UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Marília Gonçalves Dutra da Silva

SIMULAÇÃO A EVENTOS DISCRETOS COM A UTILIZAÇÃO DE INTELIGÊNCIA COMPUTACIONAL EM MÓDULO DE DECISÃO NO *SOFTWARE* URURAU

CAMPOS DOS GOYTACAZES, RJ.
Julho de 2014

UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Marília Gonçalves Dutra da Silva

SIMULAÇÃO A EVENTOS DISCRETOS COM A UTILIZAÇÃO DE INTELIGÊNCIA COMPUTACIONAL EM MÓDULO DE DECISÃO NO *SOFTWARE* URURAU

Dissertação apresentada ao Programa de Mestrado em Pesquisa Operacional e Inteligência Computacional, da Universidade Candido Mendes – Campos / RJ, para obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Orientador: Prof. João José de Assis Rangel - D.Sc.
Co-orientador: Prof. Ítalo de Oliveira Matias - D.Sc.

CAMPOS DOS GOYTACAZES, RJ.
Julho de 2014

MARÍLIA GONÇALVES DUTRA DA SILVA

SIMULAÇÃO A EVENTOS DISCRETOS COM A UTILIZAÇÃO DE INTELIGÊNCIA COMPUTACIONAL EM MÓDULO DE DECISÃO NO SOFTWARE URURAU

Dissertação apresentada ao Programa de Mestrado em Pesquisa Operacional e Inteligência Computacional, da Universidade Candido Mendes – Campos / RJ, para obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Aprovada em ___ de_____ de 2014.

BANCA EXAMINADORA

_____

Prof. João José de Assis Rangel, D.Sc.
Universidade Cândido Mendes

_____

Prof. Italo de Oliveira Matias, D.Sc.
Universidade Cândido Mendes

_____

Prof. Rogério Atem de Carvalho, D.Sc.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense

_____

Prof. Túlio Almeida Peixoto, M.Sc.
Universidade Cândido Mendes

CAMPOS DOS GOYTACAZES, RJ.

Julho de 2014

AGRADECIMENTOS

Agradeço a Deus pelo o amor incondicional.
Ao meu orientador, professor João José de Assis Rangel, pela dedicação e competência.
Ao meu esposo David, incansável em me ajudar e apoiar.
A minha amada filha Sarah pela alegria e carinho.
Aos meus pais por me incentivarem.
Ao bolsista Jhonathan e aos professores Ítalo e Túlio por toda disponibilidade em colaborar.
Ao Instituto Federal de Educação, Ciência e Tecnologia Fluminense pela bolsa de estudos.

Confie no Senhor de todo o seu coração e não se apoie em seu próprio entendimento.

Provérbios 3:5 NVI

RESUMO

SIMULAÇÃO A EVENTOS DISCRETOS COM A UTILIZAÇÃO DE INTELIGÊNCIA COMPUTACIONAL EM MÓDULO DE DECISÃO NO *SOFTWARE* URURAU

O objetivo deste trabalho foi investigar a representação do comportamento de um guarda de trânsito em modelos de simulação a eventos discretos aplicados a problemas reais de tráfego urbano. Para representar os aspectos do comportamento humano do guarda, propôs-se a utilização de redes neurais artificiais e a utilização de dados históricos de decisões anteriores para um treinamento supervisionado da rede neural artificial. Para construir o modelo de simulação com representação de decisões humanas utilizou-se o ambiente de simulação a eventos discretos Ururau. Criou-se também para o Ururau um recurso decisório, baseado no *framework* Encog, capaz de decidir o fluxo de uma entidade no modelo de acordo com dados dinâmicos obtidos do modelo de simulação e submetidos a uma rede neural artificial treinada. Além de construir um modelo de simulação com a representação das decisões de um guarda de trânsito atuando em um cruzamento real da cidade de Campos dos Goytacazes, foram também simuladas situações hipotéticas de aumento da taxa de chegada de veículos e pedestres. Os resultados obtidos possibilitaram investigar a viabilidade do uso de redes neurais artificiais para representar as decisões de guardas de trânsito na simulação de problemas de tráfego urbano. Espera-se também que possam ser feitas analogias entre os problemas de trânsito e problemas de manufatura, de forma a viabilizar outras pesquisas e trabalhos futuros.

PALAVRAS-CHAVE: Simulação a Eventos Discretos, Ururau, Encog, Decisões Humanas, Redes Neurais Artificiais

ABSTRACT

DISCRETE EVENT SIMULATION USING COMPUTATIONAL INTELLIGENCE APPLIED TO THE INTERNAL DECISION MODULE IN URURAU SOFTWARE.

The objective of this study was to investigate the representation of the behavior of a traffic cop in models of discrete event simulation applied to real problems of urban traffic. To represent aspects of human behavior the traffic cop, we proposed the use of artificial neural networks and the use of historical data from past decisions for supervised training of artificial neural network. To build the simulation model to represent human decisions used the simulation environment discrete event Ururau . It also created to Ururau a decision-making resource, based on Encog framework, able to decide the flow of an entity in the model according to data obtained from the dynamic simulation and subjected to a trained neural network model. In addition to building a simulation model to represent the decisions of a traffic warden acting in a real city intersection of Campos dos Goytacazes , were also simulated hypothetical situations of increased arrival rate of vehicles and pedestrians . The results enabled us to investigate the feasibility of using artificial neural networks to represent the decisions of traffic wardens in the simulation of urban traffic problems. It is also expected that analogies can be made between the traffic problems and manufacturing problems, in order to facilitate further research and future work.

KEYWORDS: Discrete Event Simulation, Ururau, Encog , Human Decisions, Artificial Neural Networks .

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AG – Algorítmo Genético
API – *Application Programming Interface*
CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
DSOL – *Distributed Simulation Object Library*
DES – *Discrete Event Simulation*
ENIAC – Electronic Numerical Integrator Analyzer and Computer
ERP – *Enterprise Resource Planning*
FIFO – *First In First Out*
FORTRAN - Formula Translating System
FOSS – *Free and open-source software*
GNU – Acrônimo recursivo de GNU is Not Unix
GPL – *General Public License*
GPSS – *General Purpose Simulation System*
GPU – Graphics Processing Unit
GUI – *Graphic User Interface*
JAVANNS – Java Neural Network Simulator
JSL – *Java Simulation Library*
LGPL – *GNU Lesser General Public License*
LGPL3 – GNU LESSER GENERAL PUBLIC LICENSE Version 3
LPPG – Linguagens de Programação de Propósito Geral
MLP – *Multilayer Perceptron*
RNA – Rede Neural Artificial
SED – Simulação a Eventos Discretos
SIMAN – Simulation Analysis
TCP – *Transmission Control Protocol*
TI – Tecnologia da Informação
VBA – *Visual Basic for Applications*
XOR – Ou Exclusivo

SUMÁRIO

1. INTRODUÇÃO

Em recente trabalho, Silva e Rangel (2012) utilizaram redes neurais artificiais para representar o comportamento de ações de pessoas em modelos de simulação a eventos discretos, nas situações onde normalmente não é possível determinar os aspectos lógicos da tomada de decisão. Neste caso, foi criado um módulo com um algoritmo de uma Rede Neural Artificial (RNA) que se comunicava com o modelo de simulação, capaz de representar de forma mais realística as decisões realizadas por pessoas nos sistemas modelados.

