da identificação dos códigos numéricos, foi utilizada a Lógica Paraconsistente Anotada (LPA), como alternativa ao emprego de técnicas tradicionais de Inteligência Artificial. A forma de tratamento de dados utilizada nesta dissertação foi o emprego de quatro diferentes Redes de análise da Lógica Paraconsistente Anotada de anotação com dois valores (LPA2v), que são capazes de fornecer resultados satisfatórios e suficientes para identificar os algarismos com base em padrões de imagens de códigos numéricos extraídos de contêineres. O *software* apresentado nesta dissertação possibilita a escolha de uma das quatro configurações diferentes de Rede de análise da Lógica Paraconsistente Anotada de anotação com dois valores. As redes podem ser treinadas com quantidade de padrões de códigos numéricos que variam de três a cinco, e em conjunto com as imagens de números extraídos de contêineres que servem de referência, geram resultados baseados na comparação dos testes com os valores obtidos na Rede de análise da Lógica Paraconsistente Anotada de anotação com dois valores. Os resultados apresentados alcançaram até 100% de assertividade na identificação dos códigos numéricos utilizando as melhores configurações de rede para cada algarismo, e servirão de base para agregar esta técnica em processos aduaneiros de terminais de contêineres.

Palavras-Chave*:* Contêiner. Código Numérico. Lógica Paraconsistente. Lógica Paraconsistente Anotada.

ABSTRACT

This dissertation describes the construction of a software model that will be used to identify numerical codes extracted from digital images of containers; the images were obtained at the port of Santos (SP). To take the decision of the identification of the numerical codes, Paraconsistent Annotated Logic (PAL), was used as an alternative to the use of traditional of Artificial Intelligence techniques. The way of data treatment used in this dissertation was the use of four different networks analysis Paraconsistent Annotated Logic with annotation of two values (PAL2v), which are able to provide satisfactory and sufficient results to identify the figures based on patterns of images of numerical codes extracted from containers. The software presented in this dissertation makes it possible to choose one of four different networks analysis Paraconsistent Annotated Logic with annotation of two values. Networks can be trained with number of numerical code patterns ranging from three to five, and with the images of extracted from containers numbers that are references, generate results based on the comparison of the tests with the values obtained in the network analysis Paraconsistent Annotated Logic with annotation of two values. The presented results achieved up to 100% of assertiveness in the identification of numerical codes using the best network configurations for each digit, and will serve as a basis for adding this technique to customs processes of container terminals.

Keywords: Container. Numeric Code. Paraconsistent Logic. Paraconsistent Annotated Logic.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interfaces*

BIC – Bureau International of Containers

ISO – International Organization for Standardization

JDK – *Java Development Kit*

LP – Lógica Paraconsistente

LPA – Lógica Paraconsistente Anotada

LPA2v – Lógica Paraconsistente Anotada com Anotação de dois valores

NAP – Sistemas ou Nós de Análise Paraconsistente

QUPC – Quadrado Unitário no Plano Cartesiano

RAP – Rede de Análise Paraconsistente

RAPcs – Rede de Análise Paraconsistente de configuração simples

RNA – Rede Neural Artificial

TEU – *Twenty-foot Equivalent Unit*

LISTA DE SÍMBOLOS

τ	Reticulado finito
~	Operador de negação
μ	Grau de crença (ou de evidência favorável)
μr	Grau de crença resultante
λ	Grau de descrença (ou evidência desfavorável)
T	Inconsistente
V	Verdadeiro
$\perp$	Paracompleto
$\mathbb{R}$	Conjunto dos números reais
$P\mu$	Sentença proposicional
Gc	Grau de Certeza
Gct	Grau de Contradição
p	Proposição Inicial
*p*T	Conotação de Inconsistente à proposição *p*
*p*1	Conotação de Verdade à proposição *p*
*p*0	Conotação de Falsidade à proposição *p*
p$\perp$	Conotação de Indefinição à proposição *p*
QUPC	Quadrado Unitário do Plano Cartesiano
$\perp \rightarrow$ f	Paracompleto tendendo ao Falso
$\perp \rightarrow$v	Paracompleto tendendo ao Verdadeiro
T$\rightarrow$ f	Inconsistente tendendo ao Falso
T$\rightarrow$ v	Inconsistente tendendo ao Verdadeiro
Qv $\rightarrow$T	Quase-verdadeiro tendendo ao Inconsistente
Qf $\rightarrow$T	Quase-falso tendendo ao Inconsistente
Qf$\rightarrow\perp$	Quase-falso tendendo ao Paracompleto
Qv$\rightarrow\perp$	Quase-verdadeiro tendendo ao Paracompleto
Gi	Grau de Incerteza
φ	Intervalo de certeza
$\varphi_{(\pm)}$	Intervalo de Certeza sinalizado
φ_E	Intervalo de Evidência Resultante
$\varphi_{E(\pm)}$	Intervalo de Evidência Resultante Sinalizado
G_{Cr}	Grau de Certeza resultante

G_{CR} Grau de Certeza real

SUMÁRIO

1. INTRODUÇÃO

A identificação precisa de contêineres é um procedimento fundamental para os terminais portuários, particularmente para que as entidades de fiscalização consigam realizar suas atividades na movimentação de carga (FONSECA et al., 2013).

Nesse contexto, a análise e caracterização de padrões de imagens no âmbito computacional tornam-se ferramentas imprescindíveis para auxiliar a extração e identificação de código numérico das imagens. Gerar melhorias na qualidade visual de certos aspectos estruturais resulta no aumento da percepção humana e na facilitação da interpretação automática por meio de *softwares*, eliminando, ou ao menos reduzindo, aspectos como subjetividade, fadiga, lentidão e custos associados à inspeção humana (PEDRINI; SCHWARTZ, 2008).

A ciência da tecnologia que permite extração de informações significativas, a partir de imagens capturadas por câmeras de vídeo, sensores, scanners, entre outros dispositivos, é conhecida como Visão Computacional (DE MILANO; HONORATO, 2010). Nessa tecnologia, há normalmente dois níveis de abstração estabelecidos: Processamento Digital de Imagens e Análise Digital de Imagens. O primeiro tem como objetivo melhorar a qualidade visual de certos aspectos estruturais, facilitando a percepção humana e a interpretação automática por meio de dispositivos (PEDRINI; SCHWARTZ, 2008). Já o segundo é elaborado com base na textura, forma, nos níveis de cinza ou também nas cores dos objetos presentes nas imagens. O seu caráter multidisciplinar pode ser considerado um agravante no grau de dificuldade, uma vez que diversos domínios de conhecimento são usualmente necessários na busca de solução para problemas, como a geometria computacional, visualização científica, estatística, teoria da informação, entre outras (PEDRINI; SCHWARTZ, 2008).

Segundo Dedgaonkar et al. (2012), vários são os métodos disponíveis para reconhecimento de caracteres. Há, por exemplo, o *Clustering*, que é o agrupamento automático de dado baseado no grau de semelhança; a Extração de Características (*Pattern Matching*), que é a análise da forma do caractere e comparação de suas características; e, por fim, a Rede Neural Artificial (RNA), que são modelos

computacionais capazes de reconhecer padrões e realizar o aprendizado de máquina, inspirados no sistema nervoso central de animais.

Nesta dissertação, optou-se por utilizar Redes de Análise Paraconsistente para tratamento de incertezas, que são constituídas de diversos Sistemas ou Nós de Análises Paraconsistentes (NAP), convenientemente interligados para analisar evidências provenientes de fontes incertas. Mesmo com informações conflitantes, essas Redes de Análise da Lógica Paraconsistente Anotada de anotação com dois valores (LPA2v) são capazes de fornecer resultados suficientes para originar uma ação, por serem capazes de tratar dados imprecisos, contraditórios e paracompletos sem o perigo de trivialização. A escolha das redes foi baseada no êxito do emprego de algumas aplicações com as características apontadas (FERRARA, 2004; MARIO, 2006; SOUZA, 2013).

A Lógica Paraconsistente Anotada (LPA) é uma classe da Lógica Paraconsistente Evidencial, que faz tratamento de sinais representados por anotações permitindo uma descrição e equacionamento por meio de algoritmos. Na Lógica Paraconsistente Anotada, as fórmulas proposicionais vêm acompanhadas de anotações. Cada anotação é pertencente a um reticulado finito que atribui valores à sua correspondente fórmula proposicional (DA SILVA FILHO; ABE, 2001).

1.1. OBJETIVO

O presente trabalho tem como objetivo fundamental criar um modelo para auxiliar a identificação de códigos numéricos extraídos de imagens digitais de contêineres, fundamentais para identificação desse equipamento, utilizando a Lógica Paraconsistente Anotada.

Neste trabalho optou-se pela utilização de 4 Redes de Análise Paraconsistente para tratamento de incertezas, que são constituídas de diversos Sistemas ou Nós de Análises Paraconsistentes (NAP), convenientemente interligados capazes de tratar dados imprecisos, contraditórios e paracompletos sem o perigo de trivialização.

O presente trabalho apresenta um estudo de caso da identificação de códigos numéricos extraídos de contêineres, desprezando as letras contidas em sua identificação, pois esse acrônimo representa a identificação do dono do contêiner e está fora do escopo do trabalho tratar caracteres que não sejam códigos numéricos.

1.2. OBJETIVOS ESPECÍFICOS

- Apresentar à área de operação de terminais portuários um modelo computacional baseado em Lógica Paraconsistente Anotada, capaz de auxiliar o processo de caracterização e identificação de códigos numéricos extraídos de contêineres, cumprindo seus requisitos legais junto às autoridades portuárias.
- Incrementar a assertividade no processo de identificação de contêineres utilizando uma lógica não clássica, a Lógica Paraconsistente Anotada, que pode oferecer outras soluções não convencionais em processos que trabalham em condições de incerteza.

1.3. RELEVÂNCIA DO TEMA

Nas últimas quatro décadas, segundo Steenken et al. (2004), o contêiner transformou-se em parte essencial no conceito de transporte de carga, alcançando inquestionável importância no transporte internacional principalmente no modal marítimo. Com o uso cada vez maior desse recipiente para o transporte de carga, o número de terminais de contêineres em portos marítimos e a competição entre os mesmos tornaram-se bastante notáveis.

As operações são hoje impensáveis sem o uso eficaz e eficiente da tecnologia da informação, bem como de uma otimização adequada dos processos operacionais.

Uma importante característica desta evolução foi a mecanização das operações de embarque e desembarque de mercadorias, que substituiu as forças animal e humana, por guindastes, correias transportadoras e contêineres. Esses últimos são embalagens práticas e resistentes, modulares e intercambiáveis, reconhecidas internacionalmente por suas características físicas e códigos de identificação (SCAZUFCA, 2008).

Para alcançar êxito no processo de armazenagem e troca de dados sobre os contêineres, utiliza-se um código internacional de identificação, normatizando e viabilizando sua modularidade. (ver Figura 1).

Figura 1: Identificação de contêiner.
(Fonte: SCAZUFCA, 2008).

Segundo a Companhia Docas do Porto de Santos – CODESP – Autoridade Portuária, a movimentação de contêineres vem sofrendo entre os anos de 2014 e 2015 um aumento anual de movimentação de unidades em 3,3%, independentemente do tamanho do contêiner ser de 20 ou 40 pés. Trata-se de um incremento de 2,6% na quantidade de unidades equivalentes a 20 pés ou *Twenty-foot Equivalent Unit* (TEU) e um aumento na quantidade de toneladas movimentadas de 5,5% (ver Tabela 1). Qualquer estudo que permita implementar melhorias na velocidade e assertividade do processo de identificação de contêineres será muito bem-aceito pelos meios acadêmico e de mercado:

CONTÊINERES (IMPORTAÇÃO E EXPORTAÇÃO)			
DESCRIÇÃO	**2014**	**2015**	**VAR %**
Unidades	2.374.426	2.453.881	3,3 (+)
TEU	3.684.845	3.779.999	2,6 (+)
Tonelagem	39.046.549	41.196.385	5,5 (+)

Tabela 1: Movimentação Anual de Contêineres no porto de Santos.
Adaptado pelo autor.
Fonte: (CODESP, 2016).

A Lógica Paraconsistente Anotada - LPA possibilita um incremento no processo de tipificação dos padrões de códigos numéricos exibidos nos contêineres, permitindo a sua identificação com maior precisão. Mesmo quando o código numérico não estiver totalmente nítido − problema hoje enfrentado pelos terminais no processo de identificação −, a LPA faz tratamento de sinais incertos, representados por anotações, propiciando uma descrição e equacionamento por meio de algoritmos (DA SILVA FILHO; ABE, 2001).

1.4. ESTRUTURA DO TRABALHO

Três capítulos compõem o presente trabalho. Ainda na introdução, serão abordados os objetivos, a relevância do tema e a estrutura do trabalho.

No primeiro capítulo, a fundamentação teórica apresenta os principais conceitos relacionados ao Processamento Digital de Imagens: Imagem Digital e Análise Digital de Imagens, bem como a Lógica Paraconsistente – LP, além das características da Linguagem de Programação *Java* utilizada para desenvolver o *software* resultante do presente trabalho.

Em seguida, as etapas e os métodos para desenvolvimento do *software* e do modelo computacional são descritos no segundo capítulo, bem como as equações utilizadas para obter as informações de leitura e identificação dos códigos numéricos.

No terceiro capítulo, serão expostos os resultados obtidos através dos ensaios realizados pelo uso do aplicativo.

2. FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica compreende as normas para a identificação de contêineres, os principais conceitos das Lógicas clássica e não clássica, Lógica Paraconsistente, Lógica Paraconsistente Anotada com Anotação de Dois Valores e as Rede de Análise Paraconsistente para tratamento de incertezas.

2.1. NORMAS PARA A IDENTIFICAÇÃO DE CONTÊINERES

A identificação de contêineres é tratada por normas que foram apresentadas pela Bureau International of Containers (BIC). A BIC propôs um código internacional de identificação e foi justamente através do International Organization for Standardization (ISO) que houve a oficialização em 1972 (HAPAG, 2002).

O contêiner de 20' (Lê-se: 20 pés) possui capacidade de carga equivalente a 21,92 toneladas. Já o contêiner de 40' (40 pés) possui capacidade igual a 26,93 toneladas (HAPAG, 2002).

Ambos são utilizados para carga seca em geral, como peças de automóveis, componentes eletrônicos, entre outros materiais.

Com relação à identificação, cada contêiner tem 4 letras, das quais 3 serão letras que correspondem ao proprietário e a quarta sempre será a letra U, de *Unit* ou Unidade. Exemplo: HAMBURG SUD: SUD, corresponde ao proprietário e U, corresponde à unidade (HAPAG, 2002).

O número de sete dígitos, sendo o sétimo, o verificador ou de controle para a segurança no transporte, é o de série do equipamento desse proprietário. Assim, o código do proprietário e o seu respectivo número são únicos, o que permite a perfeita identificação de uma unidade de contêiner (HAPAG, 2002).

2.2. INTRODUÇÃO À LÓGICA CLÁSSICA E NÃO CLÁSSICA

A Lógica Clássica, em algumas situações, não representa satisfatoriamente o mundo real, que se caracteriza por ser incompleto e contraditório. Tal Lógica trabalha com dois valores (verdadeiro ou falso), representada pelos níveis lógicos 0 ou 1.

Recentemente, diversas lógicas consideradas Não Clássicas, pois trabalham com sinais de informações representados de modo incompleto ou contraditório, têm

surgido e se mostrado eficientes para a aplicação direta em inúmeros campos do conhecimento.

A seguir serão apresentados os fundamentos da Lógica Paraconsistente – LP, que é considerada uma lógica Não Clássica, pois admite o tratamento de informações contraditórias na sua estrutura teórica. A LP tem sido usada em relevantes trabalhos na área de Inteligência Artificial, por ser considerada uma boa solução para o tratamento de situações incertas de modo não trivial (DA SILVA FILHO, 2007).

2.2.1. Lógica Clássica e Não Clássicas

Relatos históricos indicam que a Lógica Clássica teve início entre os anos de 384-322 a.C., com Aristóteles e seus discípulos. Tais filósofos buscavam um instrumento para definir os princípios universais do pensamento, estabelecendo regras práticas. Segundo Da Silva Filho e Abe (2000), tem-se as seguintes definições para a Lógica Clássica:

- Lógica é um conjunto de normas deduzidas das leis psicológicas com a finalidade de dirigir as operações do pensamento;
- Lógica é a Ciência que estuda as Leis do Raciocínio;
- A Lógica Estabelece as leis do Raciocínio, a maneira certa de como a razão deve operar, pouco importando se o raciocínio tem fundamento ou não na realidade.

A Lógica Clássica possui os seguintes princípios:

- Princípio da identidade: todo objeto é idêntico a si mesmo: $p = p$.
- Princípio da identidade proposicional, o qual toda proposição implica nela mesma: $p \rightarrow p$
- Princípio da contradição: de duas proposições contraditórias (i.e., uma é a negação da outra), uma delas deve ser falsa: $\neg (p \vee \neg p)$.
- Princípio do meio (ou do terceiro) excluído: de duas proposições contraditórias, uma delas deve ser verdadeira: $p \vee \neg p$.

Assim, apenas duas situações são consideradas na Lógica Clássica. É possível classificá-la como binária, o qual os dois estados lógicos possíveis são os estados verdadeiro e falso, também representados pelos valores numéricos 1 (um)

ou 0 (zero). São nos dígitos binários que a tecnologia atual se fundamenta, tornando-se dessa forma impossível modelar e tratar situações contraditórias, vagas, ambíguas e pouco óbvias (DA SILVA FILHO; ABE, 2000).

Para algumas situações, a Lógica Clássica não demonstra de forma coerente o mundo real, que é incompleto e contraditório. Nesse cenário, surgem diversas Lógicas Não Clássicas, aplicáveis a inúmeros campos de conhecimento de forma satisfatória. Tais lógicas operam com sinais de informações representados de modo incompleto ou contraditório. Pode-se considerar que essas lógicas complementam a Lógica Clássica Tradicional (LEMES NETO; VENSON, 2002) (ver figura 2):

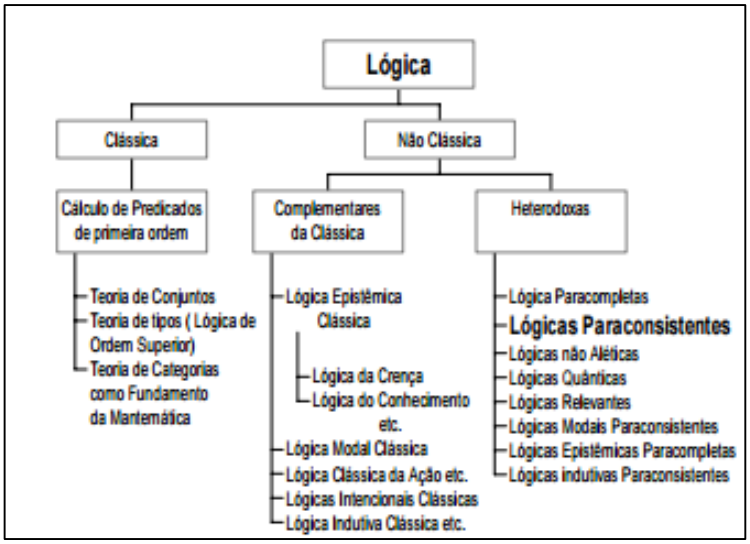


Figura 2: Classificação de algumas lógicas.
(Fonte: LEMES NETO; VENSON, 2002).

2.2.2. Lógica Paraconsistente

Representada por apenas dois valores possíveis, verdadeiro ou falso, a Lógica Clássica tem característica binária (níveis 0 ou 1). Isso implica em não retratar de modo satisfatório as aplicações que refletem o mundo real, incompleto e contraditório.

De acordo com Lemes Neto e Venson (2002), o modelo binário tem grandes dificuldades de tratar inconsistências, paradoxos, incertezas e ambiguidades que acontecem no mundo real. Na área de Inteligência Artificial, a Lógica Clássica não é eficaz, pois, como descrevem os autores, incertezas, contradições e ambiguidades são sempre presentes.

O surgimento das Lógicas Não Clássicas ocorre em função da ineficácia da Lógica Clássica em tratar sinais de informações representados de maneira incompleta e contraditória. Nesse cenário, a Lógica Paraconsistente – LP admite o tratamento de informações contraditórias na sua estrutura teórica (DA SILVA FILHO, 2006).

Como apresentado em Da Silva Filho, Abe e Lambert-Torres (2008), em 1948 trabalhos independentes desenvolvidos pelo polonês Stanislaw Jàskowski e pelo brasileiro Newton C. A. Da Costa (de 1954 em diante) deram início ao que se conhece como Lógica Paraconsistente. O principal destaque dessa lógica é o fato de infringir o princípio da não contradição encontrado na Lógica Clássica.

Em 1976, no III Simpósio Latino-Americano de Lógica e Matemática, realizado na UNICAMP (Campinas, SP), o peruano José Francisco Miró-Quesada pronunciou pela primeira vez a expressão Lógica Paraconsistente, dando início à divulgação do termo pelo meio científico em geral (DA SILVA FILHO, 1999).

Segundo Abe (2013), a Lógica Paraconsistente introduz duas novas categorias, além do Verdadeiro e do Falso. Pode-se ter proposições classificadas como Verdadeiras, Falsas, Inconsistentes ou Paracompletas. Classifica-se como Inconsistente uma proposição que possui uma evidência que sugere que ela seja Verdadeira, e outra evidência que sugere que ela é Falsa. Já uma proposição Paracompleta é aquela que não se tem evidência de que ela seja Verdadeira, tampouco Falsa.

2.2.3. Teorias Inconsistentes e Triviais

Como destaca Silva (2011), a Lógica Paraconsistente serve como base para teorias inconsistentes e não triviais. Aplicando-se os conceitos da Lógica Paraconsistente tem-se a possibilidade de manipular sistemas inconsistentes de informação, não subentendendo a trivialidade da teoria.

Aplicando-se apenas a Lógica Clássica, que é limitada a apenas valores Falsos e Verdadeiros, tem-se muitas vezes a sua ineficácia ou impossibilidade de aplicação.