Os softwares de simulação de forma geral, representam as decisões dos sistemas sob análise nos modelos, com regras básicas, utilizando operadores lógicos, como: “<”, “>”, “==”, “AND”, “OR”, “XOR” e suas derivações. Além disso, em outros casos, utilizam-se apenas probabilidades obtidas em dados históricos para representar essas decisões.

De certa forma, alguns aspectos dos sistemas que configuram decisões mais inteligentes, que muitas vezes estão associadas a ações humanas, são representadas de forma simplificada nestes modelos. Assim, a possibilidade representar estas ações de forma mais realística nos modelos de simulação é um desafio que se apresenta atualmente, e tem motivado pesquisadores como Robinson *et. al* (2001). Apesar de poucas citações encontradas, quase 71% dos softwares de simulação permitem a customização dos modelos através de algoritmos provenientes de outras linguagens, como pode ser encontrado em Swain (2007).

No trabalho de Silva *et al.* (2012), foi criado um módulo que se comunicava através de *sockets* TCP (*Transmission Control Protocol)* com o modelo de simulação no *software* Ururau. Esse módulo foi criado para executar a RNA onde uma decisão, proveniente da ação de uma pessoa, pôde ser representada mais realisticamente durante a simulação. No entanto, a comunicação com o algoritmo através de sockets TCP pode acarretar, muitas vezes, em atraso na execução do modelo, podendo tornar inconveniente este tipo de recurso em modelos de maior porte.

A proposta deste trabalho é criar um modelo de simulação a eventos discretos utilizando inteligência computacional para tomada de decisões não triviais, por meio de um módulo de decisão inteligente acoplado ao ambiente de simulação Ururau.

Para tal, pretende-se simular um cruzamento do tráfego urbano, tendo como exemplo o trabalho desenvolvido por Baptista e Rangel (2013) no qual os autores lançaram mão da semelhança entre os fenômenos dinâmicos e estocásticos associados ao fluxo de veículos nos cruzamentos e puderam avaliar o sistema de controle em estudo.

A análise de tal sistema pôde então representar de forma análoga o comportamento existente em diversas situações típicas encontradas em linhas de montagens industriais. A principal diferença é que as vias urbanas são públicas e os autores puderam coletar os dados de forma livre.

1.1. CONTEXTUALIZAÇÃO E MOTIVAÇÃO

O desenvolvimento de trabalhos e pesquisas envolvendo a Simulação a Eventos Discretos tem sido amplo e abrangente nas mais diversas áreas que vai desde a engenharia passando pela matemática, medicina, biologia, ciências naturais e inúmeras outras.

Basta realizar uma consulta mais aprofundada nas bases de dados científicas para notar que há um aumento gradativo de pesquisas na área, mostrando sua importância e relevância científica.

Outro estudo com ampla gama de aplicações está relacionado com a Inteligência Computacional. Se aplicado aos modelos de simulação, nota-se que várias técnicas como Algoritmos Genéticos (AG), Redes Neurais Artificiais (RNA) e Lógica *Fuzzy*, são empregadas às variáveis de saída do modelo com o objetivo de otimizá-las ao longo da execução do mesmo, por meio de mecanismos de retroalimentação.

Contudo o assunto ainda é pouco explorado em relação ao uso de simuladores capazes de aplicar algum tipo de Inteligência Computacional durante a execução do modelo, antes mesmo de se conhecer as variáveis de saída.

1.2.OBJETIVO GERAL

O objetivo do trabalho é aplicar um modelo de simulação a eventos discretos construído com o *software* Ururau em um sistema real com utilização de Inteligência Computacional em módulo de decisão.

1.3.OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho estão relacionados com os itens a seguir:

- Criação de um modelo simulando um cruzamento do trânsito com o objetivo de experimentar e analisar as diferentes tomadas de decisão proveniente de uma RNA;
- Estudar diferentes *frameworks* buscando a melhor solução a ser empregada com o Ururau;
- Acoplar recursos de RNA ao Ururau utilizando um *framework* para criação de redes neurais artificiais aplicáveis aos mais variados modelos de simulação a eventos discretos;
- Criação de uma rede neural a ser treinada utilizando informações coletadas de um cruzamento do tráfego urbano;
- Estabelecer relações análogas entre problemas de tráfego urbano e das linhas de produção para serem testados e avaliados no modelo de simulação.

1.4.JUSTIFICATIVA E CONSIDERAÇÕES

Não foram encontradas ocorrências na literatura de *softwares* simuladores que possuem as características descritas anteriormente. Os trabalhos de Robinson *et.al.* (1998), Silva *et.al.* (2012) e Bergmann *et al.* (2014) são capazes de simular situações onde uma determinada decisão é tomada por meio de comunicação externa com o *software* simulador.

Dessa forma, apresenta-se como motivação para a realização deste trabalho os seguintes pontos:

i) Pouca ou nenhuma existência de ambientes de simulação relatados na literatura, com capacidade de aplicar inteligência computacional a partes do sistema que necessitam refletir comportamentos de ações humanas;

ii) Potencialidade de se ter recursos que possam representar de forma realística as decisões, que no dia a dia dos processos industriais dependem de pessoas responsáveis por tomadas de decisões e cujos simuladores possibilitam somente a representação dessas decisões sob forma de percentuais e condições;

iii) Continuidade do trabalho desenvolvido por Silva *et.al.* (2012) nos seguintes aspectos:

a. Desenvolvimento de recursos nativos no Ururau para representar de modo realístico decisões humanas em modelos de simulação a eventos discretos;

b. Possibilidade de criar um modelo baseado em um sistema real a partir de dados coletados em um cruzamento de tráfego urbano.

No trabalho de Silva *et.al.* (2012) o módulo inteligente se comunica diretamente com o núcleo do Ururau e o JSL (*Java Simulation Library*), como apresentado na Figura 1.1.

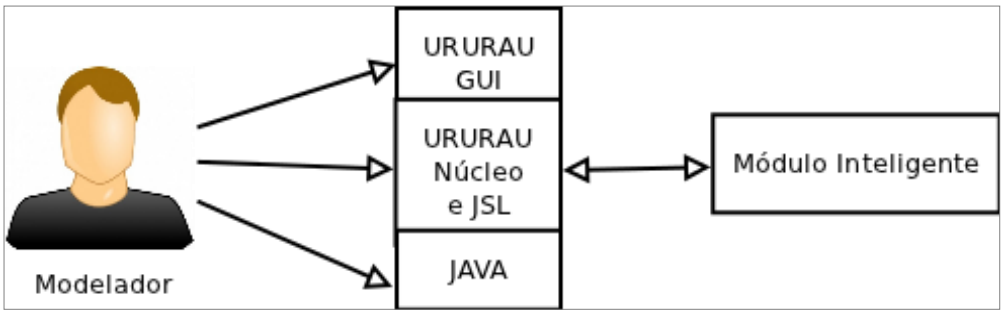

Figura 1.1: Interligação entre o Ururau e o módulo inteligente.
Fonte: SILVA *et.al.* (2012).

Além do aspecto acima, Silva *et.al.* (2012) desenvolveram um tipo de comunicação entre o Ururau e o módulo inteligente baseada em *sockets* TCP conforme a figura 1.2.

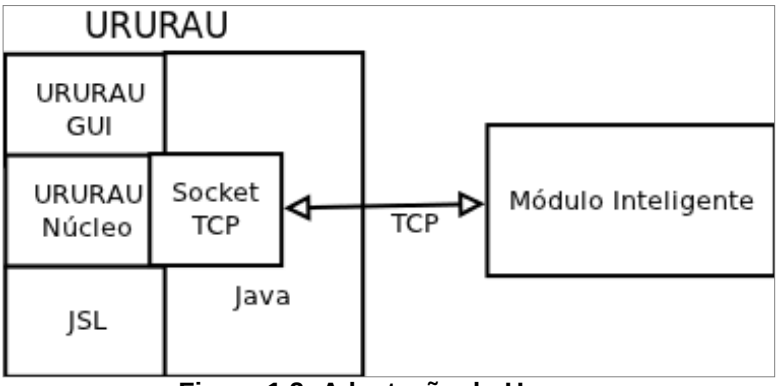

Figura 1.2: Adaptação do Ururau.
Fonte: SILVA *et.al.* (2012).

1.5.DELIMITAÇÕES DO TRABALHO

O escopo deste trabalho delimita-se na Simulação a Eventos Discretos fazendo uso da Inteligência Computacional, mais precisamente das Redes Neurais Artificiais. A perspectiva é que seja construído um componente de decisão no Ururau que possibilite modelar decisões inteligentes quando a utilização de operadores lógicos e de porcentagens, comumente utilizados nos ambiente de simulação conhecidos atualmente, não consigam representar de forma realística as tomadas de decisão.

1.6.ESTRUTURA DO TRABALHO

O trabalho está estruturado em 5 capítulos, da seguinte forma:

O Capítulo 1 traz a introdução, contextualização e motivações para realização do trabalho, além da definição dos objetivos.

O Capítulo 2 apresenta a revisão bibliográfica que faz um levantamento de várias bibliotecas linguagens e ambientes de simulação, além de um levantamento bibliométrico.

O Capítulo 3 trata dos materiais e métodos utilizados para o desenvolvimento do trabalho, onde são demonstradas analogias entre sistemas de tráfego urbano e sistemas industriais, decisão com Inteligência Computacional em Modelos de Simulação, bem como a Metodologia da Simulação e o Modelo de Simulação com Decisão em Inteligência Computacional.

No Capítulo 4 são apresentados os resultados e as discussões a respeito do estudo, testes e experimentos.

Por fim, no Capítulo 5, são relatadas as considerações finais e descritas sugestões para trabalhos futuros.

2. REVISÃO DE LITERATURA

2.1.REFERENCIAL TEÓRICO

2.1.1.Simulação

De acordo com Pantuza Júnior *et al*.(2011), entre as ferramentas que auxiliam na tomada de decisões, destaca-se a Pesquisa Operacional, que consiste na aplicação de métodos que auxiliam no processo de tomada de decisões, citando como exemplo desses métodos a simulação computacional. Da mesma forma, Sargent (2013) afirma que os modelos de simulação são cada vez mais usados para resolver problemas e para ajudar na tomada de decisões.

Compreende-se a simulação como uma imitação do comportamento de um sistema real, que inclui a criação e observação de um sistema artificial, de modo que seja possível tirar conclusões sobre o sistema representado (BANKS *et al.,* 2010).

A simulação envolve a criação de modelos que imitam os comportamentos de interesse. A experimentação do modelo permite gerar observações dos comportamentos para se tentar entender, resumir ou generalizar esses comportamentos (WHITE JR & INGALLS 2009).

Chwif e Medina (2010) afirmam que um modelo de simulação procura capturar o comportamento de um dado sistema e representá-lo como um modelo computacional, sendo necessária a capacidade do modelo representar os diversos fenômenos aleatórios existentes no sistema, construídos a partir da observação e levantamento de dados do fenômeno.

Para Guidorizzi *et al.* (2009), a simulação facilita a tomada de decisão uma vez que possibilita variações de parâmetros e estudos de cenários diversos.

De acordo com Chwif e Medina (2010) a simulação computacional pode ser classificada em três categorias básicas: Simulação de Monte Carlo, Simulação Contínua e Simulação a Eventos Discretos.

2.1.2.Simulação de Monte Carlo

Law (2007) define Simulação de Monte Carlo como um esquema de números aleatórios que variam entre 0 e 1, que é utilizado para resolução de problemas estocásticos ou determinísticos.

A simulação de Monte Carlo não considera o tempo explicitamente como variável e pode ser usada desde a integração numérica de funções matemáticas como a simulação de riscos em um empreendimento. Este tipo de simulação pode ser realizada por meio de planilhas eletrônicas, por bibliotecas de *software* com pelo menos algum gerador de número aleatório. Um exemplo de simulador Monte Carlo é o suplemento *@Risk* da *Palisade*® para a planilha eletrônica *Microsoft Excel*® (PEIXOTO *et al*. 2012).

2.1.3.Simulação Contínua

De acordo com Law (2007), a Simulação Contínua refere-se à modelagem de um sistema através de uma representação em que o estado das variáveis muda continuamente ao longo do tempo.

Peixoto e*t al.* (2012) afirma que o sistema contínuo é descrito sob forma de equações diferenciais, sendo este tipo de simulação utilizado para simular circuitos elétricos ou um sistema de reservatórios conectados por tubos e controlados por válvulas, como também simular problemas de difusão de calor e massa, dentre outros.

2.1.4.Simulação a Eventos Discretos

A Simulação a Eventos Discretos, refere-se a modelagem de um sistema em que as variáveis mudam de estado ao longo do tempo a partir da ocorrência de um evento (LAW, 2007).

Banks *et al.*(2010) afirma que a alteração do estado das variáveis ocorre apenas em um conjunto discreto de pontos no tempo, e que a análise dos modelos de simulação são realizados por meio de métodos numéricos ao invés de métodos analíticos.