De acordo com Da Silva Filho, Abe e Lambert-Torres (2008), os enunciados demonstrados como verdadeiros em uma teoria são conhecidos como teoremas. Seja **T** uma teoria fundada sobre uma lógica **L**, e suponha-se que a linguagem de **L**

e **T** contenha um símbolo para negação. A teoria **T** diz-se inconsistente se ela possuir teoremas contraditórios, isto é, que uma é negação da outra; caso contrário, **T** diz-se consistente. A teoria **T** diz-se trivial se todas as fórmulas de **L** forem teoremas de **T**; e, hipótese contrária, **T** chama-se não trivial.

Se **L** for uma das lógicas comuns, como a Clássica, a teoria **T** é trivial se e somente se for inconsistente. Lógicas como essas não separam os conceitos de inconsistência e trivialidade, pois, segundo a Lógica Clássica, uma teoria inconsistente também é trivial e vice-versa. Como tal é um resultado indesejável, a Lógica Clássica não admite a contradição como elemento aceitável sem torná-la trivial (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

2.2.4. Lógica Paraconsistente Anotada

Originária das famílias das lógicas não clássicas, as lógicas Paraconsistentes Anotadas são empregadas inicialmente em programação lógica por Subrahmanian (1987). Tais aplicações deram origem à teoria geral da programação anotada, que se mostraram aplicáveis a bases de dados que contêm contradições. As ideias foram estendidas por outros pesquisadores e a empregaram no raciocínio sobre redes de herança. Essas técnicas também englobam os raciocínios *fuzzy* e formalismos temporais (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

Conforme apresentado em Da Silva Filho (1999), as fórmulas proposicionais da Lógica Paraconsistente Anotada vêm acompanhadas de suas respectivas anotações, cada qual pertencente a um reticulado finito τ (ver figura 3). No reticulado, são atribuídos valores às fórmulas proposicionais correspondentes, onde as constantes anotacionais representarão os estados τ = {T, V, F, $\perp$}, correspondentes respectivamente a Inconsistente (T), Verdadeiro (V), Falso (F) e Indeterminado ou Paracompleto ($\perp$).

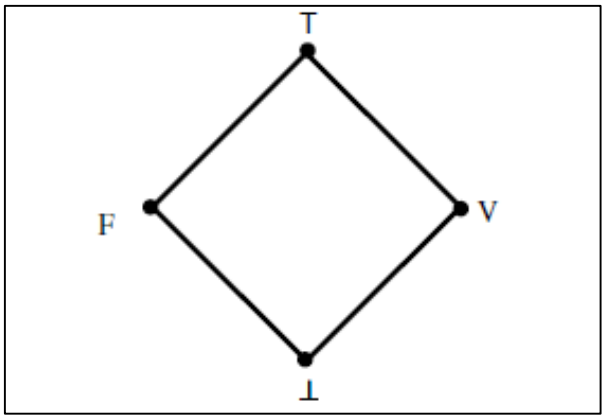


Figura 3: Reticulado de quatro vértices.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

Na Lógica Clássica, apenas dois estados são admitidos, sendo eles: Verdadeiro ou Falso, que são representados por 1 ou 0 respectivamente. Aplicando o operador lógico de negação **(~)** sobre esses estados tem-se o seguinte resultado:

$$\sim (1) = 0$$

$$\sim (0) = 1$$

Aplicando-se o operador de negação **(~)** na Lógica Paraconsistente Anotada, tem-se o seguinte cenário: o operador sobre τ é **~ : | τ |⟶ | τ |**, que de forma intuitiva operará da seguinte maneira (ver tabela 2):

Negação	Resultado	Descrição
~ (V)	F	A negação de uma proposição “verdadeira” é “falsa”
~ (F)	V	A negação de uma proposição “falsa” é “verdadeira”
~ (T)	T	A negação de uma proposição “inconsistente” é “inconsistente”
~ (⊥)	⊥	A negação de uma proposição “Paracompleta” é “Paracompleta”

Tabela 2: Aplicação do Operador Lógico de Negação na Lógica Paraconsistente Anotada. Adaptado pelo autor.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

As proposições da LPA são do tipo $P\mu$, onde P é uma proposição no sentido comum e μ é uma constante de anotação (DA SILVA FILHO, 2013).

A proposição P_μ pode ser interpretada como: **“**creio na proposição P com grau de crença de até μ, sendo que cada grau de crença atribuído à proposição estará contido no conjunto de valores compostos pelas constantes anotacionais do reticulado {V, F, T, ⊥}, que dará uma das conotações {“verdade”, ”falsidade”, ”inconsistência”, “indeterminação”} à proposição” (DA SILVA FILHO, 2013).

Portanto, conforme Da Silva Filho (1999), uma sentença proposicional associada ao reticulado da LPA será lida:

*P*T – “A anotação ou Grau de Evidência T atribui uma conotação de inconsistência à proposição P”.

*P*V – “A anotação ou Grau de Evidência V atribui uma conotação de verdade à proposição P”.

*P*F – “A anotação ou Grau de Evidência F atribui uma conotação de falsidade à proposição P”.

P⊥- “A anotação ou Grau de Evidência atribuiu uma conotação de indeterminação à proposição P”.

2.2.5. Lógica Paraconsistente Anotada com Anotação de Dois Valores

Na Lógica Paraconsistente Anotada, a anotação pode ser composta por 1, 2 ou n valores. A apresentação dos sinais de forma contraditória e com necessidade de tratamento dessa contradição, oriundos da mesma fonte ou de fontes diferentes, torna a LPA com a anotação de dois valores (LPA2v) adequada para ajudar a solução das inconsistências.

Dois valores de crença, ou de graus de evidência, são associados para cada proposição. A análise desses valores resulta em uma saída denominada estado lógico resultante. Desse modo, pode-se utilizar um reticulado τ, associado a LPA2v, tal que $\tau = \{(\mu, \lambda) \mid \mu, \lambda \in [0, 1] \subset \Re\}$ (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

Na figura 4 abaixo, tem-se representado um reticulado de quatro vértices associado à Lógica Paraconsistente Anotada de anotação com dois valores (LPA2v):

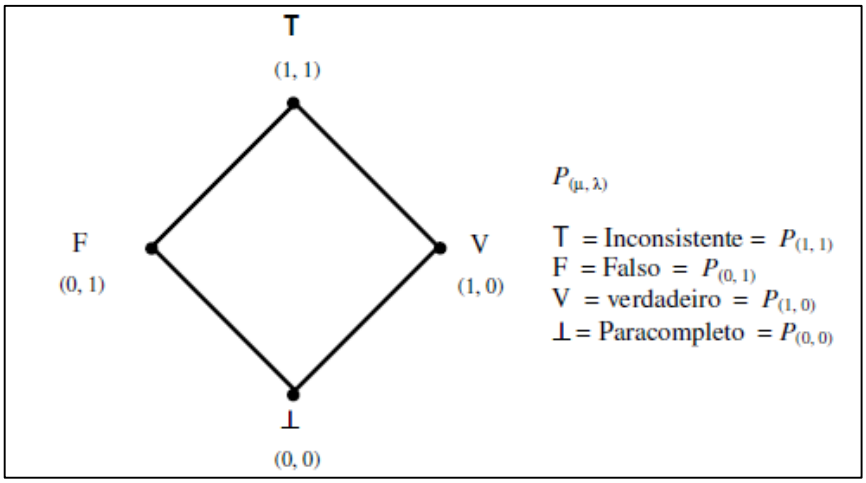


Figura 4: Reticulado finito de Hasse com anotação associado à LPA2v.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

Da Silva Filho, Abe e Lambert-Torres (2008) definem que o primeiro elemento do par ordenado (μ) representa o grau em que as evidências favoráveis sustentam a proposição *P*, e o segundo elemento (λ) representa o grau em que as evidências desfavoráveis ou contrárias negam ou rejeitam a proposição *P*.

De maneira intuitiva, da associação de uma anotação (μ, λ) a uma proposição *P* é então possível observar no reticulado as seguintes anotações:

(1,0) – indicando a "existência de evidência favorável total e evidência desfavorável nula" − atribui à proposição uma leitura que *P* é verdadeira.

(0,1) – indicando a "existência de evidência favorável nula e evidência desfavorável total" − atribui à proposição uma leitura que *P* é falsa.

(1,1) – indicando a "existência de evidência favorável total e evidência desfavorável total" − atribui à proposição uma leitura que *P* é inconsistente.

(0,0) – indicando a "existência de evidência favorável nula e evidência desfavorável nula" − atribui à proposição uma leitura que *P* é paracompleta ou indeterminada.

De acordo com as anotações acima, deduz-se uma importante propriedade da lógica LPA2v que pode ser considerada equivalente às proposições ¬*P* (μ, λ) e *P* (λ, μ), uma vez que a negação de *P* (μ, λ) é a mesma proposição *P* com graus de evidência invertidos (DA SILVA FILHO, 1999).

Representa-se na figura 5 abaixo o diagrama de análise Paraconsistente. Suas entradas são indicadas com o grau de evidência favorável (crença) representado por μ e grau de evidência desfavorável (descrença) representado por λ. Os valores são submetidos ao reticulado paraconsistente, que após análise

devolve um estado lógico de saída, possibilitando uma conclusão e consequentemente uma tomada de decisão.

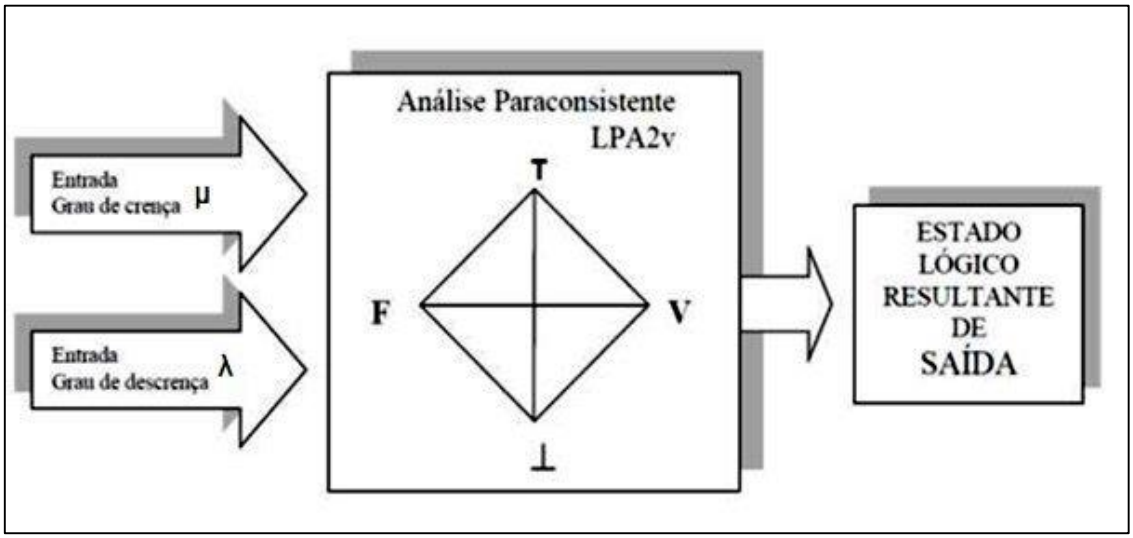


Figura 5: Sistema básico de análise paraconsistente.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

As atribuições dos valores dos graus de crença e de descrença têm como objetivo coletar evidências e solucionar o problema de sinais contraditórios, e por meio de análises, modificar o comportamento do sistema para que a intensidade das contradições diminua (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

2.2.6. Interpretações e Relações algébricas entre o quadrado unitário no plano cartesiano (QUPC) e o reticulado da LPA2v

Com o objetivo de obter uma melhor representação de uma anotação na LPA2v e encontrar uma metodologia de interpretação no seu reticulado representativo τ, algumas interpretações algébricas podem ser feitas para permitir a utilização da Lógica Paraconsistente no tratamento de incertezas (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

O sistema de coordenadas cartesianas é inicialmente adotado para o plano. Os pontos representarão as anotações de uma dada Proposição. Na figura 6 abaixo tem-se o sistema proposto, chamado de Quadrado Unitário no Plano Cartesiano (QUPC).

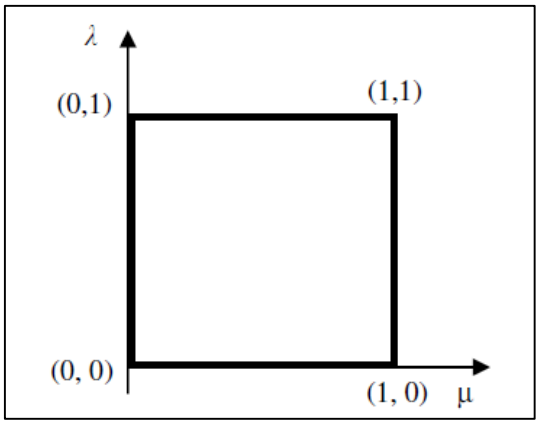


Figura 6: Quadro Unitário no Plano Cartesiano – (QUPC).
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

No QUPC, os valores do Grau de Evidência favorável μ ficam expostos no eixo x, e os valores do Grau de Evidência desfavorável λ no eixo y. Para cada sistema de coordenadas adotado, as anotações (Grau de Evidência favorável μ, Grau de Evidência desfavorável λ) $\mu d_e\ \tau$ são identificadas τ com diferentes pontos no plano.

Assim, associa-se T a (1, 1), ⊥ a (0, 0), F a (0, 1) e V a (1, 0) (ver figura 7).

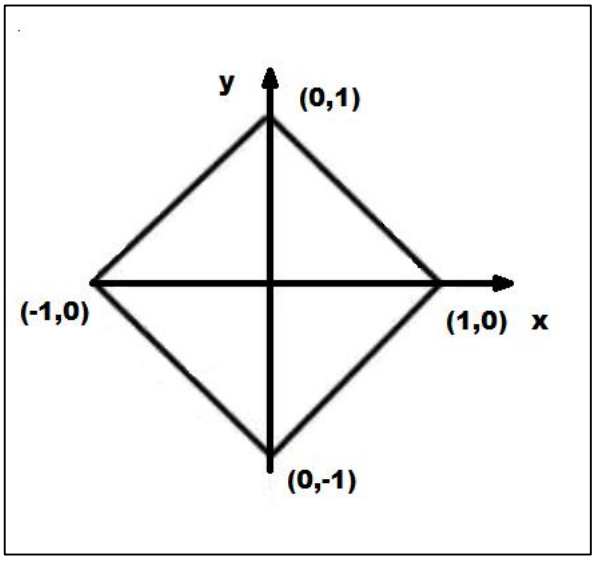


Figura 7: $\mathcal{L}$:- Reticulado τ munido de um novo sistema de coordenadas.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

Na figura 8, é demonstrado o aumento de escala do QUPC de $\sqrt{\mathbf{2}}$, e na figura 9 é demonstrada a rotação de 45º em relação à origem do QUPC.

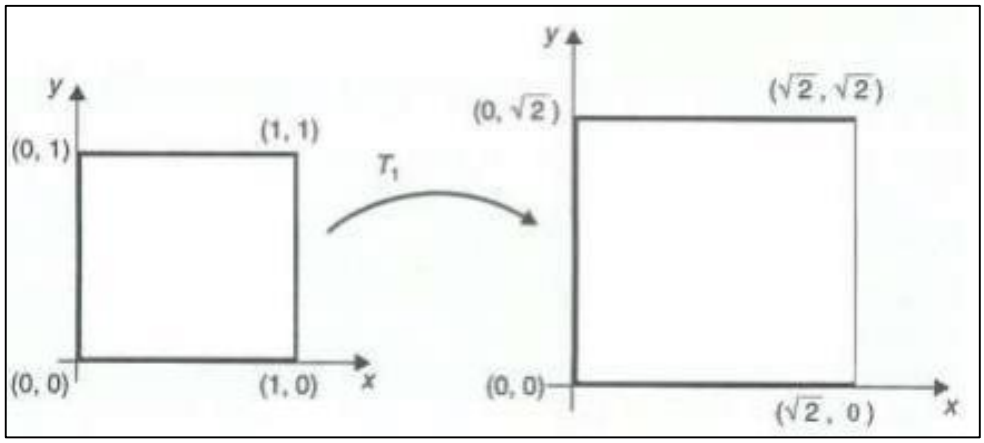


Figura 8: Aumento de escala do QUPC de $\sqrt{2}$.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

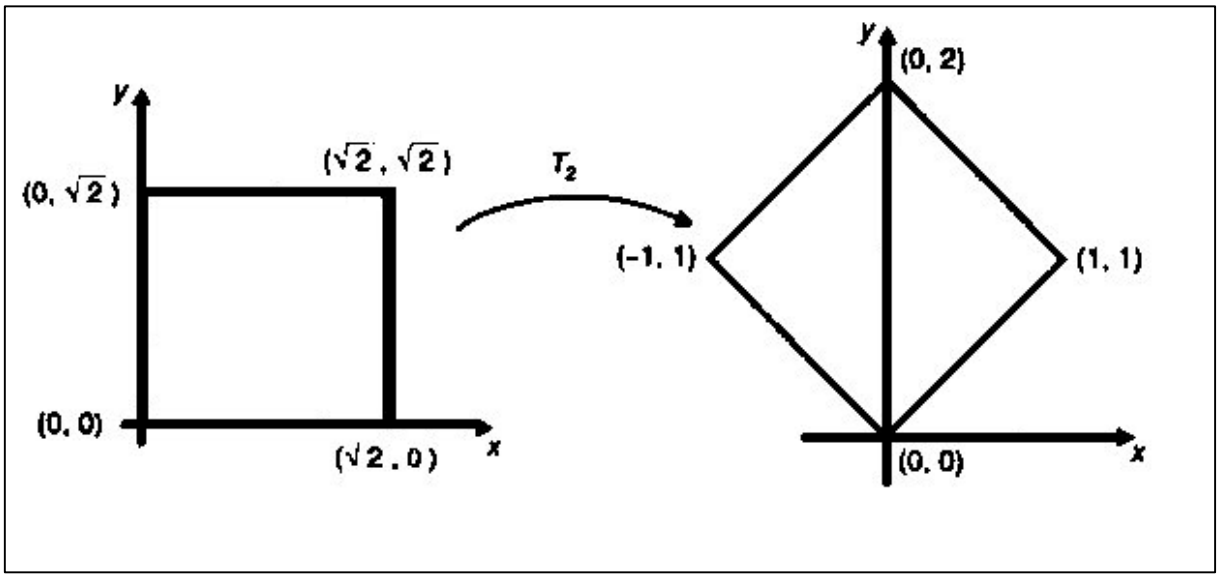


Figura 9: Rotação de 45º em relação à origem.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

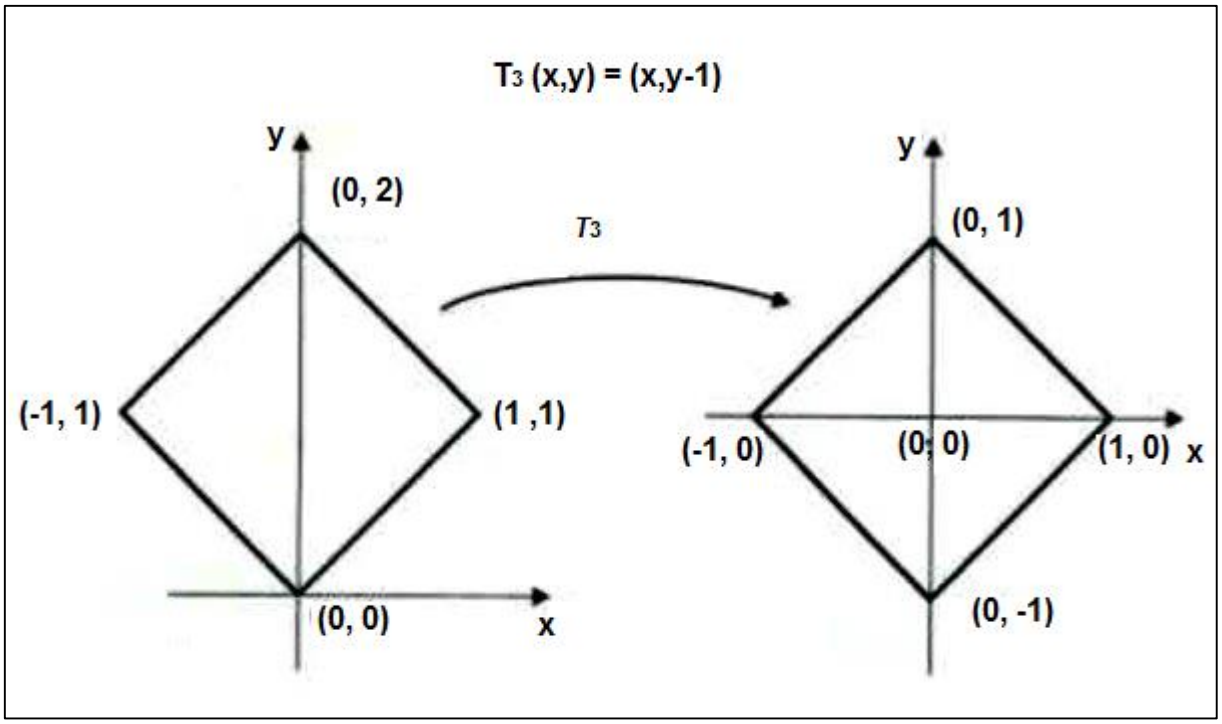


Figura 10: Translação de valores entre QUPC e o reticulado da LPA2v.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

2.2.7. Algoritmo Para-Analisador

Segundo Da Silva Filho, Abe e Lambert-Torres (2008), com os cálculos dos valores dos eixos que compõem o reticulado da LPA2v, é possível delimitar internamente várias regiões de diversos tamanhos e formatos. Obtêm-se assim uma discretização e, a partir das regiões delimitadas do reticulado, extrair estados lógicos resultantes, que serão adquiridos pela interpolação dos Graus de Certeza G_c e de Contradição G_{ct}. Dessa forma, para cada ponto haverá uma única região delimitada no reticulado, equivalente a um estado lógico resultante da análise. A Figura 11 mostra a representação do reticulado LPA2v seccionado em 12 (doze) regiões, ao que, ao final da análise, se obterá como resposta um dos 12 possíveis estados lógicos.