De acordo com Peixoto *et al*. (2012) uma simulação a eventos discretos pode ser convertida de maneira aproximada em uma simulação contínua quando o número de entidades no sistema for relativamente grande para um curto intervalo de tempo.

2.1.5.Aplicação da Simulação

São vastas as áreas de aplicação da simulação, a seguir são relacionadas algumas delas, descritas por Chwif e Medina (2010), Banks *et al.*(2010) e Law (2007):

- Aeroportos e portos: despacho de bagagens, dimensionamento de postos de *check in*, para verificar se a quantidade de equipamentos e pessoas é suficiente para atuar na carga e descarga de navios e contêineres;
- Bancos: política de abertura e fechamento de caixas, número de caixas automáticos, tempo de espera nas filas, problemas de *layout;*
- Reengenharia de Processos de Negócios: remodelar e reestruturar organizações e seus processos, sem que seja necessário parar o seu funcionamento antes de uma análise mais aprofundada por meio da simulação;
- Serviços (*call centers, fast food,* hospitais, entre outros): analisar tempos de espera, utilização de recursos , garantia dos níveis de serviço, etc;
- Projeto e análise de Sistemas de Manufatura: movimentação e armazenagem de materiais, análise de estoques, linhas de montagens;
- Área militar;
- Construção de estradas e rodovias;

Segundo Bergmann *et al*. (2014) na área de produção e logística, a simulação é uma ferramenta bem aceita para o planejamento, avaliação e monitoramento de processos relevantes. Há também um interesse crescente no uso de simulação como ferramenta de apoio à decisão operacional no controle de fabricação.

2.1.6.Breve História da Simulação

De acordo com Goldsman *et al.* (2010), a história da simulação pode ser escrita por diversas perspectivas:

i. Uso da simulação – análise, treinamento, pesquisa;
ii. Tipos de modelos de simulação – eventos discretos, contínua, combinação discreta contínua;
iii. Linguagens ou ambientes de simulação – GPSS (*General Purpose Simulation System*), SIMSCRIPT, SIMULA, SLAM, Arena, AutoMod, Simio;
iv. Domínio de aplicação ou comunidades de interesse – comunicação, manufatura, transporte, etc.

Os autores, contudo, fazem um relato que consideram informal, objetivando destacar pessoas, lugares e acontecimentos, bem como estimular a documentação das contribuições históricas sobre a Simulação a Eventos Discretos e Simulação de Monte Carlo.

Goldsman *et al.* (2010), dividem a história da simulação considerando principalmente:

a) era pré-computador até a Segunda Guerra Mundial (1777-1945) com o surgimento do método de Monte Carlo;
b) período formativo (1945-1970) com o surgimento do primeiro computador eletrônico de propósito geral (ENIAC) e o trabalho desenvolvido por Stanislaw Ulam, John von Neumann, Nicholas Metropolis e outros que utilizaram o método Monte Carlo em computadores eletrônicos para resolver problemas na difusão de nêutrons que surgiram no projeto da bomba de hidrogênio e que eram analiticamente intratáveis.
c) período da expansão (1970-1982) se distingue por melhorias, ampliações e acréscimos na área de Simulação a Eventos Discretos, envolvendo ensino, pesquisa e prática.

Chwif e Medina (2010), apresentam uma breve abordagem sobre a história da simulação. Segundo os autores, no início da década de 50, a simulação era realizada por meio da programação em FORTRAN. A partir de 1961 surgiu o GPSS,

que é uma linguagem de simulação ou uma linguagem de programação direcionada à simulação. Já na década de 80, surgiram os primeiros simuladores, tornando a construção dos modelos mais gráfica e menos textual. Atualmente os simuladores dominam o mercado, embora uma linguagem de simulação seja mais flexível do que um simulador.

2.1.7.Linguagens, Ambientes e Bibliotecas utilizadas em Simulação a Eventos Discretos

De acordo com Law (2007), existem quatro categorias de *softwares* utilizados para desenvolver modelos de simulação:

i. Linguagens de Programação de Propósito Geral;
ii. Bibliotecas de Simulação;
iii. Linguagens Específicas para Simulação;
iv. Ambientes de Simulação.

2.1.7.1.Linguagens de Programação de Propósito Geral (LPPG)

As LPPGs não são especificamente designadas para uso na simulação, fazendo com que atualmente poucas pessoas as utilizem para este fim, contudo, para aplicações em algumas áreas, pacotes baseados na linguagem Java sejam utilizados (LAW, 2007).

Além do Java, outras linguagens podem ser utilizadas para implementação de um modelo de simulação, como por exemplo: Basic, C e Fortran (BANKS *et al.* 2010).

2.1.7.2.Bibliotecas de Simulação

De acordo com Peixoto *et al*. (2012), uma das formas de implementar modelos de simulação, é utilizar bibliotecas para fazer um simulador específico ou adicionar a simulação como ferramenta de análise em um sistema de informação existente. Pode-se citar como exemplo, duas bibliotecas em Java: a biblioteca DSOL (*Distributed Simulation Object Library*) e a biblioteca JSL (*Java Simulation Library*).

2.1.7.2.1.Biblioteca DSOL

A biblioteca DSOL permite uma modelagem usando vários formalismos como: ser guiado a eventos, a processo e a atividade. Também permite simulação de sistemas de eventos discretos combinada com simulação de sistemas de tempo contínuo. Esta biblioteca possibilita que informações distribuídas em vários sistemas sejam utilizadas para alimentar o modelo de simulação (PEIXOTO *et al*. 2012).

2.1.7.2.2.Biblioteca JSL

A biblioteca de simulação JSL que trabalha com simulação de eventos discretos e os modelos podem ser desenvolvidos como em um sistema de filas ou em uma abordagem mais detalhada por modelagem orientada a objetos (PEIXOTO *et al*. 2012).

De acordo com (Peixoto *et al.* 2013), o JSL permite a construção de modelos orientados a processo e, quando necessário, adição de novos comandos de processo quando o modelo de simulação fica mais complexo. Este também usa a linguagem Java, que é multiplataforma e tem licença GPL (*General Public License*) para desenvolvimento em código livre.

2.1.7.3.Linguagens Específicas para Simulação

Chwif e Medina (2010), afirmam que em 1961 surgiu o GPSS que é uma linguagem de simulação ou uma linguagem de programação direcionada à simulação.

Os modelos de simulação podem ser construídos usando as linguagens específicas de simulação, como SIMAN ou GPSS (LAW,2007; BANKS et al. 2010).

De acordo com Sargent (2013), a construção de modelos de simulação usando linguagens de simulação resulta em menos erros e na redução do tempo de construção do modelo, contudo, as LPPG oferecem maior flexibilidade ao modelador.

2.1.7.4.Ambientes de Simulação

Os simuladores ou os ambientes de simulação, surgiram na década de 80, com interface própria para diminuir a quantidade de linhas de programação. Com a evolução das interfaces gráficas dos sistemas operacionais dos computadores, os *softwares* de simulação ficaram bem mais fáceis de operar, uma vez que a construção dos modelos tornou-se mais gráfica e menos textual. Atualmente os simuladores são os mais utilizados no mercado, embora que uma linguagem de programação seja mais flexível (Chwif e Medina 2010).

Abaixo seguem alguns dos principais simuladores destacados por Law, (2007) e Banks *et al.* (2010).

2.1.7.4.1.Arena®

Arena® é um pacote de simulação de propósito geral desenvolvido pela *Rockwell Automation* que é muito utilizado nas áreas de manufatura, cadeias de suprimentos, defesa, saúde e *call centers*. O Arena® possui módulos que são organizados em *templates.* O *templates* "*Basic Process"* possui módulos que são utilizados em todos os modelos virtuais para representar intervalos, departamentos, serviços e lógicas de decisão das entidades. Já o *templates* "*Advanced Process"* contém módulos que são utilizados para executar lógica de processo mais avançada e para acessar dados em arquivos externos. O *templete.* "*Advanced Transfer"* possui módulos para modelar diversos tipos de esteiras, empilhadeiras, veículos entre outros. Por fim, o *template* "Flow Process" é utilizado para modelar tanques, tubos, válvulas e operações de processamento em lotes (LAW, 2007).

De acordo com Banks *et al.*(2010) o Arena® nas versões básica e profissional, pode ser utilizado em simulação de sistemas discretos e de sistemas contínuos. Utiliza a linguagem de simulação SIMAN e possui uma arquitetura aberta que inclui VBA *(Visual Basic for Applications)* embutido. Permite a transferência de dados com outras aplicações, bem como o desenvolvimento de interface personalizada. A figura 2.1 representa o ambiente de desenvolvimento do Arena®.

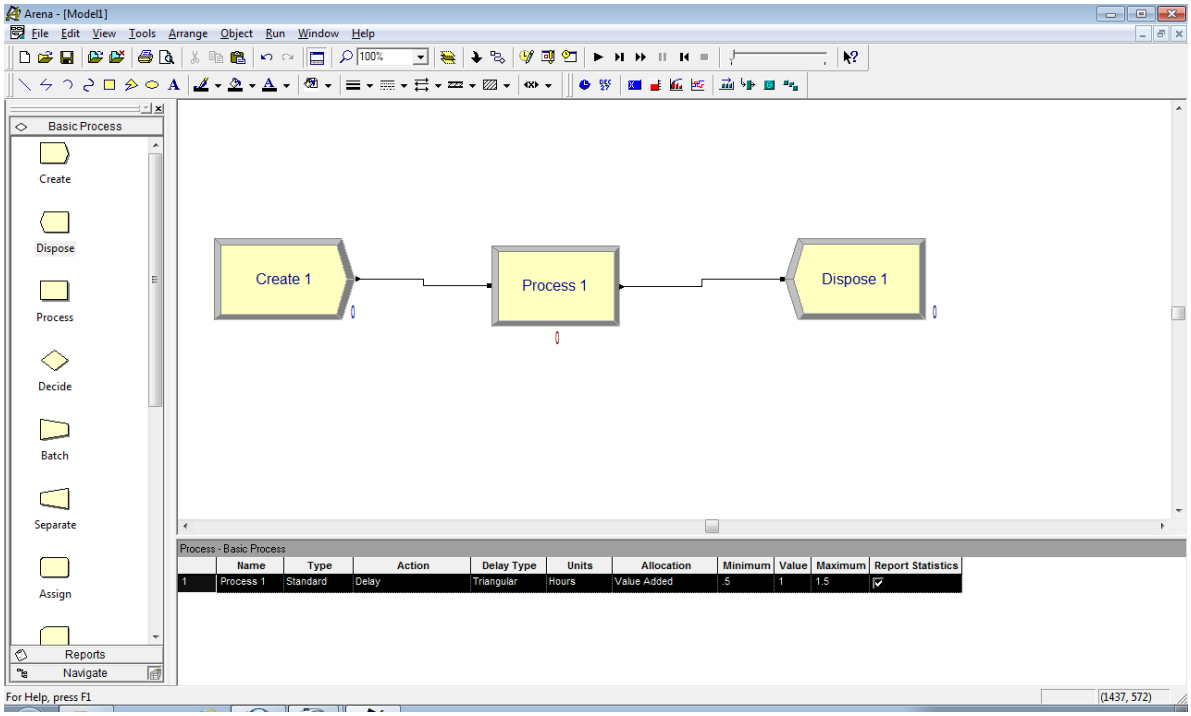

Figura 2.1: Ambiente de desenvolvimento do Arena®.

2.1.7.4.2.ProModel

O ProModel é uma pacote de simulação para manufatura desenvolvido pela PROMODEL *Corporation.* Praticamente todo modelo desenvolvido a partir do ProModel utiliza os seguintes construtores: (LAW, 2007).

a) Locais – utilizado para modelar máquinas, filas, esteiras ou tanques;
b) Entidades – utilizadas para representar peças, matéria prima, ou informação;
c) Chegadas – utilizado para especificar quais peças entram no sistema;
d) Processos – utilizados para definir a rota de peças através do sistema e especificar as operações que são realizadas para cada parte em cada local;
e) Recursos – utilizados para modelar recursos estáticos ou dinâmicos como trabalhadores ou empilhadeiras.

O ProModel oferece suporte para animação 2D e 3D. O *layout* da animação é criado automaticamente quando o modelo é desenvolvido, utilizando gráficos de bibliotecas pré-definidas ou importadas pelo usuário. (BANKS *et al.* 2010).

A figura 2.2 apresenta o ambiente de desenvolvimento do ProModel.