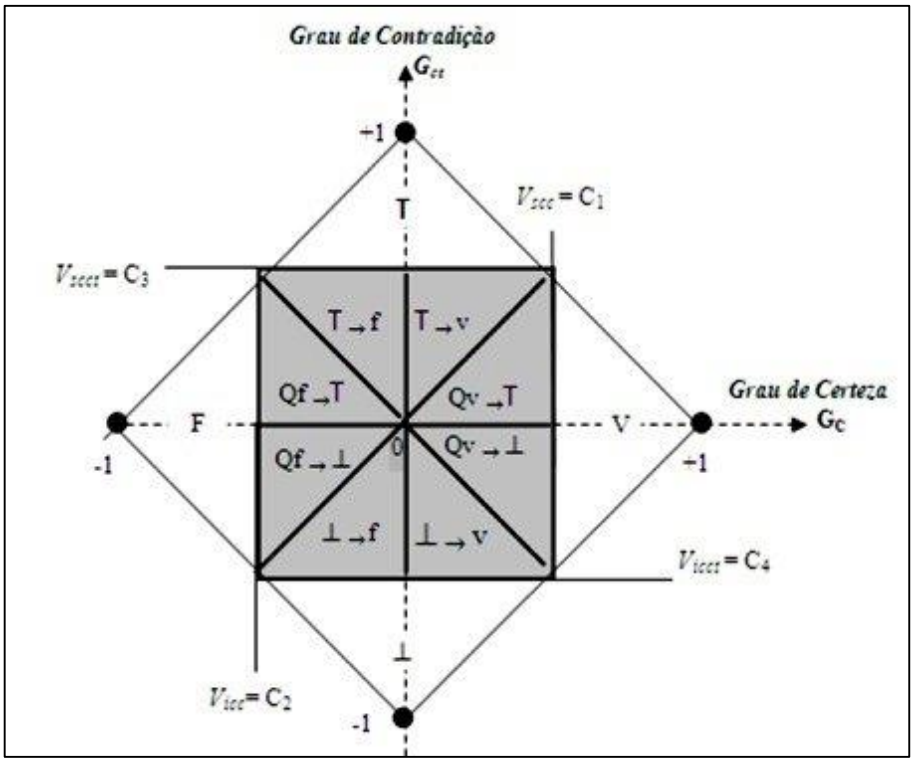


Figura 11: Representação do reticulado da LPA2v seccionado em 12 regiões delimitadas originando 12 estados lógicos resultantes.
(Fonte: DA SILVA FILHO, 2013)

São representados os quatro estados Lógicos Extremos e oito Não Extremos que compõem o reticulado:

Estados Lógicos Extremos

T – Inconsistente

F – Falso

⊥ – Indeterminado ou Paracompleto

V – Verdadeiro

Estados Lógicos Não Extremos

⊥ →f – Indeterminado tendendo ao falso

⊥ →v – Indeterminado tendendo ao verdadeiro

T →f – Inconsistente tendendo ao falso

T →v – Inconsistente tendendo ao verdadeiro

Qv →T – Quase verdadeiro tendendo ao inconsistente

Qf →T – Quase falso tendendo ao inconsistente

Qf →⊥ – Quase falso tendendo ao indeterminado

Qv →⊥ – Quase verdadeiro tendendo ao indeterminado

Tomando por base o reticulado da Figura 11, é possível obter o algoritmo Para-Analisador, que poderá ser implementado em qualquer linguagem de programação, representado a seguir:

Definições dos valores:

Vscc – Valor Superior de Controle de Certeza

Vscct – Valor Superior de Controle de Contradição

Vicc – Valor Inferior de Controle de Certeza

Vicct – Valor Inferior de Controle de Contradição

Variáveis de Entrada:

μ – Grau de Evidência Favorável

λ – Grau de Evidência Desfavorável

Valores de Saída:

S1 = saída discreta

S2a = saída analógica

S2b = saída analógica

Sendo:

$0 \leq \mu \leq 1$

$0 \leq \lambda \leq 1$

Algoritmo:

/* Calcular graus de Certeza e Contradição: */

$Gc = \mu - \lambda$

$Gct = \mu + \lambda - 1$

/* Estados Lógicos Extremos */

Se $Gc \geq Vscc$ então S1 = V

Se $Gc \leq Vicc$ então S1 = F

Se $Gct \geq Vscct$ então S1 = T

Se $Gct \leq Vicct$ então S1 = $\perp$

/*Estados Lógicos Não Extremos */

Para 0 ≤ Gc < Vscc e 0 ≤ Gct < Vscct

Se Gc ≥ Gct então S1 = Qv →T

Senão S1 = T →v

Para 0 ≤ Gc < Vscc e Vicct < Gct ≤ Vscct

Se Gc ≥ |Gct| então S1 = Qv →⊥

Senão S1 = ⊥ →v

Para Vicc < Gc ≤ 0 e Vicct < Gct ≤ Vscct

Se |Gc| ≥ |Gct| então S1 = Qf →⊥

Senão S1 = ⊥ →f

Para Vicc < Gc ≤ 0 e 0 ≤ Gct < Vscct

Se |Gc| ≥ Gct então S1 = Qf →T

Senão S1 = T →f

S2a = Gct

S2b = Gc

/* fim */

2.2.8. Sistema ou nó de Análise Paraconsistente Típico – NAP

Conforme Da Silva Filho, Abe e Lambert-Torres (2008), através da interpretação da LPA2v, foram construídos alguns algoritmos capazes de produzir tratamento e controle de sinais de informações incertas e contraditórias. Esses algoritmos paraconsistentes foram denominados Sistemas ou Nós de análise Paraconsistente (NAP) que, ao serem interligados, compõem redes de análise de tomada de decisão com diferentes topologias.

Nessas redes de análise Paraconsistente, os NAP fazem tratamento de sinais de informação conforme os fundamentos da Lógica Paraconsistente. Suas entradas são alimentadas pelos Graus de Evidência retirados de base de dados de Conhecimento incerto. Os NAP utilizam as equações obtidas da metodologia da LPA2v e obtêm os Graus de Certeza real, $G_{CR,}$ acompanhados de seus respectivos Intervalos de Certeza, φ. Esse processo possibilita conclusões a respeito de determinadas proposições (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

Cada Sistema ou Nó de Análise Paraconsistente (NAP) é capaz de receber evidências e fornecer um valor de certeza acompanhado do seu Intervalo de Certeza. Portanto, considera-se um Nó de análise Paraconsistente como um

Sistema de Análise Paraconsistente que recebe Graus de evidência nas suas entradas e fornece dois valores: O Grau de Certeza real, G_{CR}, e outro que representa o seu Intervalo de Certeza sinalizado, φ(±). Assim, um Nó típico de Análise Paraconsistente é construído pelo “Algoritmo de Análise Paraconsistente da LPA2V” (ver figura 12):

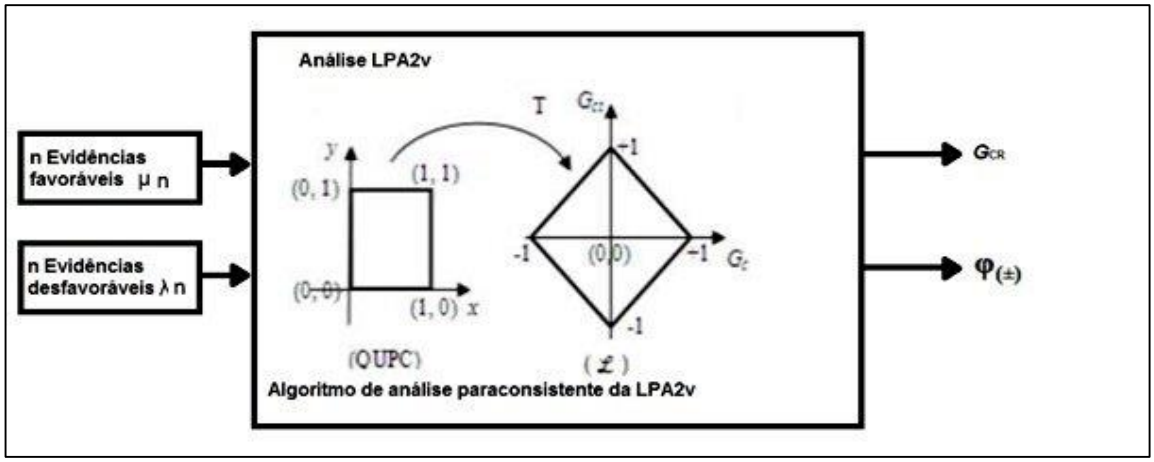


Figura 12: Representação em bloco de um nó de análise Paraconsistente típico.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

A representação simbólica de um NAP é apresentada na Figura 13, em que se tem duas entradas de graus de evidência − favorável e desfavorável − a respeito da proposição analisada, e duas saídas de resultados: o Grau de Certeza Real G_{CR} e o Intervalo de Certeza simbolizado por φ(±).

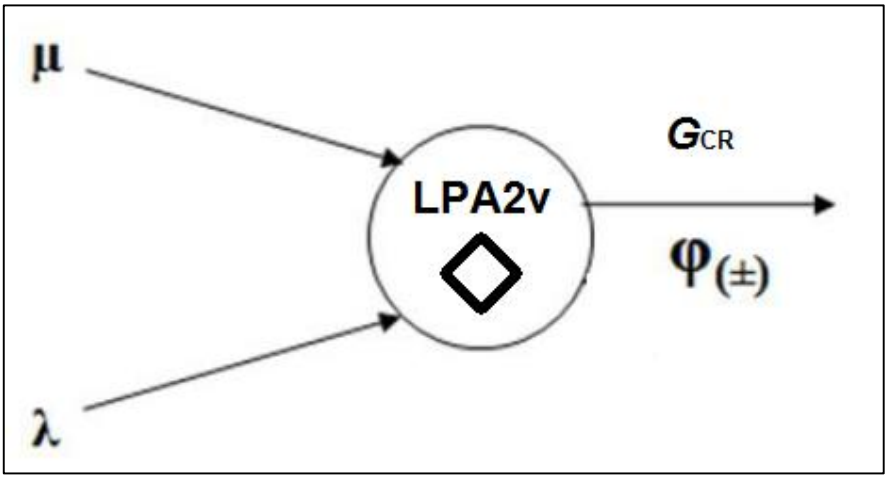


Figura 13: Representação simbólica de um nó de análise Paraconsistente típico.
(Fonte: SOUZA, 2013)

2.2.9. Normas para o nó de análise Paraconsistente – NAP

Segundo Da Silva Filho, Abe e Lambert-Torres (2008), o Nó de Análise Paraconsistente (NAP) recebe uma anotação composta por valores de graus representativos das evidências favoráveis μ e desfavoráveis λ a determinada proposição *P*. Portanto, em cada NAP se fará a análise de apenas uma única proposição, na qual com base nos conceitos da LPA2v, os sinais de evidências devem ser tratados. Desse modo, devem-se considerar algumas normas, como as listadas a seguir:

- Nos sistemas ou Nós de Análise Paraconsistente não devem ser somados, e nem subtraídos, ou considerados as médias dos Graus de Evidência de entrada μ e λ, Graus de Certeza Gc, Graus de Certeza real GCR ou Graus de contradição Gct.
- Não devem ser somados, nem subtraídos ou consideradas as médias dos valores de Intervalos de Certeza.
- Os valores de Grau de Certeza resultante só deverão ser reforçados ou enfraquecidos através de injeção de novas evidências (novos valores) nas entradas do Sistema.
- O reforço ou enfraquecimento do Grau de Certeza resultante por meio de evidências complementares só deve ser efetuado até o limite estabelecido pelo intervalo de certeza. Após esse valor, deverão ser ajustadas as evidências com novos valores para que se diminuam as contradições.
- Os valores dos Graus de Evidências poderão ser ajustados simultaneamente, para aumento ou para diminuição do Grau de Certeza resultante (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

2.2.10. Algoritmos dos Nós de Análise Paraconsistente – NAP

Conforme apresentado por Da Silva Filho, Abe e Lambert-Torres (2008), no algoritmo de análise Paraconsistente é possível acrescer o Cálculo do Grau de Evidência resultante, que quando estiver acima de 0,5 existe uma afirmação a

respeito da proposição analisada. Já valores acima de 0,5 tendendo a 1 indicam que estão no sentido de estabelecer um estado lógico verdadeiro à proposição.

Quando o valor do Grau de Evidência Resultante for igual a 1, existe uma afirmação total à proposição, portanto o estado lógico é totalmente Verdadeiro. Contrapondo a essa afirmação, se o Grau de Evidência resultante for abaixo de 0,5 existe uma refutação à proposição analisada e caso o valor for igual a zero, existe uma negação total à proposição analisada.

Um sistema ou Nó de Análise Paraconsistente contém os algoritmos da LPA2v (ver figura 14):

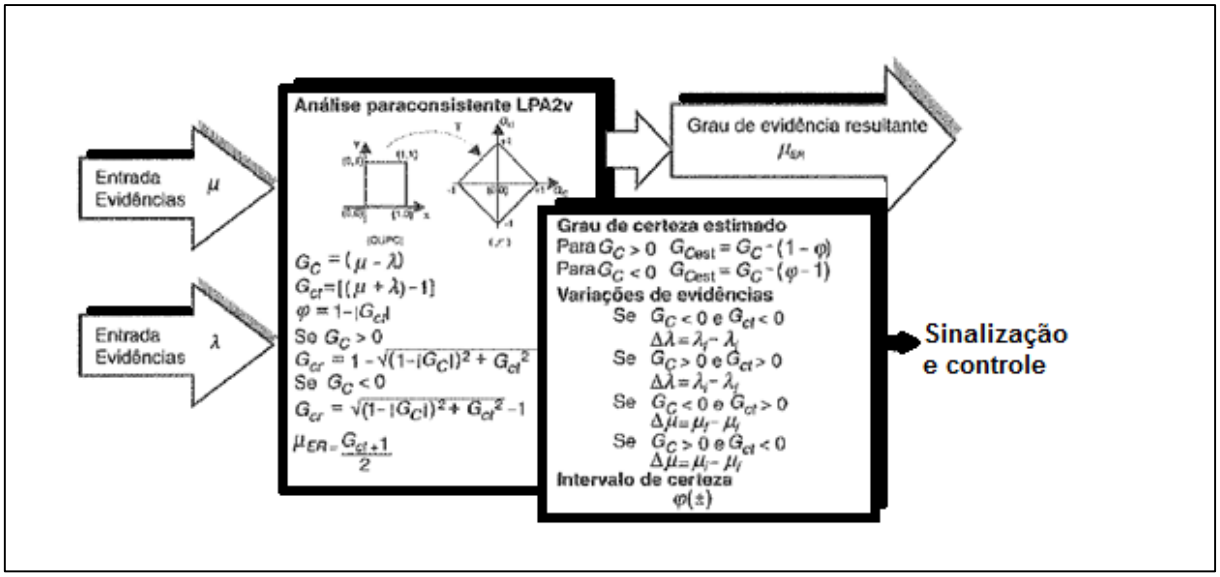


Figura 14: Sistema ou nó de análise Paraconsistente com evidência de entrada, e na saída o grau de certeza transformado em grau de evidência resultante.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

2.2.11. Algoritmo de análise Paraconsistente da LPA2v com saída de grau de evidência resultante real

A rede de análise Paraconsistente apresentada nesse trabalho faz uso do seguinte algoritmo, representado na figura 15, e com os 11 passos descritos na sequência (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008):

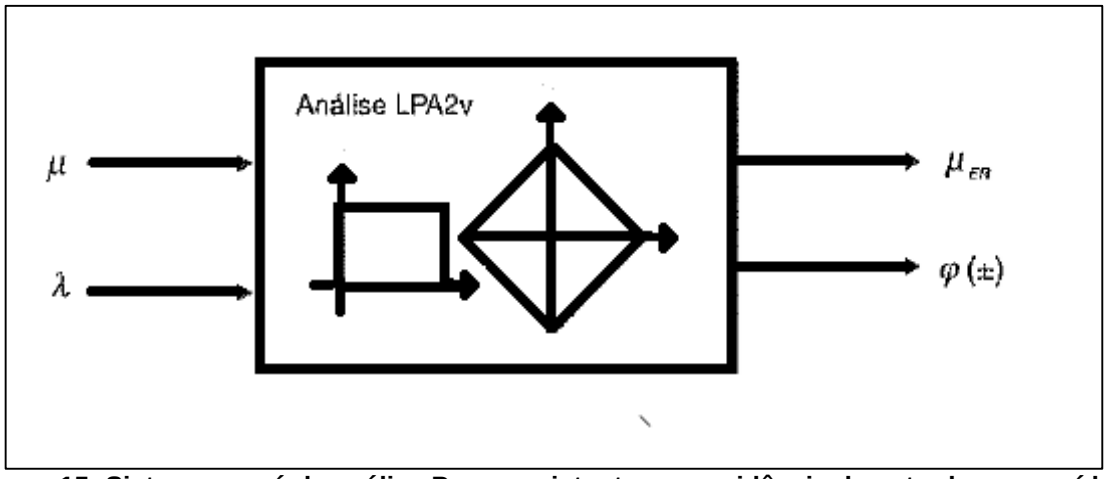


Figura 15: Sistema ou nó de análise Paraconsistente com evidência de entrada, e na saída o grau de certeza transformado em grau de evidência resultante.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

1. Insira os valores de entrada

   μ */ Grau de Evidência favorável (0 ≤ μ ≤1)

   λ */ Grau de Evidência desfavorável (0≤ λ ≤1)

2. Calcule o Grau de Contradição

   Gct = (μ+λ) -1

3. Calcule o Intervalo de Certeza

   Φ = 1 - |Gct|

4. Calcule o Grau de Certeza

   Gc = μ - λ

5. Calcule a distância D

   D = $\sqrt{(1-|Gc|)^2 + Gct^2}$

6. Determine o Grau de Certeza real

   Se Gc > 0, Gcr = (1 – D)

   Se Gc < 0, Gcr = (D – 1)

7. Determine o Sinal da Saída

   Se Φ < 0,25 ou D > 1 então faça

   S1 = 0,5 e S2 = Φ(±): Indefinição e vá para o item 11.

   Senão vá para o próximo item.

8. Determine a sinalização do Intervalo de Grau de Certeza

   Se Gct < 0, sinalize negativo Φ = Φ(-)

   Se Gct > 0, sinalize positivo Φ = Φ(+)

   Se Gct = 0, sinalize zero Φ = Φ(0)

9. Calcule o Grau de Evidência resultante real

$$\mu_{ER} = \frac{G_{CR}+1}{2}$$

10. Apresente os resultados na saída

    Faça S1 = μER e S2 = Φ(±)

11. Fim

2.2.12. Rede de Análise Paraconsistente

Nós de Análises Paraconsistentes (NAP) interligados entre si podem compor uma Rede de Análise Paraconsistente na qual em cada um dos nós é efetuada a análise de uma única proposição. Considerando-se que exista uma Proposição-objeto P0 e que para se tomar decisão sobre ela sejam necessárias as análises de diversas proposições, cada NAP se ocupará de analisar uma única proposição parcial. Portanto, para se obterem os valores suficientes para a tomada de decisão sobre a proposição-objeto, o resultado da análise paraconsistente produzido em cada NAP é combinado com os resultados dos outros NAP. Essas combinações de resultados levam ao estabelecimento de determinado Grau de Certeza à Proposição-objeto, que é a meta final da análise efetuada pela rede (DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008).

Na figura 16 é apresentado um modelo de Rede de Análise Paraconsistente de configuração simples (RAPcs). Nesse modelo, a Proposição-objeto de saída P0 é analisada a partir das informações dadas pelas evidências μEr1 e μEr2, que chegam após as análises paraconsistentes das Proposições P1 e P2 efetuadas pelos NAP.

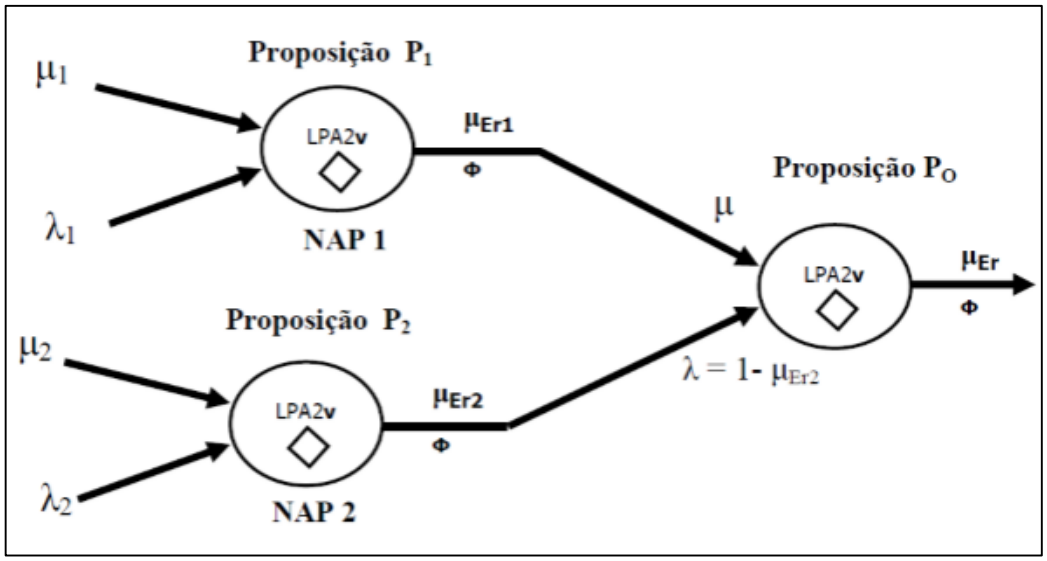


Figura 16: Rede de análise Paraconsistente em configuração simples.
(Fonte: DA SILVA FILHO; ABE; LAMBERT-TORRES, 2008)

2.3. LINGUAGEM DE PROGRAMAÇÃO, AMBIENTE DE DESENVOLVIMENTO E TRATAMENTO DE IMAGENS

Nesta dissertação, serão apresentados a ferramenta para tratamento de imagens, a Linguagem de Programação e o ambiente de desenvolvimento empregados na confecção do *software* utilizado como apoio para demonstrar o modelo desenvolvido ao longo do trabalho. O *software* permite ao usuário selecionar as imagens de números de contêineres; os padrões que servem de base para análise; os dados da varredura *pixel* a *pixel* das imagens selecionadas. Desta maneira, gera relatórios para comparação dos números com os padrões diversos.

2.3.1. Linguagem de Programação

A linguagem de programação escolhida para o desenvolvimento do *software* que auxilia o presente trabalho foi *Java*. Suas características proporcionam a construção de aplicações em diversas áreas da tecnologia, como aplicações distribuídas na web e em rede, aplicativos móveis e portáteis, além de amplas bibliotecas para tratamento de imagens (DEITEL; DEITEL, 2010).