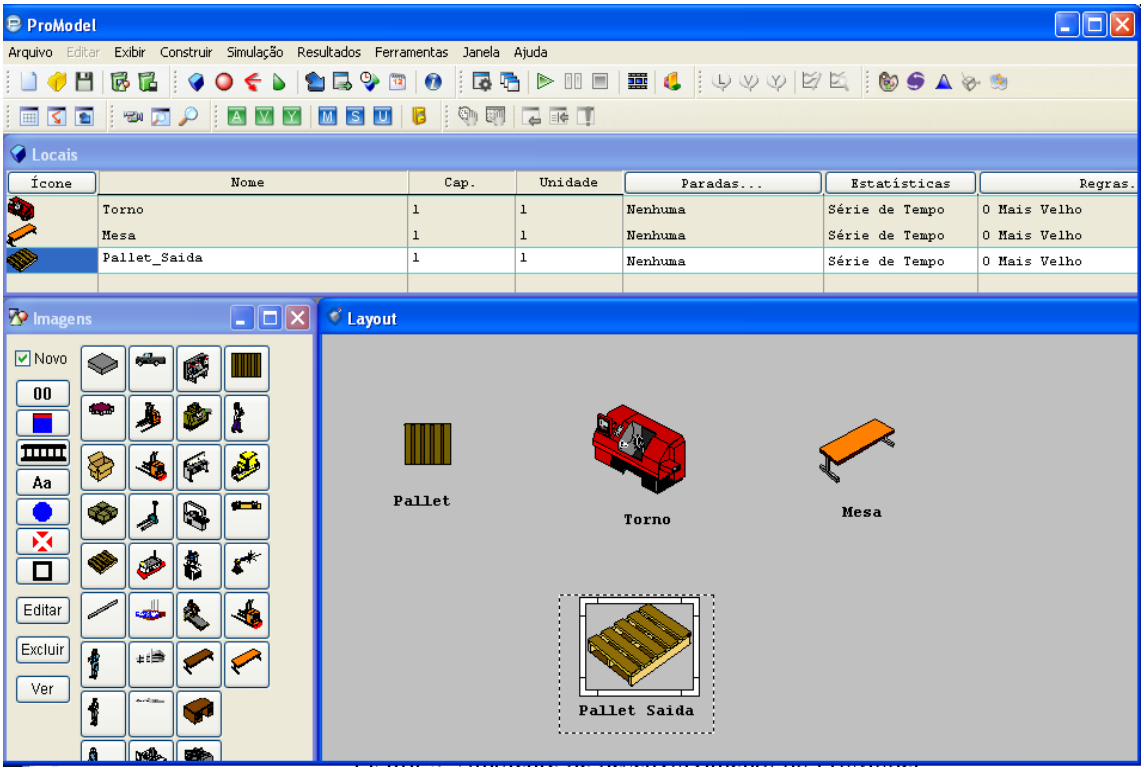

Figura 2.2: Ambiente de desenvolvimento do ProModel®.

2.1.7.4.3.URURAU

O Ururau é um ambiente de simulação construído com código aberto e distribuído sob licença de software livre. Tal característica permite seu uso acadêmico com a exploração de algoritmos internos e a criação de novas funcionalidades. O Ururau utiliza recursos gráficos para a criação de modelos de simulação utilizando a linguagem de modelagem IDEF-SIM proposta por Montevechi *et al*. (2010). Os recursos básicos do Ururau são: menu principal, barra de ferramentas, elementos de modelagem e navegador (SILVA *et al.,* 2012). A figura 2.3 apresenta a interface gráfica do Ururau, com destaque para o principais recursos disponíveis na ferramenta.

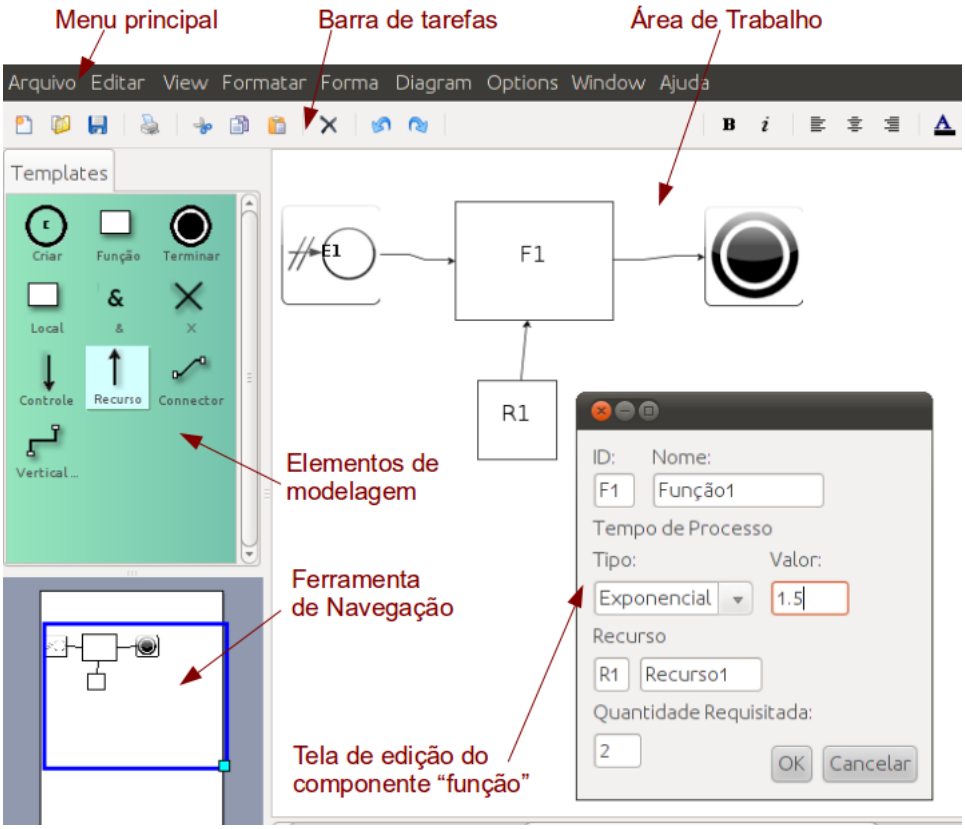

Figura 2.3: Ambiente de desenvolvimento do URURAU.

De acordo com (Peixoto *et al.* 2013), anteriormente ao Ururau, no Brasil, não se conhece citação de nenhum outro ambiente de simulação a eventos discretos que tenha sido desenvolvido com o propósito de executar modelos de simulação tanto em interface gráfica (GUI – *Graphic User Interface*) quanto em uma API – *Application Programming Interface* (de maneira complementar a GUI).

SILVA *et al.*,(2012) afirma que o Ururau oferece a flexibilidade de criar e modificar modelos usando a interface gráfica ou em linhas de código, podendo isto, ocorrer em três níveis diferentes: i) Nível Alto – recursos de ambientes de simulação (utilizando a GUI do Ururau), ii) Nível Intermediário – acessando recursos similares a linguagens de simulação (acessando o núcleo do Ururau e o JSL), e por fim , o iii) Nível Baixo – utilizando recursos de LPPG (Java). Tal característica possibilita afirmar que o Ururau oferece em uma única ferramenta, as vantagens dos ambientes de simulação, linguagens de simulação e também das LPPG.

2.1.8.O problema do trânsito

As redes de tráfego urbano apresentam alto grau de concorrência e são caracterizados pela partilha de recursos e de conflitos DOTOLI e FANTI (2006).

Diversos são os problemas da atualidade relacionados ao trânsito. López-Neri (2009) aponta que o grande número de veículos provoca problemas como: engarrafamentos, poluição atmosférica e sonora, consumo de combustível, *m,.*entre outros. Esses problemas podem ser reduzidos por meio da utilização dos atuais recursos urbanos, através da análise de desempenho sob diferentes políticas de controle de semáforos ao longo do dia.