Segundo Deitel e Deitel (2010), o *Java Development Kit* (JDK), possui um conjunto de bibliotecas, apresentadas como *Java* API (*Application Programming Interfaces*). Cada uma delas engloba um grande conjunto de classes organizadas e

distribuídas em pacotes. Os pacotes são organizados de forma a conter classes com funcionalidade básica e relacionadas a um determinado ramo de programação.

No presente trabalho, foi utilizado *Java* em sua versão 1.7.0.8, permitindo uma manipulação detalhada e avançada das imagens digitais de caracteres de contêineres.

2.3.2. Ambiente de Desenvolvimento

O ambiente de desenvolvimento empregado nesse trabalho foi o NetBeans, que é um projeto bem-sucedido de código aberto, com uma grande base de utilizadores e uma crescente comunidade com mais de cem parceiros mundiais (GONÇALVES, 2008). A Sun Microsystems fundou o projeto NetBeans em junho de 2000 e continua a ser o seu principal patrocinador.

O NetBeans IDE é um ambiente de desenvolvimento, uma ferramenta para programadores que permite escrever, compilar, depurar e instalar programas. O IDE é completamente escrito em *Java*, mas pode suportar qualquer linguagem de programação. Existe também um grande número de módulos para estender as funcionalidades do IDE NetBeans. O NetBeans IDE é um produto livre, sem restrições à sua forma de utilização (GONÇALVES, 2008).

2.3.3. Tratamento de Imagem

O tratamento das imagens utilizado nesse experimento foi feito com o uso do *software ImageJ*, que é um programa de plataforma livre para processar e analisar imagens digitais com foco em imagens científicas, inspirado no NIH Image para Macintosh. Esse *software* trabalha em versão web através de um *applet Java* ou pode também operar na versão executável, que trabalha com JDK *Java* 1.5 ou superior (RASBAND, 2012).

Esse *software* é capaz de calcular área e valores de *pixels* em áreas selecionadas pelo usuário, possibilitando mensurar distâncias e ângulos, trata-se de uma forte ferramenta para deixar os números de contêineres dentro de um mesmo padrão para ser identificado pelo programa desenvolvido neste trabalho (RASBAND, 2012).

3. MATERIAIS E MÉTODOS

Para a criação do modelo no auxílio e a identificação de códigos numéricos utilizando a LPA, foi empregado um conjunto de imagens digitais de contêineres. Elas foram coletadas durante a operação de um terminal portuário especializado na movimentação de contêineres, localizado à margem esquerda do porto de Santos (SP), no período de julho de 2013 a agosto de 2015.

Inicialmente, foram selecionadas imagens de contêineres cuja identificação pela numeração fosse difícil, mesmo através da utilização de *software* dedicado a essa funcionalidade. O algoritmo baseado na LPA será utilizado para auxiliar a identificação de códigos numéricos, considerando as possibilidades de captura dos mesmos através de uma fotografia digital, e então relacionará o perfil do algarismo capturado com os padrões existentes, e fará uma indicação do provável número do mesmo.

3.1. PADRÕES DE CARACTERES NUMÉRICOS

Para atingir os objetivos propostos, um conjunto de padrões de imagens de números foram criados usando o *Microsoft Word* com o intuito de servir de base de comparação com os números dos contêineres selecionados. Todos esses padrões são figuras de 50 x 50 *pixels* que foram binarizadas, processo que torna a figura composta apenas por *pixels* brancos e pretos.

O algoritmo baseado na LPA varrerá todos os padrões existentes na base e computará o número de *pixels* pretos existentes em cada uma das regiões a serem analisadas. Então o algoritmo relacionará o perfil do algarismo capturado com os padrões existentes, e fará uma indicação do provável número do mesmo.

A figura 17 a seguir exibe o repositório com 5 imagens padrões para cada um dos algarismos, gerando 50 imagens no padrão 50 x 50 *pixels*.

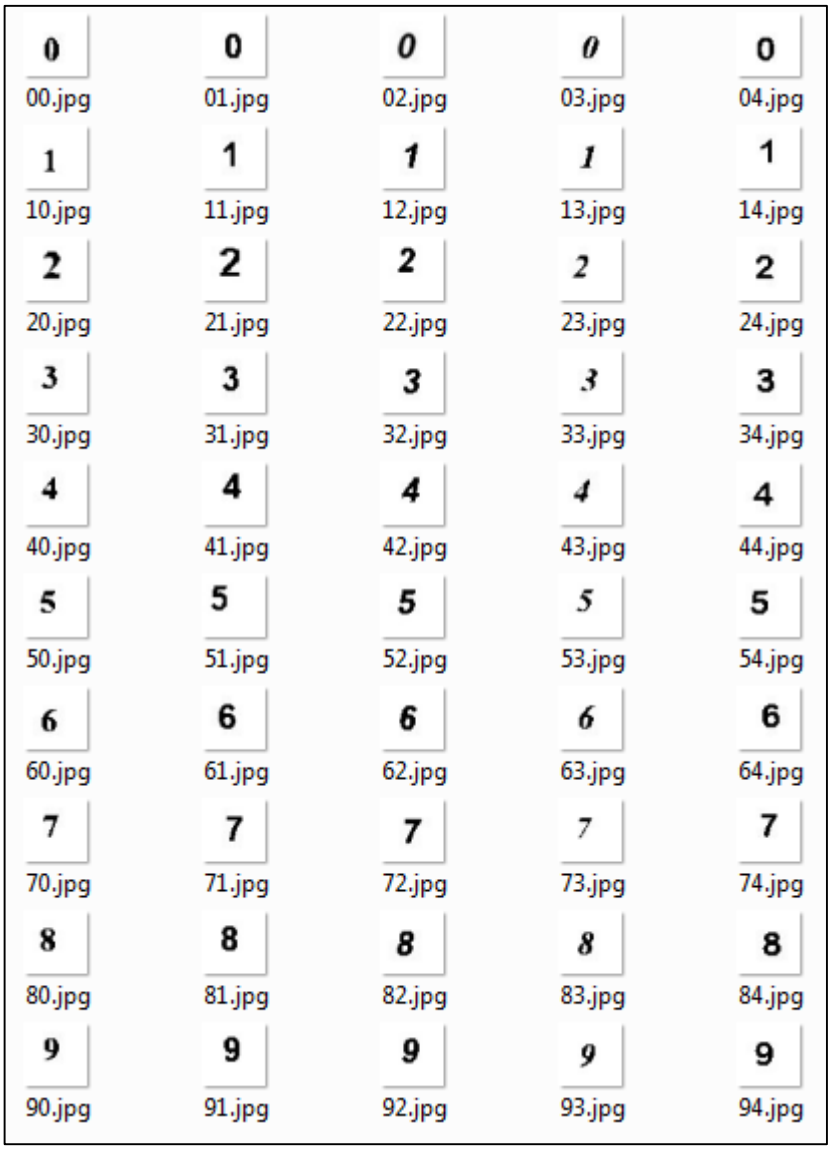


Figura 17: Repositório de padrões de imagens.

3.2. APLICAÇÃO PARA A ANÁLISE DE IMAGENS – ARRANJO PARACONSISTENTE

As imagens selecionadas e os padrões gerados demandam análise *pixel* a *pixel* para correta caracterização do número extraído da identificação do contêiner. Esse processo é fundamental para que seja possível treinar uma das redes de NAPs a fim de identificar o código numérico do mesmo.

Para trabalhar com as informações, uma aplicação *Java* chamada de Arranjo Paraconsistente foi desenvolvida para varrer *pixel* a *pixel* todos os padrões e fazer uma contagem de *pixels* pretos em cada uma das imagens disponíveis no repositório, aplicar as imagens a uma rede de NAPs e determinar qual o número que foi identificado a partir dessa análise.

Na figura 18, tem-se a tela principal do *software* de Arranjo Paraconsistente.

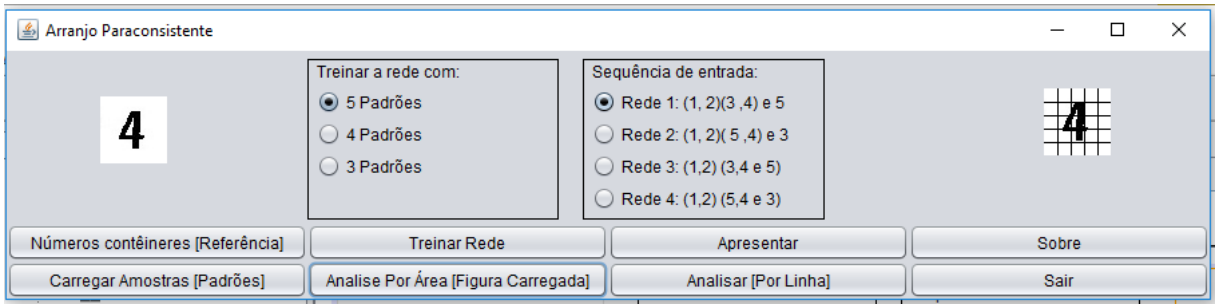


Figura 18: Tela principal do programa Arranjo Paraconsistente.

A aplicação é amigável, e possui duas áreas para apresentação de imagens. Uma à esquerda, com a seleção realizada pelo usuário, e outra à direita, com a aplicação da Análise por área. Convencionou-se dividir a figura em 25 subáreas (ver figura 19) com o tamanho de 10 por 10 *pixels*, resultando em cinco linhas que alimentarão os NAPs.

Figura 19: Código numérico 2 divido em 25 subáreas para análise do *software* Arranjo Paraconsistente.

O *software* permite treinar 4 tipos diferentes de redes com quantidades de padrões variando entre 3 e 5, conforme a escolha do usuário.

Como se pode observar na figura 18, a aplicação possui 8 botões com as seguintes funcionalidades:

- Números contêineres: permite ao usuário acessar o repositório de imagens de caracteres extraídos de contêineres.

- Carregar Amostras: permite ao usuário selecionar uma imagem do repositório de padrões para testar a rede.
- Treinar Rede: permite ao usuário treinar a rede com uma das 4 sequências de entradas pré-definidas. O resultado será obtido submetendo as figuras presentes no repositório de números de contêineres a quantidade de padrões selecionados na aplicação, deixando o seu resultado armazenado na memória.
- Análise Por Área: efetua o cálculo dos valores das 25 subáreas da figura selecionada para submeter à rede.
- Apresentar: gera um relatório com os valores limites das redes.
- Analisar Figura Carregada: faz a análise final da figura carregada, aplicando seus valores ao resultado constante na rede e apresenta o resultado final da aplicação com a sugestão dos possíveis números identificados.
- Sobre: apresenta um breve descritivo da aplicação e seu autor.
- Sair: permite ao usuário encerrar a aplicação.

O resultado final, após o usuário carregar a imagem, treinar a rede, fazer a análise por área e solicitar o veredito final ao *software*, é apresentado na figura 20.

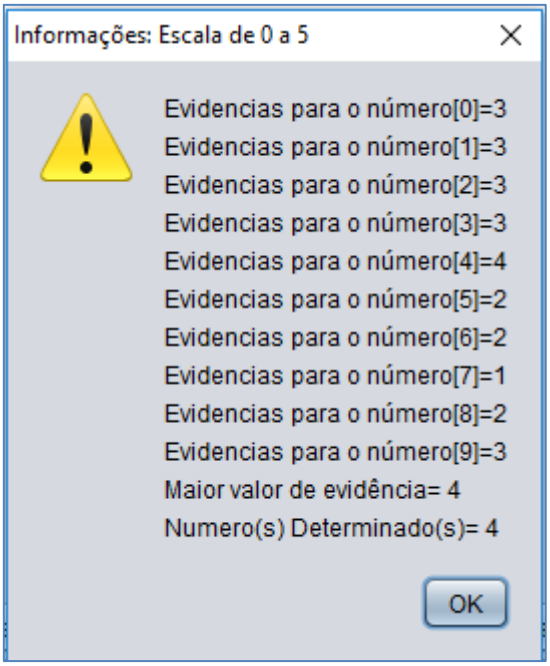


Figura 20: Tela com resultado da análise após uso da Rede no programa Arranjo Paraconsistente.

A sequência de uso do *software* é apresentado na figura 21, e os códigos *Java* serão apresentados nos quadros descritos em cada uma das sequências.

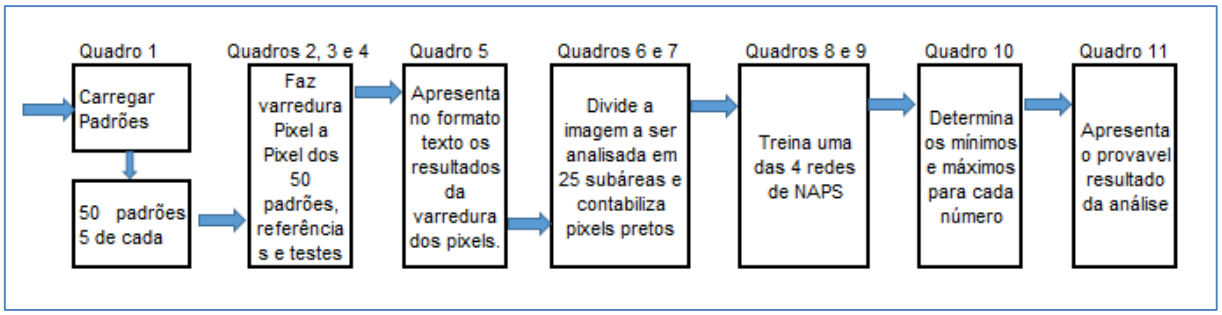


Figura 21: Sequência dos quadros e códigos no *software* Arranjo Paraconsistente.

No quadro 1, a imagem a ser testada é carregada com o uso do algoritmo:

Quadro 1 – Algoritmo para carregar a imagem a ser trabalhada.

```
private void CarregarImagemReferencia(){
    try{
        JFileChooser chooser = new JFileChooser("c:\\OCR\\NumerosContainer\\Tratados");
        FileNameExtensionFilter filter = new FileNameExtensionFilter("JPG Images","jpg");
        chooser.setFileFilter(filter);
        int returnVal = chooser.showOpenDialog(this);
        if(returnVal == JFileChooser.APPROVE_OPTION){
             ImagemCarregada = chooser.getSelectedFile().toString();
             PadraoTreinar = chooser.getSelectedFile().getName().substring(0,1);
             System.out.println("Achou:"+PadraoTreinar);
             System.out.println(ImagemCarregada);
             Image img;
             Image imgCarregada;
             img = ImageIO.read(new File(ImagemCarregada));
             imgCarregada = ImageIO.read(new File(ImagemCarregada));
             BufferedImage imagem = (BufferedImage)img;
             BufferedImage imagemCarregada = (BufferedImage)imgCarregada;
             img = (Image)imagem;
             imageLabel.setIcon(new ImageIcon(img));
             imageLabel.setSize(imagem.getWidth(), imagem.getHeight());
             imgCarregada = (Image) imagemCarregada;
             imageAnaliseArea.setIcon(new ImageIcon(imgCarregada));
             imageAnaliseArea.setSize(imagemCarregada.getWidth(), imagemCarregada.getHeight());
             repaint();
             isImagemCarregada = true;
        }
    }catch(HeadlessException | IOException e){
        JOptionPane.showMessageDialog(this, "Erro:"+e.getMessage(),"Erro", JOptionPane.ERROR_MESSAGE);
    }
}
```

No trecho de código (quadro 2), a extração de todos os *pixels* das 50 imagens que servem de padrões para a Rede de Análise Paraconsistente é realizada. Isso é feito no momento em que o *software* é inicializado, já que a quantidade de padrões que é escolhida pelo usuário e a rede que será utilizada para obter o resultado não

interfere no processo, deixando disponível em memória os valores para análise nas etapas posteriores.

Quadro 2 – Algoritmo para captura de *pixels* das imagens padrões.

```
private void ObterSubAreas() throws IOException{
    try{
        // ******************************
        // Carga dos Padrões - 50 figuras
        // ******************************
        File folder = new File("C:\\OCR\\Padroes");
        File[] listOfFiles = folder.listFiles();
        for (int i = 0; i < listOfFiles.length; i++) {
            if (listOfFiles[i].isFile()) {
                ImagemCarregada = folder.toString() + "\\" +listOfFiles[i].getName();
                File img = new File(ImagemCarregada);
                BufferedImage in = ImageIO.read(img);
                Raster raster = in.getRaster();
                int[] pixel = new int[3];
                int pretos, outros, altura,largura;
                for(int h=0;h<in.getHeight();h++){
                    for(int w=0;w<in.getWidth();w++)
                    {
                        raster.getPixel(w,h,pixel);
                        arraySubAreaPixelsPadroes[i][h][w][0]=pixel[0];
                        arraySubAreaPixelsPadroes[i][h][w][1]=pixel[1];
                        arraySubAreaPixelsPadroes[i][h][w][2]=pixel[2];
                    }
                }

                for(int matrizaltura=0; matrizaltura<5;matrizaltura++){
                    for(int matrizlargura = 0;matrizlargura<5;matrizlargura++){
                        pretos = 0;
                        outros = 0;
                        for(int submatrizaltura=0; submatrizaltura<10;submatrizaltura++){
                            for(int submatrizlargura = 0;submatrizlargura<10;submatrizlargura++){
                                altura = (matrizaltura*10)+submatrizaltura;
                                largura = (matrizlargura*10)+submatrizlargura;
                                pixel[0] = 0;
                                pixel[1] = 0;
                                pixel[2] = 0;
                                raster.getPixel(largura,altura,pixel);
                                if ((pixel[0] == 0) && (pixel[1] == 0) && (pixel[2] == 0)){
                                  pretos++;
                                }else{
                                  outros++;
                                }
                            }
                        }
                        arraySubAreasPadroes[i][(matrizaltura*5)+matrizlargura]=pretos;
                    }
                }
              repaint();
            } else if (listOfFiles[i].isDirectory()) {
                System.out.println("Directory " + listOfFiles[i].getName());
            }
        }
```

Com base no trecho de código (quadro 3), é executada a extração de todos os *pixels* das 10 imagens que servem de referência para números de contêineres, que alimentarão a Rede de Análise Paraconsistente. Este processo também é feito no momento em que o *software* é inicializado.

Quadro 3 – Algoritmo para captura de *pixels* das imagens referência.

```
// **********************************
// Carga das Referências - 10 figuras
// **********************************

folder = new File("C:\\OCR\\NumerosContainer\\Tratados");
listOfFiles = folder.listFiles();
for (int i = 0; i < listOfFiles.length; i++) {
    if (listOfFiles[i].isFile()) {
        ImagemCarregada = folder.toString() + "\\" +listOfFiles[i].getName();
        File img = new File(ImagemCarregada);
        BufferedImage in = ImageIO.read(img);
        Raster raster = in.getRaster();
        int[] pixel = new int[3];
        int pretos, outros, altura,largura;
        for(int h=0;h<in.getHeight();h++){
            for(int w=0;w<in.getWidth();w++)
            {
                raster.getPixel(w,h,pixel);

                arraySubAreaPixelsReferencias[i][h][w][0]=pixel[0];
                arraySubAreaPixelsReferencias[i][h][w][1]=pixel[1];
                arraySubAreaPixelsReferencias[i][h][w][2]=pixel[2];
            }
        }

        for(int matrizaltura=0; matrizaltura<5;matrizaltura++){
            for(int matrizlargura = 0;matrizlargura<5;matrizlargura++){
                pretos = 0;
                outros = 0;
                for(int submatrizaltura=0; submatrizaltura<10;submatrizaltura++){
                    for(int submatrizlargura = 0;submatrizlargura<10;submatrizlargura++){
                        altura = (matrizaltura*10)+submatrizaltura;
                        largura = (matrizlargura*10)+submatrizlargura;
                        pixel[0] = 0;
                        pixel[1] = 0;
                        pixel[2] = 0;
                        raster.getPixel(largura,altura,pixel);
                        if ((pixel[0] == 0) && (pixel[1] == 0) && (pixel[2] == 0)){
                          pretos++;
                        }else{
                          outros++;
                        }
                    }
                }
                arraySubAreasReferencias[i][(matrizaltura*5)+matrizlargura]=pretos;
            }
        }
      repaint();
    } else if (listOfFiles[i].isDirectory()) {
        System.out.println("Directory " + listOfFiles[i].getName());
    }
}
```

Para finalizar o processo inicial de extração de *pixels* das imagens, o trecho de código (quadro 4) é executado no momento da inicialização do aplicativo, extraindo os *pixels* das imagens que servem de teste para Rede de Análise Paraconsistente.

Quadro 4 – Algoritmo para captura de *pixels* das imagens para teste.

```
        // **********************************
        // Carga dos testes - 10 figuras
        // **********************************
        folder = new File("C:\\OCR\\NumerosContainer\\Tratados2");
        listOfFiles = folder.listFiles();
        for (int i = 0; i < listOfFiles.length; i++) {
            if (listOfFiles[i].isFile()) {
                ImagemCarregada = folder.toString() + "\\" +listOfFiles[i].getName();
                File img = new File(ImagemCarregada);
                BufferedImage in = ImageIO.read(img);
                Raster raster = in.getRaster();
                int[] pixel = new int[3];
                int pretos, outros, altura,largura;
                for(int h=0;h<in.getHeight();h++){
                    for(int w=0;w<in.getWidth();w++)
                    {
                        raster.getPixel(w,h,pixel);

                        arraySubAreaPixelsTeste[i][h][w][0]=pixel[0];
                        arraySubAreaPixelsTeste[i][h][w][1]=pixel[1];
                        arraySubAreaPixelsTeste[i][h][w][2]=pixel[2];
                    }
                }
                for(int matrizaltura=0; matrizaltura<5;matrizaltura++){
                    for(int matrizlargura = 0;matrizlargura<5;matrizlargura++){
                        pretos = 0;
                        outros = 0;
                        for(int submatrizaltura=0; submatrizaltura<10;submatrizaltura++){
                            for(int submatrizlargura = 0;submatrizlargura<10;submatrizlargura++){
                                altura = (matrizaltura*10)+submatrizaltura;
                                largura = (matrizlargura*10)+submatrizlargura;
                                pixel[0] = 0;
                                pixel[1] = 0;
                                pixel[2] = 0;
                                raster.getPixel(largura,altura,pixel);
                                if ((pixel[0] == 0) && (pixel[1] == 0) && (pixel[2] == 0)){
                                  pretos++;
                                }else{
                                  outros++;
                                }
                            }
                        }
                        arraySubAreasTeste[i][(matrizaltura*5)+matrizlargura]=pretos;
                    }
                }
              repaint();
            } else if (listOfFiles[i].isDirectory()) {
                System.out.println("Directory " + listOfFiles[i].getName());
            }
        }

    }catch(Exception e){
        JOptionPane.showMessageDialog(this, "Erro:"+e.getMessage(),"Erro", JOptionPane.ERROR_MESSAGE);
    }
    repaint();
}
```

De acordo com a descrição do código demonstrado nos quadros 2, 3 e 4, todas as imagens do repositório de padrões, referência e teste terão seus *pixels* carregados em memória e apresentados em forma de texto, tornando-se disponível para inserção em planilha e geração de relatórios (quadro 5) e gráficos como apresentado no Apêndice A.

Quadro 5 – Relatório com a proporção de *pixels* pretos por área da figura.

```
Figura [1] Sub-área[1]:0/100=       0.0
Figura [1] Sub-área[2]:0/100=       0.0
Figura [1] Sub-área[3]:0/100=       0.0
Figura [1] Sub-área[4]:0/100=       0.0
Figura [1] Sub-área[5]:0/100=       0.0
Figura [1] Sub-área[6]:0/100=       0.0
Figura [1] Sub-área[7]:0/100=       0.0
Figura [1] Sub-área[8]:0/100=       0.0
Figura [1] Sub-área[9]:0/100=       0.0
Figura [1] Sub-área[10]:0/100=       0.0
Figura [1] Sub-área[11]:0/100=       0.0
Figura [1] Sub-área[12]:9/100=       0.09
Figura [1] Sub-área[13]:20/100=       0.2
Figura [1] Sub-área[14]:0/100=       0.0
Figura [1] Sub-área[15]:0/100=       0.0
Figura [1] Sub-área[16]:0/100=       0.0
Figura [1] Sub-área[17]:7/100=       0.07
Figura [1] Sub-área[18]:16/100=       0.16
Figura [1] Sub-área[19]:0/100=       0.0
Figura [1] Sub-área[20]:0/100=       0.0
Figura [1] Sub-área[21]:0/100=       0.0
Figura [1] Sub-área[22]:0/100=       0.0
Figura [1] Sub-área[23]:0/100=       0.0
Figura [1] Sub-área[24]:0/100=       0.0
Figura [1] Sub-área[25]:0/100=       0.0
```

Para a análise das imagens dos números pelas Redes de Análise Paraconsistente, cada imagem com número é dividida em 25 subáreas que são submetidas ao *software* para montar uma das 4 redes de análise Paraconsistente capazes de gerar resultados que servirão de base para determinação de qual é o número que é apresentado para a aplicação.

No Quadro 6, o trecho de código que divide a figura a ser analisada em 25 subáreas e apresenta novamente com a linha de grade para dar uma ideia ao usuário dos conteúdos das subáreas.

Quadro 6 – Algoritmo para dividir a imagem em 25 subáreas e contabilizar *pixels* pretos.

```
private void AnalisePorArea(){
    if(!isImagemCarregada){
        JOptionPane.showMessageDialog(this, "Obrigatório selecionar uma figura antes de solicitar o análise!",
        "Erro: Ação não permitida", JOptionPane.ERROR_MESSAGE);
    }else{
        try{
            repaint();
            int[] pixel = new int[3];
            int pretos, outros, h,w;
            double l1,l2,l3,l4;

            for (int i = 0; i < arraySubAreaPixelsPadroes.length; i++) {
                resultados[i]=0;
            }

            ImageIcon ico = (ImageIcon)imageLabel.getIcon();
            BufferedImage imagem = (BufferedImage)((Image) ico.getImage());
            Raster raster = imagem.getRaster();

            for(int matrizaltura=0; matrizaltura<5;matrizaltura++){
                for(int matrizlargura = 0;matrizlargura<5;matrizlargura++){
                    pretos = 0;
                    outros = 0;
                    for(int submatrizaltura=0; submatrizaltura<10;submatrizaltura++){
                        for(int submatrizlargura = 0;submatrizlargura<10;submatrizlargura++){

                            h = (matrizaltura*10)+submatrizaltura;
                            w = (matrizlargura*10)+submatrizlargura;
                            pixel[0] = 0;
                            pixel[1] = 0;
                            pixel[2] = 0;

                            raster.getPixel(w,h,pixel);

                            if ((pixel[0] == 0) && (pixel[1] == 0) && (pixel[2] == 0)){
                                pretos++;
                            }else{
                                outros++;
                            }
                        }
                    }
                    subAreasComparar[(matrizaltura*5)+matrizlargura]=pretos;

                }
            }
            repaint();
            ImageIcon icoAnaliseArea = (ImageIcon)imageAnaliseArea.getIcon();
            BufferedImage imagemAnaliseArea = (BufferedImage)((Image) icoAnaliseArea.getImage());
```

O trecho de código (Quadro 7) submete as áreas da figura a ser analisada a uma das 4 redes disponíveis, conforme a escolha feita pelo usuário no *software* Arranjo Paraconsistente.

Quadro 7 – Segunda parte do método Análise por Área respeitando a rede escolhida.

```
            for(int vl = 0; vl < 50; vl++)
            {
                imagemAnaliseArea.setRGB(10, vl, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(20, vl, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(30, vl, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(40, vl, Color.BLACK.getRGB() );

                imagemAnaliseArea.setRGB(vl, 10, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(vl, 20, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(vl, 30, Color.BLACK.getRGB() );
                imagemAnaliseArea.setRGB(vl, 40, Color.BLACK.getRGB() );
            }

            int padraotreinar = Integer.parseUnsignedInt(PadraoTreinar);
            for(int i=0;i<25;i++){
                arrayResultadoNAPSNumeroComp[i]= NAP((double)arraySubAreasReferencias[padraotreinar][i]/100,(double)subAreasComparar[i]/100);
            }

            for(int i=0;i<5;i++){
                int fator = 5 * i;
                l1 = NAP(arrayResultadoNAPSNumeroComp[fator],arrayResultadoNAPSNumeroComp[fator+1]);
                switch (sequenciaRede) {
                    case 2:
                        l2 = NAP(arrayResultadoNAPSNumeroComp[fator+2],arrayResultadoNAPSNumeroComp[fator+3]);
                        l3 = NAP(l2,arrayResultadoNAPSNumeroComp[(fator+4)]);
                        l4 = NAP(l1,l3);
                        break;
                    case 1:
                        l2 = NAP(arrayResultadoNAPSNumeroComp[fator+4],arrayResultadoNAPSNumeroComp[fator+3]);
                        l3 = NAP(l1,l2);
                        l4 = NAP(l3,arrayResultadoNAPSNumeroComp[(fator+2)]);
                        break;
                    case 0:
                        l2 = NAP(arrayResultadoNAPSNumeroComp[fator+2],arrayResultadoNAPSNumeroComp[fator+3]);
                        l3 = NAP(l1,l2);
                        l4 = NAP(l3,arrayResultadoNAPSNumeroComp[(fator+4)]);
                        break;
                    default:
                        System.out.println("REDE 3");
                        l2 = NAP(arrayResultadoNAPSNumeroComp[fator+4],arrayResultadoNAPSNumeroComp[fator+3]);
                        l3 = NAP(l2,arrayResultadoNAPSNumeroComp[(fator+4)]);
                        l4 = NAP(l1,l3);
                        break;
                }
                arrayLinhasNumeroComp[i] = l4;
            }
            JOptionPane.showMessageDialog(this, "Análise por Área concluída! "
                , "Informações: Análise por área", JOptionPane.WARNING_MESSAGE);

        }catch(NumberFormatException | HeadlessException e){
             JOptionPane.showMessageDialog(this, "Erro:"+e.getMessage(),"Erro", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

3.3. REDES PARA AUXILIAR A IDENTIFICAÇÃO DE CÓDIGO NUMÉRICO DE NÚMEROS DE CONTÊINERES

Ao selecionar uma das imagens dos números extraídos dos contêineres, o usuário pode submeter a uma das 4 redes de NAPs escolhida no *software* Arranjo Paraconsistente. Ao selecionar a rede 1, o valor de μ_{1L1} sempre será o número proporcional de *pixels* pretos obtido pelo *software* na primeira das 25 áreas do número de contêiner referência, e λ_{1L1} o valor equivalente em *pixels* pretos da mesma região para um dos 5 padrões existentes do mesmo número. Para efeitos de cálculo, o valor de λ sempre será o complemento de μ obtido na mesma região em um dos 5 padrões, ou seja (1- μ), para alimentar cada um dos nós.

Os valores após o L representam a linha que está em análise, exemplo L1, L2, L3, L4 e L5 de cada um dos números. Já os valores de μ e λ acompanharão a sequência de 1 a 5 de cada uma das linhas. Na figura 22 tem-se a representação da rede 1 de NAPS para a primeira linha de um número qualquer selecionado,

representando as suas entradas os valores μ_{1L1}, λ_{1L1}, μ_{2L1}, λ_{2L1}, μ_{3L1}, λ_{3L1}, μ_{4L1}, λ_{4L1}, μ_{5L1} e λ_{5L1}.

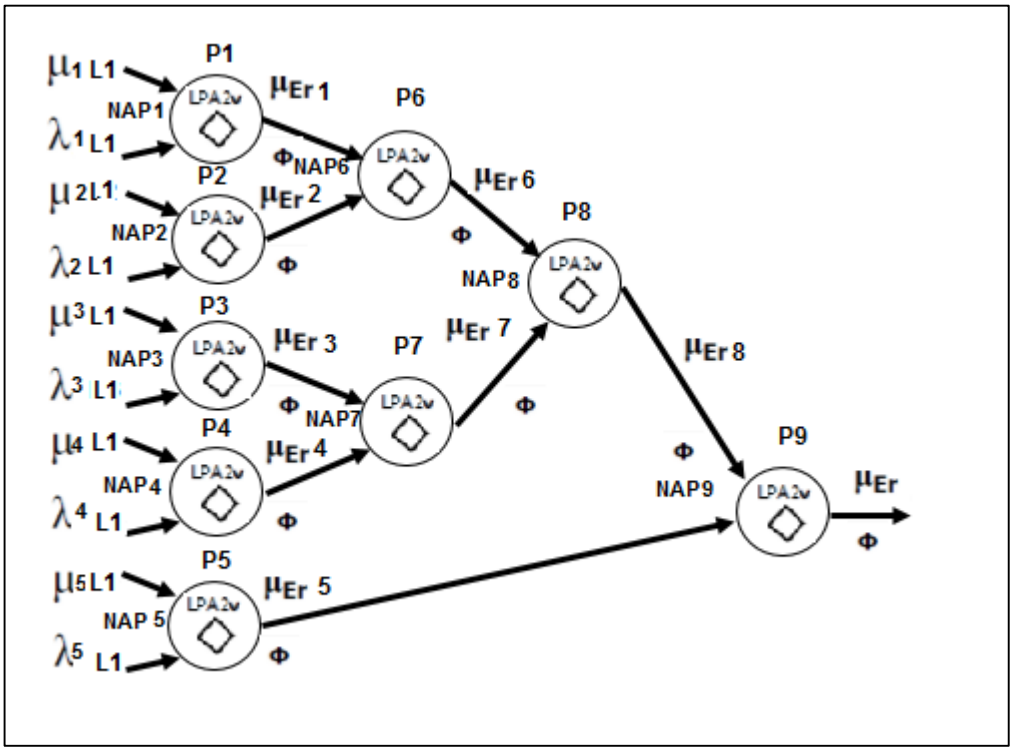


Figura 22: Rede 1 de Análise Paraconsistente empregada neste trabalho.

A Rede 2 é uma alteração da Rede 1, onde os valores de entrada do NAP 5 e NAP 3 foram invertidos, resultando na arquitetura de rede 2 apresentada na figura 23.

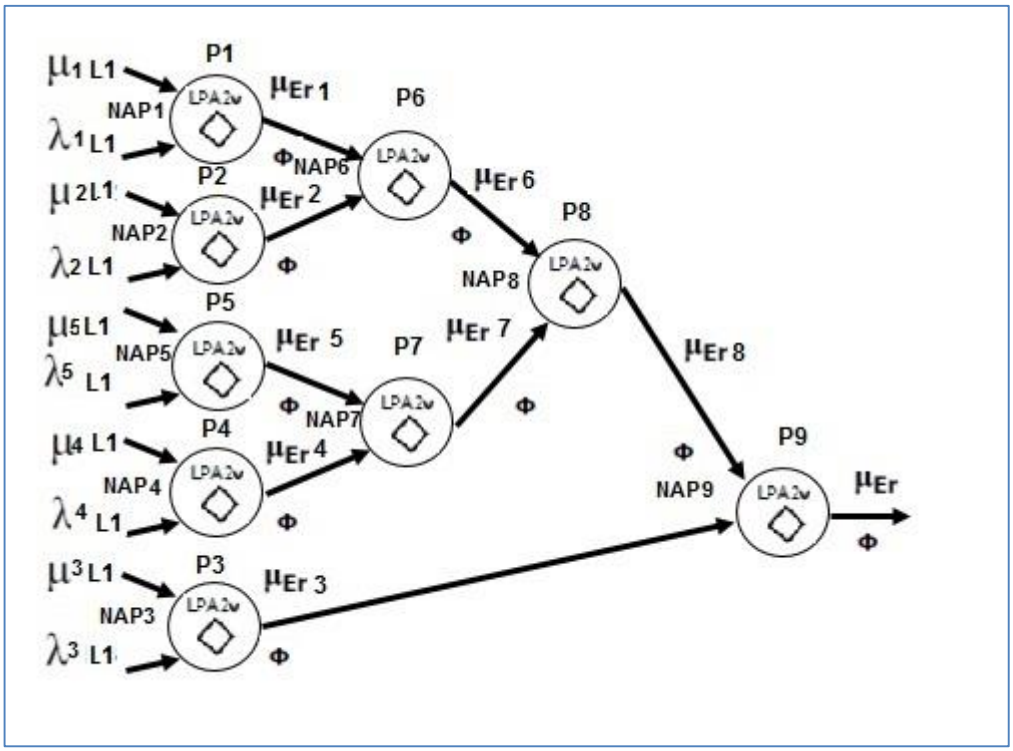


Figura 23: Rede 2 de Análise Paraconsistente empregada neste trabalho.

Na figura 24 é apresentada a arquitetura de rede 3, onde os valores resultante dos NAPS 3 e 4 são submetidos ao NAP 7, que por sua vez recebe os valores resultantes do NAP 5 e torna os valores do NAP 5 mais significativo nos cálculos.

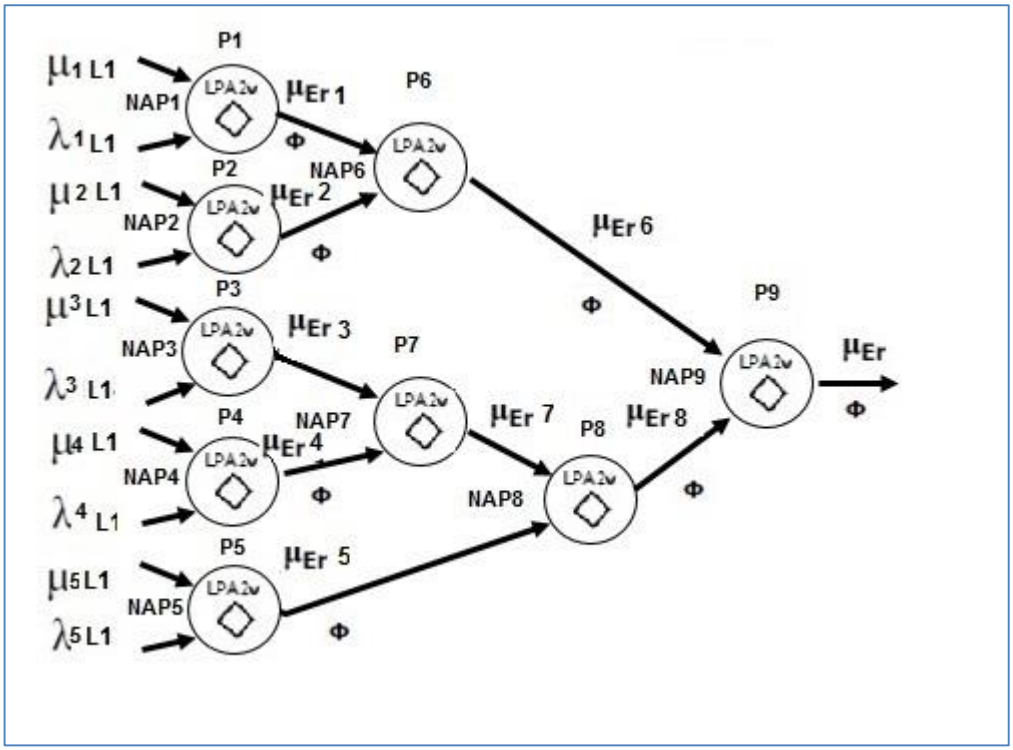


Figura 24: Rede 3 de Análise Paraconsistente empregada neste trabalho.

A última configuração de rede emprega a arquitetura similar a apresentada na rede 3 da figura 24, porém com a inversão dos valores de entrada dos NAPS 5 e 3 (ver figura 25).

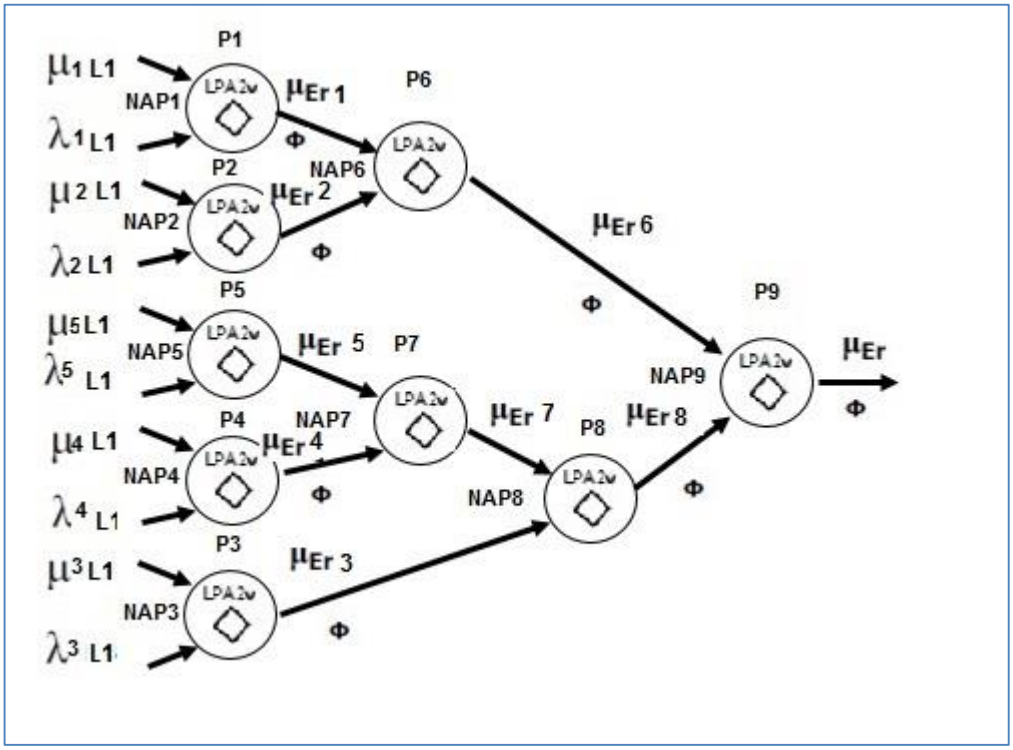


Figura 25: Rede 4 de Análise Paraconsistente empregada neste trabalho.

O código fonte desenvolvido em *Java* para cálculo do NAP é apresentado no quadro 8:

Quadro 8 – Algoritmo do NAP implementado em Java.