Para Guidorizzi *et al.* (2009) o trânsito nas grandes cidades é encarado como um dos principais problemas e desafios para as próximas gerações, uma vez que dentre outras coisas, o aumento da frota de veículos é desproporcional ao crescimento de vias pavimentadas.

De acordo com Baptista e Rangel (2011), o aumento de meios de transporte motorizados, acarretou um maior volume no fluxo de veículos, que em algumas vezes continua sendo absorvido pela mesma estrutura viária das cidades.

Para Resende (2008) *apud* Guidorizzi *et al.* (2009), o trânsito em grandes cidades precisa de tratamento urgente, pois se aproxima do colapso. Fazendo uma avaliação, o autor descreve que entre as causas do problema, estão o tempo de duração dos horários de pico e a extensão do congestionamento provocado em um minuto por um evento qualquer.

A técnica mais comum para regular e gerir as áreas de tráfego urbano é o controle por meio do sinal, cujas estratégias atualmente disponíveis podem ser agrupadas em duas classes. A primeira está relacionada como os sistemas de tempo fixo e a segunda com um sistema dinâmico que responde as características do sistema (DOTOLI e FANTI 2006).

Segundo Baptista e Rangel (2011), um dos fatores que mais influencia na fluidez do tráfego automotivo em uma via semaforizada, é a má sincronia entre os temporizadores dos semáforos, gerando congestionamento de automotivos principalmente nos pontos críticos onde há interseção entre vias cujos semáforos encontram-se desregulados ou dessincronizados.

GUIDORIZZI *et al.* (2009) afirma que a simulação dentro da pesquisa operacional, é um método que apoia a tomada de decisão para problemas complexos, com muitas variáveis aleatórias, que se aplica aos problemas de gargalos formados pelo trânsito em pontos críticos das grandes cidades e que esses gargalos são similares aos gargalos das linhas de produção.

2.1.9.Sistemas de Tráfego Urbano e Simulação a Eventos Discretos

A Simulação tem sido bastante adotada pelos engenheiros e encarregados de planejar as políticas de sinalização da rede de tráfego. Na área urbana a micro simulação adapta-se melhor, pois possibilita a análise em detalhe do comportamento de veículos, o desempenho de ruas e cruzamentos e a eficácia das estratégias de controle de tráfego (LÓPEZ-NERI, 2009).

Para López-Neri (2009), a abordagem da micro simulação de um sistema de tráfego urbano pode ser considerada como um Sistema de Evento Discreto.

Dotoli e Fanti (2006) descrevem que os sistemas de tráfego urbano podem ser vistos como sistemas orientados a eventos e como sistemas assíncronos. Sua dinâmica depende das complexas interações entre o tempo de vários eventos discretos, como chegadas ou partidas de veículos nos cruzamentos, e início ou conclusão das diversas fases do tempo do sinal dos planos dos semafóricos, controlando cruzamentos.

De acordo com Timm e Kamat (2008), inúmeros fatores devem ser considerados no projeto de uma simulação para refletir com precisão as condições do mundo real. Neste caso em específico, pode ser considerado, por exemplo, a taxa do fluxo de veículos (veículos/hora), a direção provável da partida (esquerda, direita, a diante), tempo entre os sucessivos veículos atingindo a intersecção e o tempo médio de liberação da intersecção (todas essas informações podem variar de acordo com o dia).

Existem programas especializados que realizam simulação de intersecções de trânsito com base em dados do mundo real, porém alguns desses sistemas apesar de eficazes para operação e modelagem de tráfego, possuem algumas limitações. Essas ferramentas são para uso específico, o que permite modelar exclusivamente o

desempenho e o comportamento do tráfego veicular, não sendo possível de serem utilizados em diferentes domínios (TIMM e KAMAT 2008).

Guidorizzi *et al.* (2009), diz que a simulação permite que os resultados para os problemas de trânsito sejam visualizados na escala de tempo sem prejuízos financeiros. Afirma ainda que os elementos dos sistemas de tráfego estão divididos em três elementos: o usuário, o veículo e a via. Os usuários são as pessoas que participam do sistema de trânsito. Os veículos são normalizados em uma unidade padrão não sendo diferenciados entre tipos e a via que é o espaço destinado à circulação.

López-Neri (2009) descreve que o sistema de tráfego urbano é composto por entidades ou componentes, como ruas e cruzamentos de rede, usuários da estrada (veículos, pedestres, ciclistas, etc), sinais de trânsito (dinâmico: semáforo, e estático: sinal do limite de velocidade). Entidades estáticas não podem mudar o seu estado, por exemplo, sinais de trânsito como limite de velocidade, fluxo de prioridade etc, enquanto entidades dinâmicas são aquelas que podem se mover através da rede de estradas e ou mudar seu próprio estado, como carros, pedestres, semáforos, etc.

Considerando os aspectos da simulação, utilizando-se a analogia descrita por Mugnela e Netto (2012), o segmento de rua (*link*) se assemelha a uma esteira, movimentando todos os carros com igual velocidade do começo ao fim do segmento, onde são empilhados. O cruzamento (nó), que pode conter um semáforo ou não e é responsável por distribuir os carros dos segmentos, e o carro como sendo a partícula básica de carga do sistema.

Com base nos trabalhos de Helbing (2003), Nagatani e Helbing (2008) é possível afirmar que o trânsito possui padrões de comportamentos que poderiam ser estudados para serem empregados em linhas de produção dos sistemas de manufatura. Além disso, é possível verificar que a simulação a eventos discretos é aplicável em ambos os casos com o objetivo de se obter resultados e realizar análises de forma mais acessível, segura e dinâmica.

2.1.10.Redes Neurais Artificiais

No final da década de 1980 ressurgiram os sistemas de processamento paralelo e distribuído, que são sistemas de computação não algorítmica, compostos

por unidades de processamento simples (neurônios artificiais) e que calculam determinadas funções matemáticas. Estes sistemas são chamados de Redes Neurais Artificiais, (BRAGA *et al.,* 2011).

As RNA têm sido motivadas desde início pelo reconhecimento de que o cérebro humano processa informações de forma diferente do computador digital convencional. O cérebro se apresenta como um computador altamente complexo, não linear e paralelo (HAYKIN, 2001).

O modelo artificial de um neurônio biológico foi proposto por McCulloch e Pitts, em 1943, com base no que se sabia na época a respeito do neurônio biológico. Sua descrição matemática resultou em um modelo representado pela figura 2.4, com *n* terminais de entrada (dentritos) que recebem os valores x_1, x_2,... x_n, e apenas um terminal de saída *y*. Para representar o comportamento das sinapses, os terminais de entrada do neurônio têm pesos acoplados w_1, w_2,... w_n , cujos valores podem ser positivos ou negativos, dependendo das sinapses correspondentes, (BRAGA *et al.,* 2011).