```
private double NAP(double mi1,double mi2){
    double lambda = 1 - mi2;
    double Gct = (mi1 + lambda) - 1;
    double Ict = 1 - Math.abs(Gct);
    double Gc = mi1 - lambda;
    double D = Math.sqrt(Math.pow((1 - Math.abs(Gc)),2) + Math.pow(Gct,2));
    double Gcr;
    if (Gc > 0){
       Gcr = 1 - D;
    }else if(Gc <0){
       Gcr = D - 1;
    }else{
       Gcr = 0;
    }
    boolean indefinicao=false;
    if((Ict<=0.25)||(Ict>1)) indefinicao = true;
    double IntervaloCerteza;
    if(Gct<0){
      IntervaloCerteza = -Ict;
    }else if(Gct>0){
       IntervaloCerteza = Ict;
    }else{
       IntervaloCerteza = 0;
    }
    double Mer = (Gcr + 1) / 2;
    return Mer;
}
```

Para exemplificar o trabalho da rede de NAPs apresentadas nas figuras 22, 23, 24 e 25 tomam-se como exemplo os valores presentes na tabela 3 com as 25 regiões calculadas para a figura do caractere 1. Os valores de cada região estão sempre compreendidos no intervalo entre 0 e 1 ($0 \leq \mu \leq 1$) e ($0 \leq \lambda \leq 1$), sendo em $\mu 1$ (grau de evidência favorável) os valores do caractere referência e $\mu 2$ os valores do primeiro padrão de comparação para o caractere 1. A terceira coluna representa o valor de λ (grau de evidência desfavorável), que é $\mu 2$ complementado:

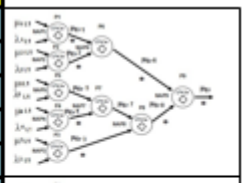
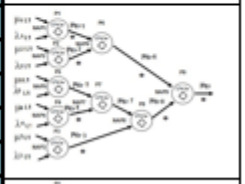
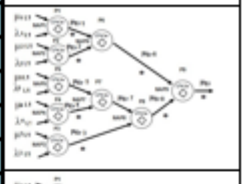
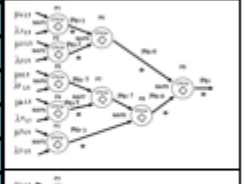
Linhas	Subárea	μ1	μ2	λ	
1	1	0,0000000	0,0000000	1,0000000	
	2	0,0000000	0,0000000	1,0000000	
	3	0,0000000	0,0000000	1,0000000	
	4	0,0000000	0,0000000	1,0000000	
	5	0,0000000	0,0000000	1,0000000	
2	6	0,0000000	0,0000000	1,0000000	
	7	0,0100000	0,0000000	1,0000000	
	8	0,2700000	0,0000000	1,0000000	
	9	0,0000000	0,0000000	1,0000000	
	10	0,0000000	0,0000000	1,0000000	
3	11	0,0000000	0,0000000	1,0000000	
	12	0,0000000	0,0300000	0,9700000	
	13	0,4000000	0,1800000	0,8200000	
	14	0,0000000	0,0000000	1,0000000	
	15	0,0000000	0,0000000	1,0000000	
4	16	0,0000000	0,0000000	1,0000000	
	17	0,0000000	0,0200000	0,9800000	
	18	0,2200000	0,1100000	0,8900000	
	19	0,0000000	0,0000000	1,0000000	
	20	0,0000000	0,0000000	1,0000000	
5	21	0,0000000	0,0000000	1,0000000	
	22	0,0000000	0,0000000	1,0000000	
	23	0,0000000	0,0000000	1,0000000	
	24	0,0000000	0,0000000	1,0000000	
	25	0,0000000	0,0000000	1,0000000	

Tabela 3: Dados que alimentam cada NAP.

Para a configuração da rede 1, o NAP1 tem suas entradas alimentadas pelos valores da linha 1 da tabela 3; o NAP2 tem suas entradas alimentadas pelos valores da linha 2; o NAP3 tem suas entradas alimentadas pela linha 3 da tabela; o NAP4 tem suas entradas alimentadas pela linha 4; e, finalmente, o NAP 5 tem suas entradas alimentadas pelos valores da linha 5 da tabela. Tem-se então a próxima interação da rede, a qual os NAPs 6 e 7 serão alimentados pelas saídas dos NAPs 1, 2, 3 e 4, respectivamente, gerando saídas para alimentar o NAP 8. A saída do NAP 8 representará uma das entradas do NAP9 final, em conjunto com a saída do NAP4.

A configuração da rede 2 é igual à da rede 1, apenas invertendo os valores de entrada dos NAPS 3 e 5.

Para a rede 3, o NAP1 tem suas entradas alimentadas pelos valores da linha 1 da tabela 3; o NAP2 tem suas entradas alimentadas pelos valores da linha 2; o NAP3 tem suas entradas alimentadas pela linha 3 da tabela; o NAP4 tem suas entradas alimentadas pela linha 4; e, finalmente, o NAP 5 tem suas entradas alimentadas pelos valores da linha 5 da tabela. Tem-se então a próxima iteração da

rede, a qual os NAPs 6 e 7 serão alimentados pelas saídas dos NAPs 1, 2, 3 e 4, respectivamente, o NAP 8 receberá em suas entradas os valores das saídas dos NAPs 5 e 7. A saída do NAP 8 representará uma das entradas do NAP9 final, em conjunto com a saída do NAP6.

A rede 4 possui a configuração semelhante à da rede 3, apenas invertendo os valores de entrada dos NAPs 3 e 5.

Independentemente da rede escolhida, tem-se o valor resultante da análise da primeira linha do número testado, compreendendo as 5 primeiras regiões com 10 *pixels*. Ao final da análise das 5 linhas de cada um dos 5 padrões de número iguais à referência, tem-se como saída da rede 1, por exemplo, os valores calculados (ver tabela 4):

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 1	0,0000000	0,0675463	0,0106066	0,0616948	0,0000000
Padrão 2	0,0000000	0,0812019	0,1236174	0,0577170	0,0000000
Padrão 3	0,0000000	0,0729298	0,1166190	0,0614919	0,0000000
Padrão 4	0,0000000	0,0691918	0,1096586	0,0579332	0,0000000
Padrão 5	0,0000000	0,0840387	0,1096586	0,0550000	0,0000000

Tabela 4: Exemplo de resultado da análise da rede de NAPs para cada linha dos 5 padrões.

Os valores apresentados na tabela 4 são extraídos do sistema, usando o código apresentado no quadro 9.

Quadro 9 – Código Java para treinamento da Rede de NAPS.

```
private void TreinarRedePadroes(){
    int j = -1;
    double mi1, mi2;
    for(int p=0;p<50;p++){
        if(p%5==0) j++;
        for(int i =0; i<25;i++){
            mi1 = (double)arraySubAreasReferencias[j][i]/100;
            mi2 = (double)arraySubAreasPadroes[p][i]/100;
            arrayResultadoNAPSPadroes[p][i] = NAP(mi1,mi2);
        }
    }
    RedePadroes();
    MinimosMaximos();
    JOptionPane.showMessageDialog(this,
                    "Rede ["+sequenciaRede+"] treinada com "+quantidadePadroes+" padrões! "
                    , "Informações: Treinamento de Rede", JOptionPane.WARNING_MESSAGE);
}
private void RedePadroes(){
    double l1,l2,l3,l4;
    int padrao=0;
    for(int p=0; p<10;p++){
        for(int l=0; l<5;l++){
            for(int c=0;c<5;c++){
                int fator = 5 * c;
                l1 = NAP(arrayResultadoNAPSPadroes[padrao][fator+0],arrayResultadoNAPSPadroes[padrao][fator+1]);
                switch (sequenciaRede) {
                    case 2:
                        l2 = NAP(arrayResultadoNAPSPadroes[padrao][fator+2],arrayResultadoNAPSPadroes[padrao][fator+3]);
                        l3 = NAP(l2,arrayResultadoNAPSPadroes[padrao][(fator+4)]);
                        l4 = NAP(l1,l3);
                        break;
                    case 1:
                        l2 = NAP(arrayResultadoNAPSPadroes[padrao][fator+4],arrayResultadoNAPSPadroes[padrao][fator+3]);
                        l3 = NAP(l1,l2);
                        l4 = NAP(l3,arrayResultadoNAPSPadroes[padrao][(fator+2)]);
                        break;
                    case 0:
                        l2 = NAP(arrayResultadoNAPSPadroes[padrao][fator+2],arrayResultadoNAPSPadroes[padrao][fator+3]);
                        l3 = NAP(l1,l2);
                        l4 = NAP(l3,arrayResultadoNAPSPadroes[padrao][(fator+4)]);
                        break;
                    default:
                        System.out.println("REDE 3");
                        l2 = NAP(arrayResultadoNAPSPadroes[padrao][fator+4],arrayResultadoNAPSPadroes[padrao][fator+3]);
                        l3 = NAP(l2,arrayResultadoNAPSPadroes[padrao][(fator+2)]);
                        l4 = NAP(l1,l3);
                        break;
                }
                arrayResutadoRedes[p][l][c]=l4;
                if (fator == 20) padrao++;
            }

        }
    }
}
```

Com a rede devidamente programada e os valores obtidos, como apresentado na tabela 4, é importante salientar que a tabela 4 apresenta os valores para 5 padrões, podendo o usuário escolher usar 3, 4 ou os 5 padrões para definir os limites aceitáveis para cada uma das 5 linhas (ver tabela 5):

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
MAIOR	0,0000000	0,0840387	0,1236174	0,0616948	0,0000000
MENOR	0,0000000	0,0675463	0,0106066	0,0550000	0,0000000

Tabela 5: Valores mínimos e máximos aceitáveis por linha na rede.

Os dados da tabela 5 representam os valores que compreendem o universo que o sistema aponta como aceitável para cada uma das 5 linhas, para sinalizar que o número testado é compatível com as evidências apresentadas pela rede e os valores obtidos utilizando o código apresentado no quadro 10.

Quadro 10 – Código Java para obtenção dos números mínimos e máximos de cada linha dos números.

```
private void MinimosMaximos(){
    double minimo,maximo;
    for(int p=0; p<10;p++){
        for(int l=0; l<5;l++){
            minimo=1;
            maximo=0;
            System.out.println("Quantidade padrões:"+quantidadePadroes);
            for(int c=0;c<quantidadePadroes;c++){
                if (arrayResutadoRedes[p][c][l]>maximo){
                    maximo = arrayResutadoRedes[p][c][l];
                }
                if (arrayResutadoRedes[p][c][l]<minimo){
                    minimo = arrayResutadoRedes[p][c][l];
                }

            }
            arrayLimitesRedes[p][l][0]=minimo;
            arrayLimitesRedes[p][l][1]=maximo;
        }
    }

}
```

3.4. GRÁFICOS PARA IDENTIFICAÇÃO E COMPARAÇÃO DE PADRÕES DE CARACTERES

Para facilitar a visualização dos testes, foram criados gráficos de evidência com uma escala que compreende de 0 a 5 linhas coincidentes, onde cada linha do número testado é verificada se está dentro dos valores mínimos e máximos extraídos das redes. Ao submeter o número aos valores da rede de todos os algarismos, o sistema sinalizará qual algarismo foi identificado, totalizando a quantidade de linhas para cada código numérico (ver gráfico 1). O primeiro gráfico sinaliza que o número 4 possui as 5 linhas coincidentes, dando certeza total da identificação desse código numérico.

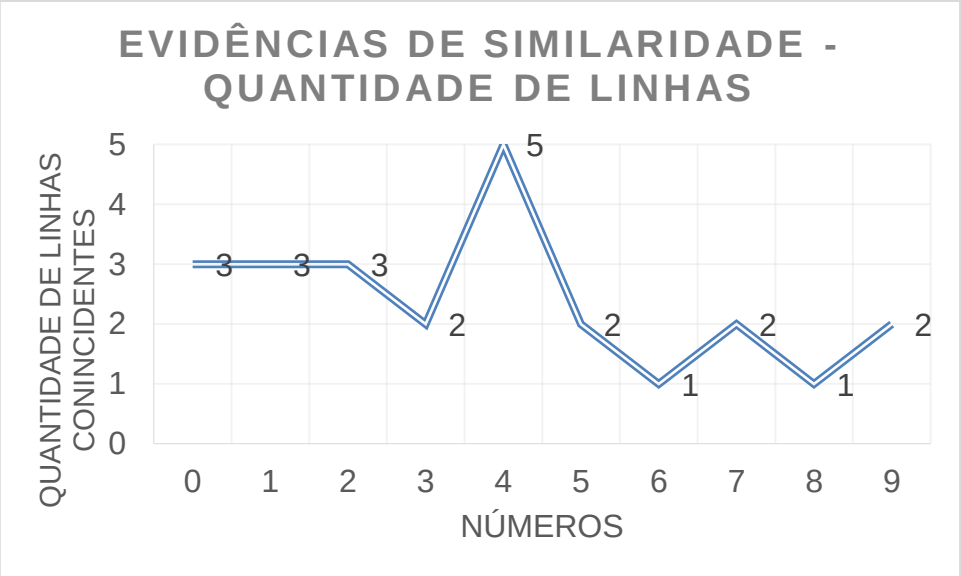


Gráfico 1: Resultado da submissão do número 4 a rede de Análise Paraconsistente.

Os valores padrões apresentados na Rede de Análise Paraconsistente 1 com 5 padrões foram obtidos através do *software* Arranjo Paraconsistente. Os ensaios para todos os números são apresentados no Apêndice A, bem como o resultado de aplicações dos algarismos que serviram de padrão para treinar a rede.

O resultado que a aplicação apresenta para cada análise de código numérico submetido ao *software* é feito através do código apresentado no Quadro 11.

Quadro 11 – Código Java para determinar número.

```
private void DeterminarNumero(){
    int maximo=0;
    String numerosEncontrados="";

    for(int i=0;i<10;i++){
        evidencias[i]=0;
    }
    for(int i=0;i<10;i++){
        if((arrayLinhasNumeroComp[0]>=arrayLimitesRedes[i][0][0])&&(arrayLinhasNumeroComp[0]<=arrayLimitesRedes[i][0][1])){
            evidencias[i]++;
        }
        if((arrayLinhasNumeroComp[1]>=arrayLimitesRedes[i][1][0])&&(arrayLinhasNumeroComp[1]<=arrayLimitesRedes[i][1][1])){
            evidencias[i]++;
        }
        if((arrayLinhasNumeroComp[2]>=arrayLimitesRedes[i][2][0])&&(arrayLinhasNumeroComp[2]<=arrayLimitesRedes[i][2][1])){
            evidencias[i]++;
        }
        if((arrayLinhasNumeroComp[3]>=arrayLimitesRedes[i][3][0])&&(arrayLinhasNumeroComp[3]<=arrayLimitesRedes[i][3][1])){
            evidencias[i]++;
        }
        if((arrayLinhasNumeroComp[4]>=arrayLimitesRedes[i][4][0])&&(arrayLinhasNumeroComp[4]<=arrayLimitesRedes[i][4][1])){
            evidencias[i]++;
        }
    }
    // Encontra a maior evidência.
    for(int i = 0;i<10;i++){
        if (evidencias[i]>maximo){
            maximo=evidencias[i];
        }
    }
    for(int i = 0;i<10;i++){
        if (evidencias[i]==maximo){
            numerosEncontrados += i+"  ";
        }
    }


    JOptionPane.showMessageDialog(this,
                        "Evidencias para o número["+0+"]="+evidencias[0]+
                        "\nEvidencias para o número["+1+"]="+evidencias[1]+
                        "\nEvidencias para o número["+2+"]="+evidencias[2]+
                        "\nEvidencias para o número["+3+"]="+evidencias[3]+
                        "\nEvidencias para o número["+4+"]="+evidencias[4]+
                        "\nEvidencias para o número["+5+"]="+evidencias[5]+
                        "\nEvidencias para o número["+6+"]="+evidencias[6]+
                        "\nEvidencias para o número["+7+"]="+evidencias[7]+
                        "\nEvidencias para o número["+8+"]="+evidencias[8]+
                        "\nEvidencias para o número["+9+"]="+evidencias[9]+
                        "\nMaior valor de evidência= "+maximo+
                        "\nNúmero(s) Determinado(s)= "+numerosEncontrados
                        , "Informações: Escala de 0 a 5", JOptionPane.WARNING_MESSAGE);

}
```

4. RESULTADOS E DISCUSSÕES

Neste capítulo, serão expostos os resultados obtidos, após as análises dos códigos numéricos submetidos às 4 diferentes configurações de redes de NAPs com quantidade de padrões testados variando de 3 a 5 para todas as redes.

Com o uso dos algoritmos da Lógica Paraconsistente Anotada (LPA), que permitem tratar os dados contraditórios, foi possível criar uma aplicação que viabiliza a análise dos dados, a aprendizagem e determinação de um padrão típico, comparando com amostras que visam determinar se são consideradas típicas ou não dentro da identificação dos códigos numéricos de contêineres.

O modelo desenvolvido auxilia a identificação de casos onde os aplicativos já em uso têm dificuldade na identificação dos códigos numéricos de contêineres.

Ocorre que muitas vezes o contêiner não tem sua identificação legível, resultando na perda do processo automático de identificação (ver figura 26). Por essa razão, a aplicação da Lógica Paraconsistente Anotada para obter a identificação pode gerar um ganho ainda não mensurável.

Figura 26: Imagem de contêiner não nítida para aplicação da LPA.

Submetendo números de contêineres extraídos, tratados e dimensionados em 50 por 50 *pixels* ao *software* de Arranjo Paraconsistente, o resultado abaixo foi alcançado para o número 0 usando a rede 1 com 5 padrões (ver figura 27).

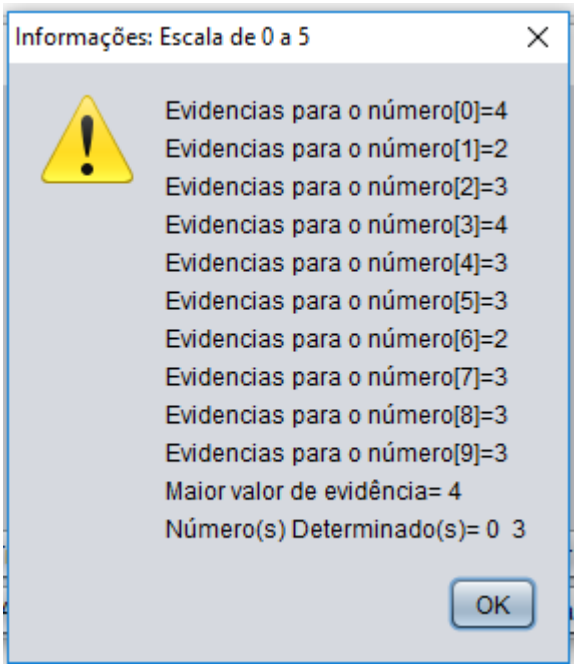


Figura 27: Resultado para análise do número 0 na rede 1 com 5 padrões.

Na figura 27, é possível perceber que, usando a rede 1 com 5 padrões, o aplicativo Arranjo Paraconsistente concluiu que os números que possuíam quatro linhas com valores dentro do intervalo aceitável poderiam ser os números 0 e 3. O *software* indicou o número correto, mas não de forma isolada, pois com esses valores o resultado também poderia ser o número 3. Na figura 28, é apresentado o resultado para a análise do número 1.

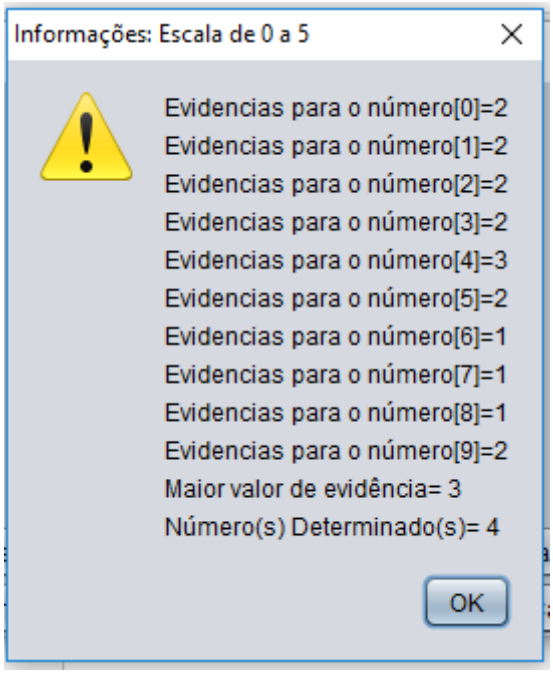


Figura 28: Resultado para análise do número 1 usando a rede 1 com 5 padrões.

Na figura 28, é possível perceber que o aplicativo Arranjo Paraconsistente concluiu, usando a rede 1 com 5 padrões, que o número apresentado era o número 4, de forma errada, pois o número submetido foi o número 1. Isso se deve muitas vezes ao tratamento da imagem e o grau de inclinação do número submetido ao aplicativo não estar adequado.

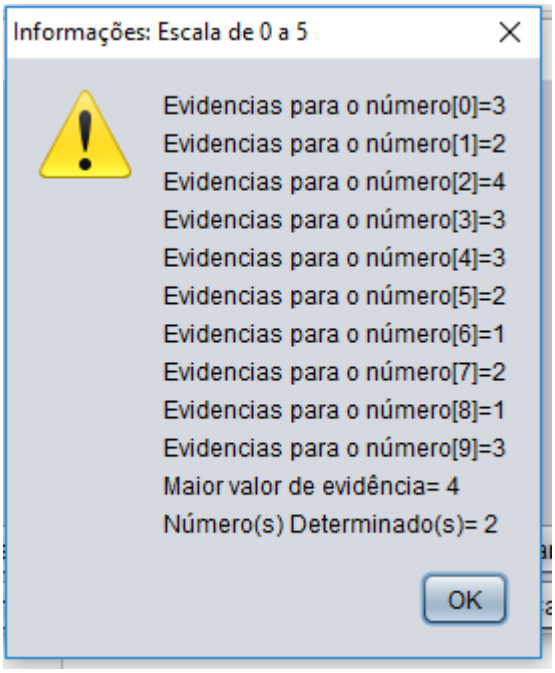


Figura 29: Resultado para análise do número 2.

Para o número 2, o aplicativo obteve resultado plenamente satisfatório, usando a rede 1 com 5 padrões, como apresentado na figura 29.

4.1. RESULTADOS OBTIDOS PARA REDE 1

Após exemplificar os resultados obtidos dos números 0, 1 e 2 da rede 1 com 5 padrões, a tabela 6 apresenta todos os resultados alcançados pelo *software* Arranjo Paraconsistente para a rede 1, empregando 3, 4 e 5 padrões.

REDE	QTD. PADRÕES	NÚMEROS TESTADOS E IDENTIFICADOS										Assertividade
		0	1	2	3	4	5	6	7	8	9	
Rede 1	5	0 e 3	4	2	1,2,3,4,5,7 e 9	4	5	6	0,2 e 5	0,2,8 e 9	2 e 9	80%
	4	0 e 3	4	2	1,2,4 e 9	4	5	6	0	0,8 e 9	2	60%
	3	0	4	2	1,2 e 4	4	1,2,3,4,5, e 9	6	0	0,8 e 9	0,1,2,3,4, e 9	70%

Identificado incorretamente

Tabela 6: Resultado para Rede 1 com 3, 4 e 5 padrões.

Analisando a tabela 6, é possível observar que com a rede treinada com 5 padrões, o nível de assertividade alcança 80% dos casos, identificando de maneira isolada os números 2, 4, 5, e 6. O número 0 foi também identificado como número 3; o número 3 também foi identificado como os números 1, 2, 4, 5, 6 e 9; o número 8 também foi identificado como os números 0, 2 e 9; o número 9 foi identificado também como número 2. Os números 1 e 7 não foram identificados com a rede nesta configuração.

A rede 1 utilizando 4 padrões alcança níveis de assertividade de 60%, não identificou os números 3 e 9, diferentemente do desempenho usando 5 padrões. Porém, o resultado positivo é igual ao obtido na configuração desta rede com 5 padrões, que aponta de maneira isolada os números 2, 4, 5 e 6.

O resultado utilizando 3 padrões melhorou contrapondo à análise efetuada com 4 padrões, a assertividade foi de 70% nesta configuração; o número 0 foi identificando de maneira isolada, diferentemente do desempenho empregando 4 e 5 padrões. Porém não é identificado de maneira isolada o número 5, que foi identificado isoladamente usando 4 e 5 padrões.

O nível de assertividade da rede 1 não se apresentou diretamente ligado a quantidade de padrões empregados, nesta rede treinada com 3, 4 e 5 padrões obteve-se assertividade de 70%, 60% e 80% respectivamente.

4.2. RESULTADOS OBTIDOS PARA REDE 2

A rede 2 possui arquitetura idêntica à rede 1, invertendo os valores de entrada dos NAPS 3 e 5; obteve-se desempenho próximo ao apresentado na rede 1. Porém, o número 2, que foi identificado com êxito de maneira isolada com qualquer quantidade de padrões na rede 1, deixa-se de obter a identificação nesta configuração (ver tabela 7).

REDE	QTD. PADRÕES	NÚMEROS TESTADOS E IDENTIFICADOS										Assertividade
		0	1	2	3	4	5	6	7	8	9	
Rede 2	5	0 e 2	4	0,3 e 4	0,3,9	4	0,2 e 5	6	5	0 e 1	0,2,3,7 e 9	60%
	4	0 e 2	4	0 e 3	0	4	0,2 e 5	6	5	1	0,2,3 e 9	50%
	3	0	4	0 e 3	0	4	0,2 e 5	6	5	1	0 e 3	40%

Identificado incorretamente

Tabela 7: Resultado para Rede 2 com 3, 4 e 5 padrões

A tabela 7 apresenta assertividade de 60% para a rede 2 treinada com 5 padrões, esta rede identificou o número 0 como 0 e também como número 2; o número 3 também como os números 0 e 9; número 5 também como os números 0 e 2; número 9 também como os números 0, 2, 3 e 7. Os números 1, 2, 7 e 8 não foram identificados.

Usando 4 padrões nesta rede, os resultados alcançados são semelhantes aos obtidos na configuração com 5 padrões, deixa-se de identificar o número 3. A assertividade da rede 2 com 4 padrões caiu para 50%.

A configuração da rede 2 utilizando 3 padrões, tem desempenho semelhante ao apresentado na rede 2 utilizando 4 padrões. Porém deixa de identificar o número 9, que obtinha sucesso nas configurações com 4 e 5 padrões, e passa a identificar de forma isolada o número 0, totalizando 40% de assertividade.

Nesta configuração de rede, foi possível evidenciar que a quantidade de padrões foi proporcional ao número de acertos, sendo que a rede configurada com 3, 4 e 5 padrões alcançou assertividade de 40%, 50% e 60% respectivamente.

4.3. RESULTADOS OBTIDOS PARA REDE 3

A rede 3 apresenta arquitetura diferente das redes 1 e 2. Nesta rede, os NAPs 3 e 4 geram uma das entradas do NAP 8 em conjunto com o valor de saída do NAP

5. Foi possível verificar que a rede 3 apresenta desempenho superior à rede 2, usando 3 e 4 padrões.

Comparada à rede 1, o mesmo nível de identificações corretas foram obtidos quando treinada com 4 padrões, conforme demonstrado na tabela 8.

REDE	QTD. PADRÕES	NÚMEROS TESTADOS E IDENTIFICADOS										Assertividade
		0	1	2	3	4	5	6	7	8	9	
Rede 3	5	0,8 e 9	2 e 4	2	2 e 5	4	5	2 e 6	0,3,5,7,8, e 9	9	2 e 5	60%
	4	0,8 e 9	2 e 4	2	5	4	5	6	0,3,5,6,7,8 e 9	2,8 e 9	2	70%
	3	0	2 e 4	2	5	4	0 e 5	6	0,3,5,7,8 e 9	8 e 9	2	70%

Identificado incorretamente

Tabela 8: Resultado para Rede 3 com 3, 4 e 5 padrões

No estudo da rede 3 com 5 padrões, o mesmo valor de 60% de assertividade da rede 2 empregando 5 padrões foi atingido; os número 1, 3, 8 e 9 não foram identificados nesta configuração. O número 0 também foi identificado como os número 8 e 9; o número 6 foi identificado também como número 2; e o número 7 foi identificado também como os número 0, 3, 5, 8 e 9.

Para a rede 3 empregando 4 padrões, o *software* obteve desempenho superior à mesma rede usando 5 padrões; o número 6 foi identificado de maneira isolada, diferentemente da mesma rede configurada com 5 padrões. O número 8 foi reconhecido, porém, foi também identificado como os números 2 e 9.

O desempenho de 70% da rede 3 com 3 padrões foi identifico ao obtido na rede 1 utilizando 3 padrões. Porém nesta configuração o número 9 não foi identificado e o número 7 foi também identificado como 0, 3, 5, 8 e 9.

Esta configuração de rede não apresentou desempenho positivo proporcional ao número de padrões empregados, com a quantidade de 3, 4 e 5 padrões, obteve-se a assertividade de 70%, 70% e 60% sucessivamente.

4.4. RESULTADOS OBTIDOS PARA REDE 4

A rede 4 possui arquitetura semelhante à apresentada na rede 3, os valores de entrada dos NAPS 3 e 5 são invertidos. É possível verificar que a rede 4 não apresentou desempenho melhor que 40%, independentemente da quantidade de padrões empregados. Nesta configuração de rede, nenhum número foi identificado de maneira isolada (ver tabela 9).

O ganho da rede 4, comparado às redes 1, 2 e 3, foi observado na identificação do número 1, que não foi identificado em nenhuma das outras configurações apresentadas. Porém o número 1 também foi identificado como os número 0, 2, 3, 4 e 9; o número 5 também foi identificado como os números 0, 1, 2, 3, 4 e 9; o número 7 também foi identificado como os números 0,1, 2, 3, 4, 5 e 9.

REDE	QTD. PADRÕES	NÚMEROS TESTADOS E IDENTIFICADOS										Assertividade
		0	1	2	3	4	5	6	7	8	9	
Rede 4	5	5	0,1,2,3,4,5 e 9	0,2 e 3	0	5	0,1,2,3,4,5 e 9	0,1,2,3,4,5 e 9	0,1,2,3,4,5,7 e 9	5	0	40%
	4	5	0,1,2,3,4,5 e 9	3	0	5	0,1,2,3,4,5 e 9	0,1,2,3,4,5 e 9	0,1,2,3,4,5,7 e 9	0,1,2,3,4,5 e 9	0	30%
	3	5	0,1,2,3,4,5 e 9	3	0	5	0,1,2,3,4,5 e 9	0,1,2,3,4,5 e 9	0,1,2,3,4,5 e 9	0,1,2,3,4,5 e 9	0	20%

Identificado incorretamente

Tabela 9: Resultado para Rede 4 com 3, 4 e 5 padrões

Utilizando esta rede configurada com 4 padrões, apenas os números 1, 5 e 7 são identificados. Esta rede deixou de identificar o número 2 e com nenhum dos números a identificação se apresentou de maneira isolada.

É possível observar que na rede 4 com 3 padrões o desempenho apresentado foi semelhante ao registrado com 4 padrões. Porém o número 7 deixa de ser identificado.

Nesta configuração de rede foi possível obter uma relação direta entre o nível de assertividade e a quantidade de padrões empregada, para 3, 4 e 5 padrões obteve-se a assertividade de 20%, 30% e 40% respectivamente.

4.5. DISCUSSÕES

A partir das análises efetuadas nos resultados obtidos pelo *software* Arranjo Paraconsistente para as redes de NAPs 1, 2, 3 e 4 demonstradas nas tabelas 6, 7, 8 e 9 respectivamente, aplicando 3, 4 e 5 padrões, foi possível obter êxito na identificação de todos os números, porém, não de forma isolada. O resultado deve ser considerado empregando o melhor desempenho em cada uma das configurações para cada um dos códigos numéricos.

Os melhores resultados obtidos para cada um dos números, empregando uma das 4 redes foram:

- Número 0: Apresenta correta identificação nas redes 1, 2 ou 3, com 100% de acerto usando os 3 padrões nas redes 1 e 3.
- Número 1: Corretamente identificado na rede 4, empregando 3, 4 ou 5 padrões. Porém, o *software* não obteve a identificação isolada em

nenhuma configuração de padrões, sendo também identificado como os números 0, 2, 3, 4, 5 e 9.

- Número 2: Apresenta correta identificação na rede 1, usando 3, 4 ou 5 padrões; na rede 3, usando 3, 4 ou 5 padrões; e na rede 4, usando 5 padrões. Obteve-se assertividade de 100% nas redes 1 e 3 com qualquer quantidade de padrões.
- Número 3: Identificado corretamente nas redes 1, com 5 padrões, e na rede 2, com 5 padrões. Porém, o *software* não obteve a identificação isolada em nenhuma configuração de padrões, o número 3 também foi identificado como 0 e 9 na rede 2 com 5 padrões, onde obteve o seu melhor desempenho.
- Número 4: Obteve-se assertividade de 100% nas redes 1, 2 ou 3, com qualquer quantidade de padrões. Na rede 4, não foi identificado corretamente com nenhuma quantidade de padrões.
- Número 5: Identificado corretamente em todas as redes com qualquer quantidade de padrões. Alcançou nível de assertividade de 100% nas redes 1 e 3, com 4 ou 5 padrões.
- Número 6: Identificado nas redes 1, 2 ou 3, com qualquer quantidade de padrões. Não obteve nenhuma identificação na rede 4. O número é identificado com 100% de assertividade nas redes 1, 2 ou 3, com qualquer quantidade de padrões, com exceção da rede 3, com 5 padrões, que também identificou o número 2.
- Número 7: Corretamente identificado utilizando a rede 3 com qualquer quantidade de padrões, também a rede 4, com 4 ou 5 padrões. Porém, o *software* não obteve a identificação isolada em nenhuma configuração de padrões, apontando também os números 0, 3, 5, 6, 8 e 9.
- Número 8: Corretamente identificado na rede 1, com qualquer quantidade de padrões; e na rede 3, com 3 ou 4 padrões. Porém, o software não obteve a identificação isolada em nenhuma configuração de padrões, indicando também como possíveis números o 0, 2 e 9.
- Número 9: Identificado nas redes 1, com 3 e 5 padrões; e na rede 2, com 4 ou 5 padrões. Porém, o software não obteve a identificação

isolada em nenhuma configuração de padrões, também apresentando os números 0 ,1, 2, 3 e 7.

5. CONCLUSÕES

Apresentamos nesta dissertação uma abordagem empregando lógica paraconsistente anotada com anotação de dois valores (LPA2v) em *softwares* para reconhecimento de código numérico extraído de imagens digitais de contêineres. Como ficou demonstrado nos resultados dessa pesquisa, que se junta a outros trabalhos nesse segmento, o uso da LPA em aplicações similares tem se mostrado bastante bem-sucedido. Nesse estudo foi possível comprovar a qualidade do emprego da rede de NAPs ao obter uma assertividade de até 100% quando usada a rede com melhor desempenho relacionado ao número que é apresentado como a ser reconhecido na entrada. Com o *hardware* que se dispõe atualmente para a captura das imagens, tudo leva a crer que analises baseadas em configurações de NAPs é possível se conseguir melhorias nos resultados com baixo custo. Fica aqui demonstrado que a escolha da LPA é um caminho a ser explorado, pois o fator determinante não se encontra necessariamente na lógica empregada no processo, mas sim, na melhoria dos resultados de identificação dos códigos numéricos extraído das imagens digitais dos contêineres.

Considerando que a indicação correta do algarismo no tratamento da imagem é um processo crítico, ficou evidente que o ajuste e a centralização da imagem, bem como a inclinação do número a ser testado, farão toda diferença nos resultados alcançados. E neste trabalho através das configurações de redes paraconsistentes utilizadas, foi possível evidenciar que se os números testados já foram submetidos à rede para formação dos padrões, obtiveram os níveis de assertividade de mais de 90%, como mostrado no Anexo A.

As redes paraconsistentes empregadas nesta dissertação foram compostas por 3, 4 ou 5 padrões para testes, de livre escolha para o usuário, onde apenas a melhor configuração para cada um dos números trouxe o sucesso na identificação isolada, não sendo possível ainda neste trabalho confirmar que a quantidade de padrões empregada poderia trazer ganho de assertividade no processo de identificação. Para trabalhos futuros o emprego de um banco de dados para armazenar os valores extraídos dos números também é algo cabível, visto que neste trabalho o principal objetivo foi alcançado, qual seja, o desenvolvimento do algoritmo que utiliza uma das 4 configurações de rede de NAPS, com possibilidade de uso de 3, 4 ou 5 padrões. Os resultados obtidos com estas configurações servirão como

contribuições para futuras pesquisas com o objetivo de estabelecer sistemas especialistas paraconsistentes com desempenhos otimizados para a correta identificação do código numérico extraído das imagens digitais de contêineres.

REFERÊNCIAS BIBLIOGRÁFICAS

ABE, J. M. **Aspectos de Computação Inteligente Paraconsistente.** São Paulo: Instituto de Estudos Avançados da Universidade de São Paulo, 2013. Disponível em: http://www.iea.usp.br/pesquisa/grupos/logica-e-teoria-da-ciencia/publicacoes/E-book%20Aspectos%20de%20Computacao%20Inteligente%20Paraconsistente.pdf/view. Acessado em: 24 Fev. 2016.

CODESP. **Resumo das movimentações de cargas no Porto de Santos: comparativo mensal e acumulado**, 2016.
Disponível em: http://www.portodesantos.com.br/imprensa.php?pagina=resano. Acesso em: 05 set. 2016.

DA SILVA FILHO, J. I.; ABE, J. M. **Fundamentos das Redes Neurais Artificiais: destacando aplicações em Neurocomputação**. São Paulo: Villipress, 2001.

DA SILVA FILHO, J. I.; ABE, J. M.; LAMBERT-TORRES, G. **Inteligência artificial com as redes de análises Paraconsistentes**: **teoria e aplicações.** Rio de Janeiro: LTC, 2008.

DA SILVA FILHO, J. I. **Métodos de Aplicações da Lógica Paraconsistente Anotada de Anotação com Dois Valores LPA2v com Construção de Algoritmo e Implementação de Circuitos Eletrônicos** − 1999. Tese (Doutorado) – Universidade de São Paulo – São Paulo, 1999. 1999. Disponível em: http://www.paralogike.com.br/Tese.pdf. Acesso em: 24 fev. 2016.

DA SILVA FILHO, J. I. **Métodos de aplicações da Lógica Paraconsistente Anotada de anotação com dois valores – LPA2v.** Revista Seleção Documental – GLPA, n. 1, Santos (SP), 2006, pp. 18-25. Disponível em: http://www.paralogike.com.br/Logica%20Paraconsistente%20Anotada%20com%20anotacao%20de%20dois%20valores%20LPA2v.pdf. Acesso em: 1 fev. 2016.

DA SILVA FILHO, J. I. **Introdução às Células Neurais Artificiais Paraconsistentes**, *Revista Seleção Documental*, n. 8, Santos (SP), 2007, pp. 5-13. Disponível em: http://www.paralogike.com.br/Introducao%20as%20Celulas%20Neurais%20Artificiais%20Paraconsistentes.pdf. Acesso em: 1 fev. 2016.

DA SILVA FILHO, J. I.; ABE, J. M. **Introdução à lógica paraconsistente anotada com ilustrações.** Santos (SP), Emmy, 2000.

DEDGAONKAR, S. G.; CHANDAVALE, A. A.; SAPKAL, A. M. **Survey of methods for character recognition**. IJEIT, v. 1, n. 5, 2012. Disponível em: http://www.ijeit.com/vol%201/Issue%205/IJEIT1412201205_36.pdf. Acesso em: 12 jan. 2016.

DEITEL, P.; DEITEL, H. **Java: como programar**. São Paulo: Pearson Prentice Hall, 2010.

DE MILANO, D.; HONORATO, L. B. **Visão Computacional**. 2010. Faculdade de Tecnologia da Universidade Estadual de Campinas, Limeira (SP). Disponível em: <http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010_IA_FT_UNICAMP_visaoComputacional.pdf>. Acesso em: 11 jan. 2016.

FERRARA, L. F. P. **Um estudo da Rede Neural Artificial Paraconsistente - RNAP em um Processo de Reconhecimento de Caracteres**, 2004. 148p. Dissertação (Mestrado). Uberlândia (MG) - Universidade Federal de Uberlândia.

FONSECA, A. P.; MONTEIRO, M. J.; COUTINHO, P. C., OLIVEIRA, A. R. **Os desafios de um porto organizado para atender a demanda dos terminais arrendados**. XXVIII ANPET – Congresso de Pesquisa e Ensino em Transportes. Curitiba (PR), 2013. Disponível em: www.anpet.org.br/xxviiianpet/anais/documents/AC444.pdf. Acesso em: 11 jan. 2016.

GONÇALVES, E. **Desenvolvendo Aplicações WEB com NetBeans IDE 6**. São Paulo: Ciência Moderna, 2008.

HAPAG LLOYD CONTAINER LINE. **Container Specification.** Hamburg, 2002. Disponível em: https://www.hapag-lloyd.com/downloads/press_and_media/publications/Brochure_Container_Specification_en.pdf. Acesso em: 27 out. 2016.

LEMES NETO, M. C.; VENSON, N. **Lógica Paraconsistente**, Departamento de Informática e Estatística da Universidade Federal de Santa Catarina, 2002. Disponível em: http://educaonline.eng.br/UNISANTA/HTML/DOWNLOAD/LIVRO/LPA/LP_Nerio_Mauricio.pdf. Acesso em: 11 fev. 2016.

MARIO, M.C. **Modelo de análise de variáveis craniométricas através das redes neurais artificiais paraconsistentes.** 2006. São Paulo: Faculdade de Medicina, Universidade de São Paulo. Disponível em: http://www.teses.usp.br/teses/disponiveis/5/5160/tde-06112006-130001/publico/MauricioConceicaoMario.pdf.Acesso em: 12 Jan. 2016.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de imagens digitais: princípios, algoritmos e aplicações**. São Paulo: Thomson Learning, 2008.

RASBAND, W.S. **ImageJ, U. S. National Institutes of Health: User Guide** Bethesda, Maryland, USA, 2012. Disponível em: http://imagej.nih.gov/ij/docs/guide. Acesso em: 12 out. 2016.

SCAZUFCA, M. **A Importância do contêiner para o desenvolvimento do comércio mundial: impactos no Porto de Santos.** 2008. II Colóquio Internacional sobre Comércio e Cidade: Uma relação de origem. Faculdade de Arquitetura e Urbanismo da Universidade de São Paulo, 2008. Disponível em: http://www.labcom.fau.usp.br/wp-content/uploads/2015/05/2_cincci/5002%20Scazufca.pdf. Acesso em: 14 set. 2016.

SILVA, A. A. **Casamento de Mapas Utilizando Redes Neurais Artificiais Paraconsistentes**, 2011. 89 p. Tese (Doutorado) – Departamento de Engenharia Eletrônica e Computação, Instituto Tecnológico da Aeronáutica. São José dos Campos (SP). Disponível em: http://www.bdita.bibl.ita.br/tesesdigitais/lista_resumo.php?num_tese=61005. Acesso em: 24 fev. 2016.

SOUZA, S. **Sistema de reconhecimento de caracteres numéricos manuscritos baseado nas redes neurais artificiais paraconsistentes**. 2013. Dissertação (Mestrado). São Paulo: Faculdade de Medicina, Universidade de São Paulo. Disponível em: http://www.teses.usp.br/teses/disponiveis/5/5160/tde-16012014-144713/publico/SheilaSouzaVersaocorrigida.pdf. Acesso em: 12 jan. 2016.

STEENKEN, D.; VOß, S.; STAHLBOCK, R. **Container terminal operation and operations research-a classification and literature review.** OR spectrum**,** v. 26, n. 1, p. 3-49, 2004.

SUBRAHMANIAN, V. S. **On the semantics of quantitative Logic Programs**, Proc. 4th. IEEE Symposium on Logic Programming, Computer Society, Washington D.C., 1987.

UNISANTA. **Manual de elaboração de teses e dissertações** – Programa de Pós-Graduação em Engenharia Mecânica da Unisanta. Disponível em: http://sites.unisanta.br/ppgmec/manual_teses_dissertacoes_ppgmec.pdf. Acesso em: 15 fev. 2016.

APÊNDICE A – Ensaios de similaridade com uso de caracteres existentes na rede 1 com 5 padrões

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
0	MAIOR	0,0000000	0,0453459	0,1039230	0,0715454	0,0000000
	MENOR	0,0000000	0,0393700	0,0668954	0,0409268	0,0000000

Tabela 10: Valores mínimos e máximos aceitáveis por linha do número 0 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 1	0,0000000	0,0422049	0,0668954	0,0409268	0,0000000

Tabela 11: Valores do teste do número 0.

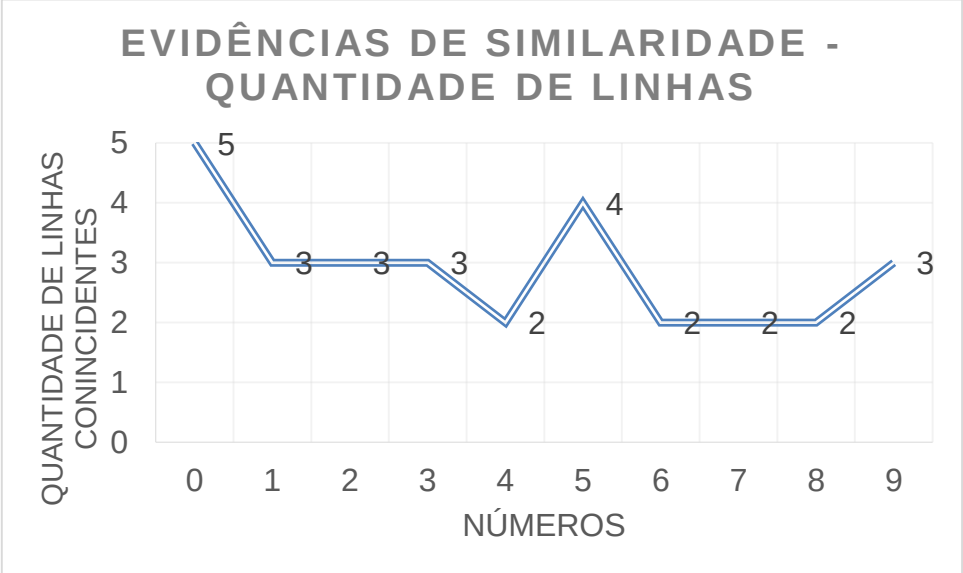


Gráfico 2: Resultado da submissão do número 0 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
1	MAIOR	0,0000000	0,0840387	0,1236174	0,0616948	0,0000000
	MENOR	0,0000000	0,0675463	0,1096586	0,0550000	0,0000000

Tabela 12: Valores mínimos e máximos aceitáveis por linha do número 1 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 1	0,0000000	0,0729298	0,1166190	0,0614919	0,0000000

Tabela 13: Valores do teste do número 1.

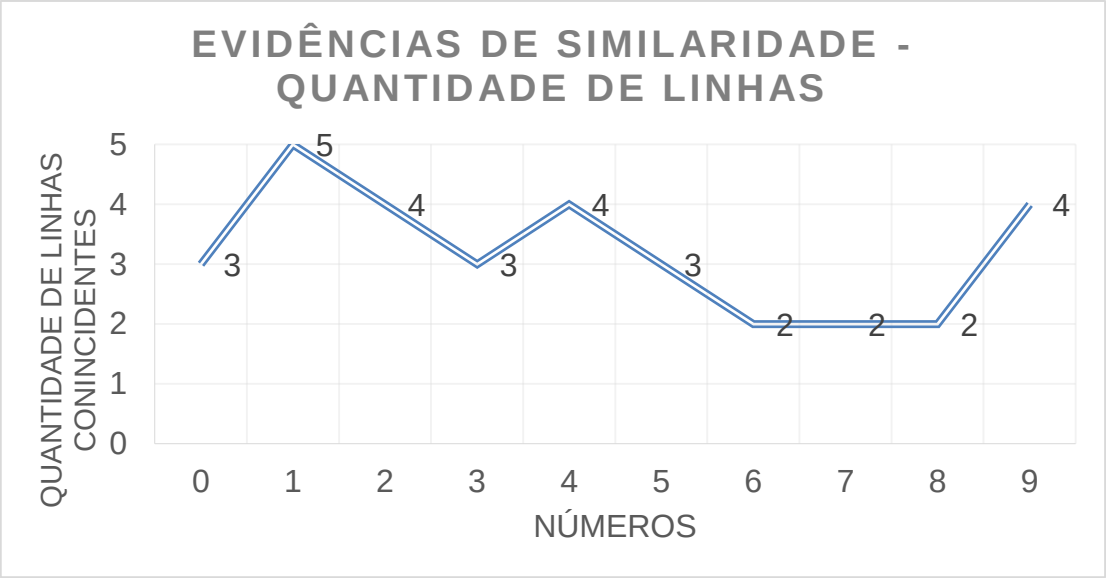


Gráfico 3: Resultado da submissão do número 1 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
2	MAIOR	0,0035355	0,0868548	0,0804674	0,0694172	0,0000000
	MENOR	0,0000000	0,0701338	0,0586302	0,0539096	0,0000000

Tabela 14: Valores mínimos e máximos aceitáveis por linha do número 2 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 1	0,0035355	0,0761167	0,0804674	0,0539096	0,0000000

Tabela 15: Valores do teste do número 2.

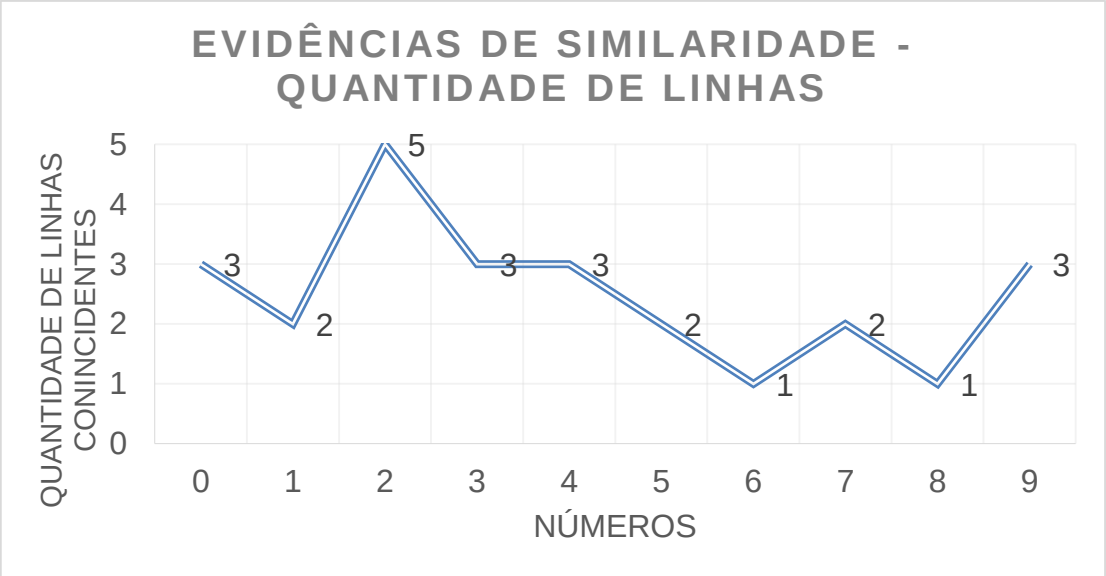


Gráfico 4: Resultado da submissão do número 2 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
3	MAIOR	0,0000000	0,0453459	0,1039230	0,0715454	0,0000000
	MENOR	0,0000000	0,0549432	0,0721976	0,0523808	0,0000000

Tabela 16: Valores mínimos e máximos aceitáveis por linha do número 3 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 3	0,0000000	0,0655267	0,0841873	0,0591080	0,0000000

Tabela 17: Valores do teste do número 3.

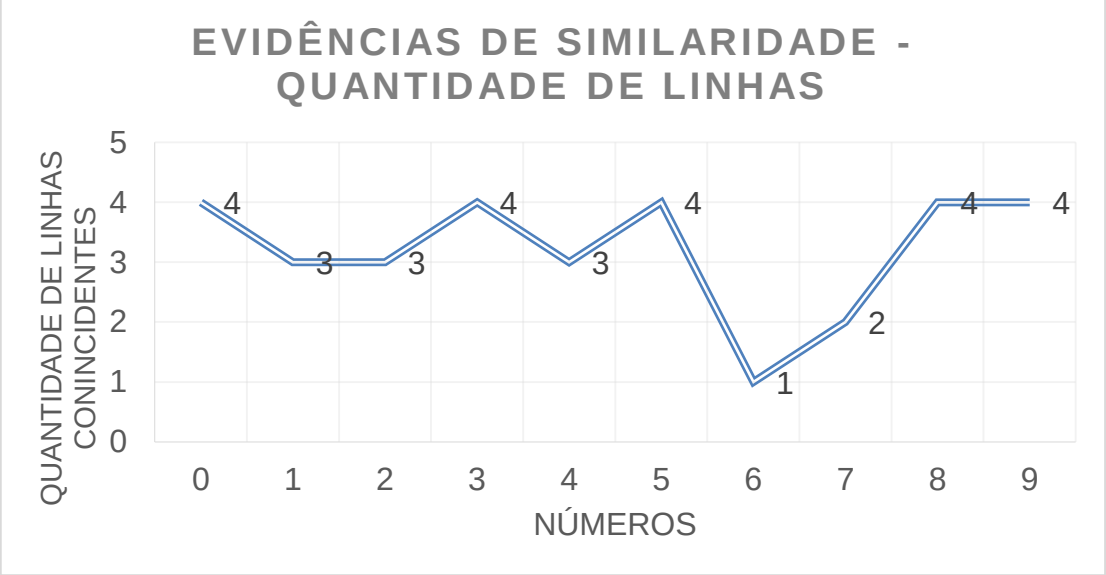


Gráfico 5: Resultado da submissão do número 3 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
4	MAIOR	0,0000000	0,1075581	0,1324764	0,0541410	0,0000000
	MENOR	0,0000000	0,0649038	0,1035314	0,0439460	0,0000000

Tabela 18: Valores mínimos e máximos aceitáveis por linha do número 4 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 4	0,0000000	0,0832917	0,1324764	0,0456892	0,0000000

Tabela 19: Valores do teste do número 4.

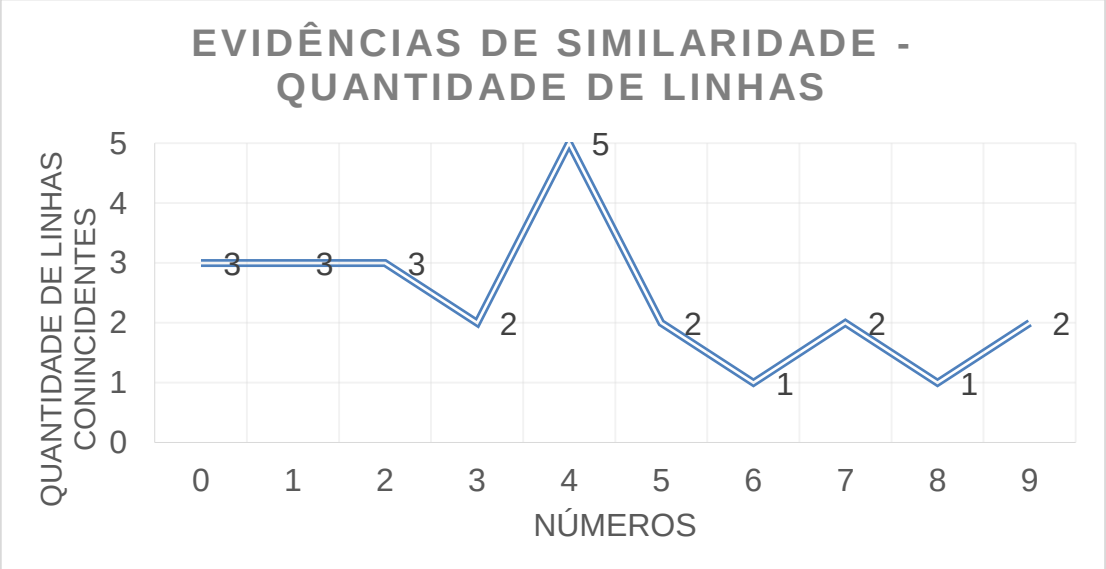


Gráfico 6: Resultado da submissão do número 4 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
5	MAIOR	0,0000000	0,0735697	0,0851469	0,0288314	0,0000000
	MENOR	0,0000000	0,0349106	0,0450000	0,0158114	0,0000000

Tabela 20: Valores mínimos e máximos aceitáveis por linha do número 5 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 5	0,0000000	0,0349106	0,0593717	0,0278388	0,0000000

Tabela 21: Valores do teste do número 5.

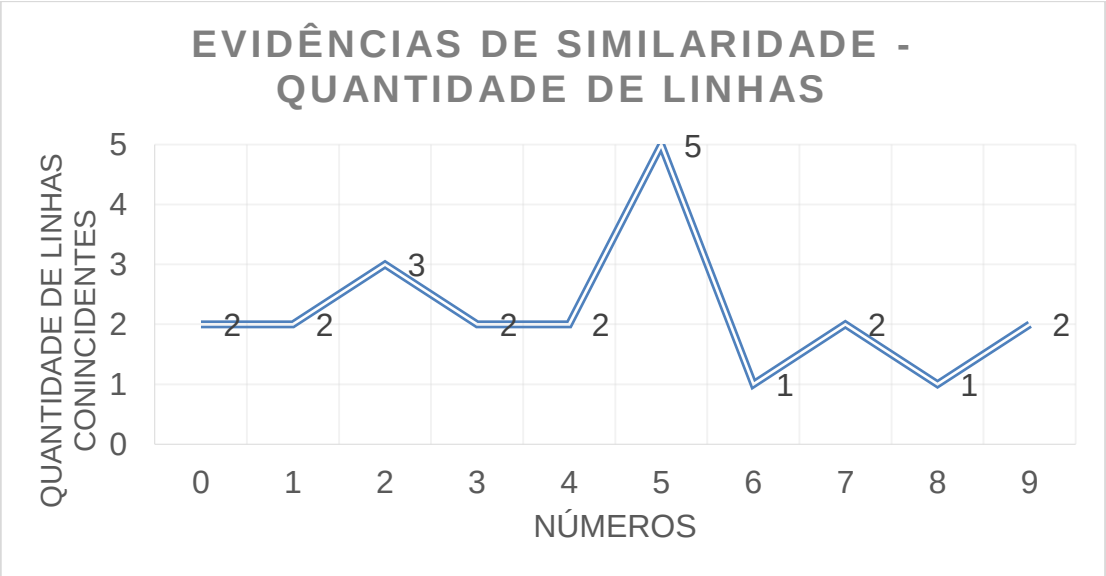


Gráfico 7: Resultado da submissão do número 5 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
6	MAIOR	0,0025000	0,0570636	0,1118872	0,0680992	0,0000000
	MENOR	0,0025000	0,0412311	0,0903811	0,0606733	0,0000000

Tabela 22: Valores mínimos e máximos aceitáveis por linha do número 6 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 6	0,0025000	0,0570636	0,1097440	0,0665207	0,0000000

Tabela 23: Valores do teste do número 6.

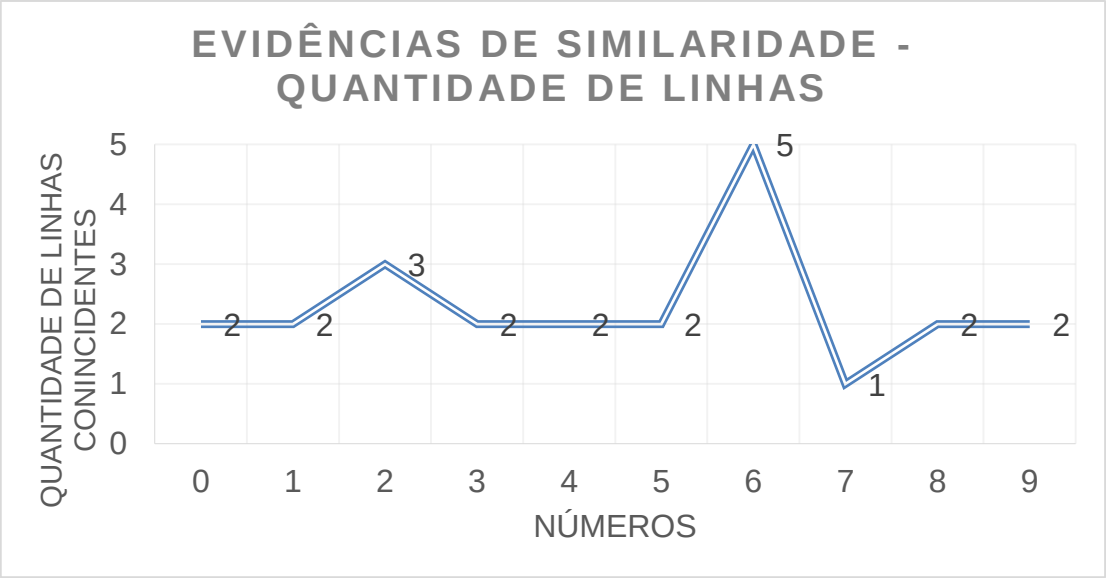


Gráfico 8: Resultado da submissão do número 6 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
7	MAIOR	0,0000000	0,0869267	0,0698212	0,0576086	0,0050000
	MENOR	0,0000000	0,0427931	0,0458939	0,0476314	0,0050000

Tabela 24: Valores mínimos e máximos aceitáveis por linha do número 7 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 7	0,0000000	0,0510514	0,0698212	0,0576086	0,0050000

Tabela 25: Valores do teste do número 7.

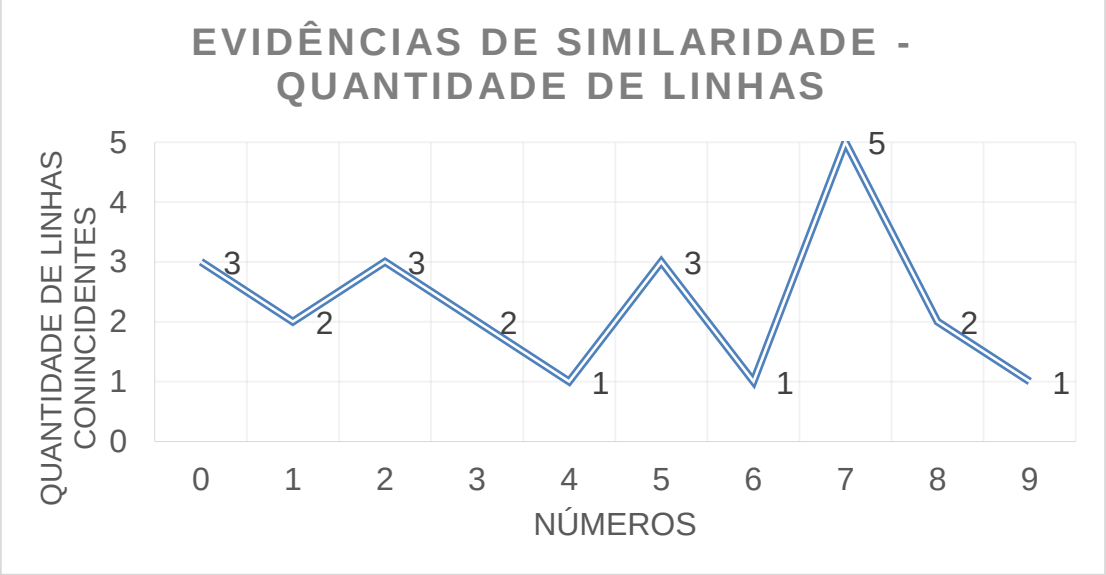


Gráfico 9: Resultado da submissão do número 7 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
8	MAIOR	0,0000000	0,0671286	0,1081087	0,0710634	0,0100000
	MENOR	0,0000000	0,0555090	0,0833292	0,0487981	0,0100000

Tabela 26: Valores mínimos e máximos aceitáveis por linha do número 8 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 8	0,0000000	0,0613392	0,1013040	0,0710634	0,0100000

Tabela 27: Valores do teste do número 8.

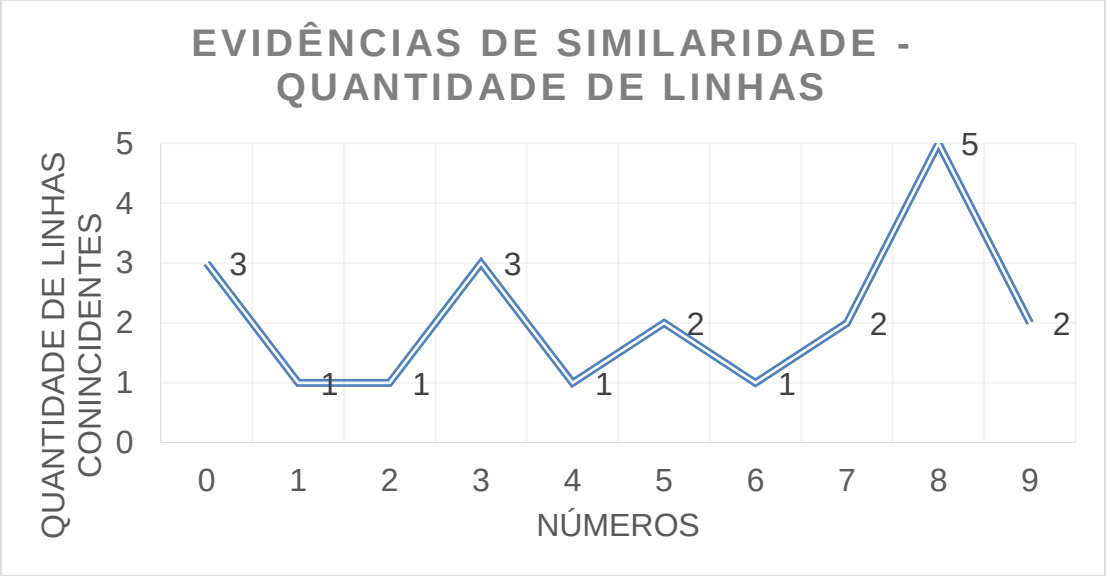


Gráfico 10: Resultado da submissão do número 8 à rede de Análise Paraconsistente.

		Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
9	MAIOR	0,0000000	0,0787004	0,0918899	0,0654790	0,0000000
	MENOR	0,0000000	0,0546008	0,0742883	0,0602080	0,0000000

Tabela 28: Valores mínimos e máximos aceitáveis por linha do número 9 na rede 1 com 5 padrões.

	Linha 1	Linha 2	Linha 3	Linha 4	Linha 5
Padrão 9	0,0000000	0,0607762	0,0742883	0,0654790	0,0000000

Tabela 29: Valores do teste do número 9.

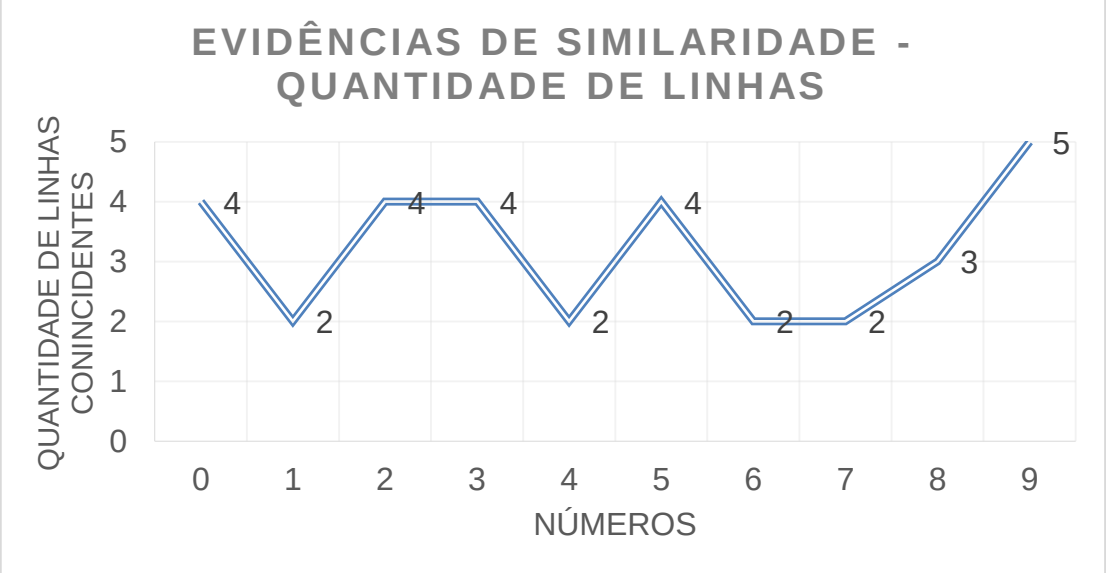


Gráfico 11: Resultado da submissão do número 9 à rede de Análise Paraconsistente.