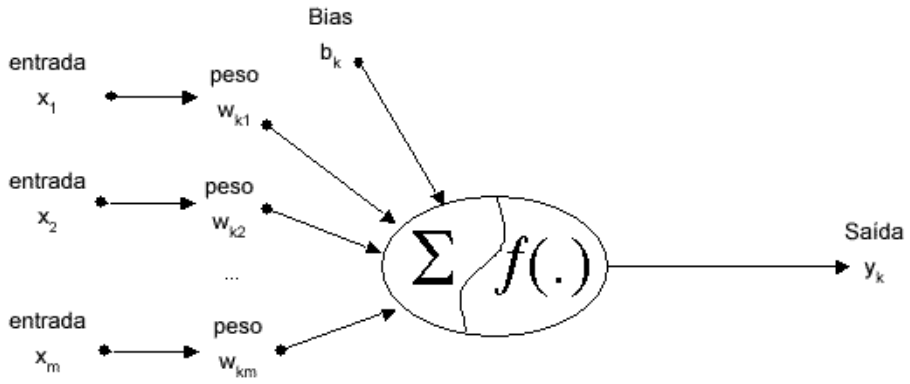

Figura 2.4: Neurônio de McCulloch e Pitts.
Fonte: Adaptado de BRAGA (2011)

2.1.10.1.Aplicação

Braga *et al.* 2011 apresenta uma relação das principais tarefas que podem ser realizadas pelas RNA, conforme pode ser visualizado no quadro 2.1.

Tarefas	Algumas Aplicações
	Reconhecimento de Caracteres
	Reconhecimento de Imagens
Classificação	Diagnóstico (médico, equipamentos,etc.)
	Análise de risco de crédito
	Detecção de fraudes
	Detecção de falhas em sistemas industriais
	Agrupamento de sequencias de DNA
	Mineração de dados
Categorização	Agrupamento de clientes
	Previsão do tempo
	Previsão financeira
Previsão	Modelagem de sistemas dinâmicos
	Previsão de sequencias de DNA

Quadro 2.1 : As principais tarefas que as RNAs podem executar
Fonte: Braga *et al.* (2011)

De acordo Heaton (2011), as RNAs são projetadas para realizarem pequenas tarefas e não uma grande aplicação. No caso de uma aplicação completa implementada com RNA, haverá uma RNA para cada tarefa específica.

Heaton (2011) afirma ainda que as RNAs atuam muito bem no reconhecimento de padrões, e apresenta a figura 2.5 como sendo a representação de uma RNA típica nestes casos.

Segundo Zuben (2003) o comportamento humano pode ser emulado por uma RNA a partir de amostras de entradas e saídas. O vetor de entrada contém o conjunto de informações recebidas por um ser humano e o vetor de saída as respectivas ações tomadas pelo ser humano com base nas informações recebidas.

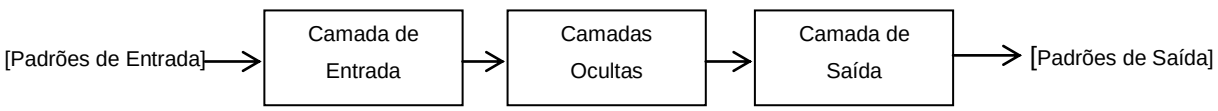

Figura 2.5: Rede Neural Artificial Típica.
Fonte: Heaton (2011).

2.1.10.2.Principais Arquiteturas de RNAs

De acordo com Haykin (2001) a arquitetura de uma RNA está relacionada com a estrutura da mesma, e esta está intimamente ligada com o algoritmo de aprendizado utilizado para treinar a rede. Três classes de arquiteturas de rede são destacadas pelo autor:

i) redes alimentadas adiante com camada única - redes *feedforward* de uma única camada – figura 2.6
ii) redes alimentadas diretamente com múltiplas camadas - redes *feedforward* de múltiplas camadas - figura 2.7.
iii) redes recorrentes.

Braga *et al.* (2011) acrescenta que a estrutura de rede *feedforward* de camada única são capazes de resolver problemas multivariáveis de múltiplas funções acopladas, porém com restrições de complexidade por ser de camada única.

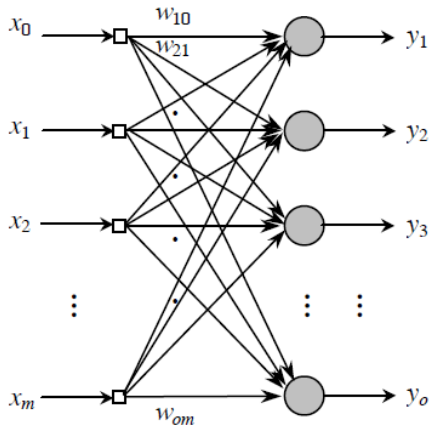

Figura 2.6: Rede *feedforward* de uma camada
Fonte: Adaptado de Braga (2011).

Já a rede *feedforward* de múltiplas camadas, permite que com a camada intermediária, a RNA ganhe uma maior capacidade computacional e universalidade na aproximação de funções contínuas.

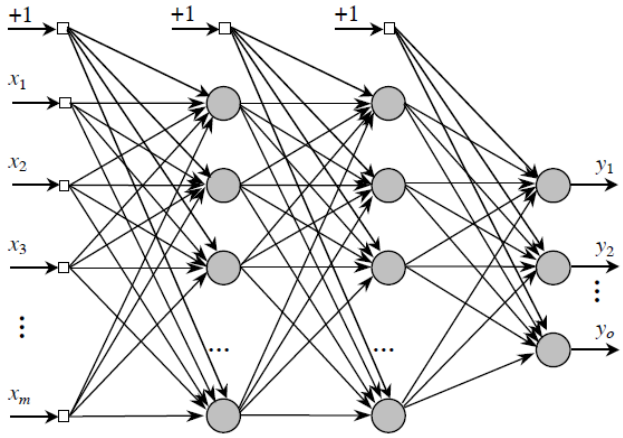

Figura:2.7 Rede *feedforward* múltiplas camadas
Fonte: Adaptado de Braga (2011).

Sobre as redes recorrentes Braga *et al.* (2011) afirma que as mesmas são utilizadas para resolução de problemas que envolvam processamento temporal, como em previsão de eventos futuros, pela sua característica dinâmica envolvendo conexões recorrentes entre neurônios.

2.1.10.3.O Perceptron

Com base no modelo do neurônio não linear de McCulloch-Pitts, Rosenblatt desenvolveu o modelo *perceptron.* O objetivo do *perceptron* é classificar corretamente o conjunto de estímulos aplicados externamente em uma de duas classes. A regra de decisão para a classificação é atribuir o ponto representado pelas entradas à cada uma das classes, dependendo da saída do *perceptron* (+1 ou -1) Haykin (2001) .

De acordo com Braga *et al.*, (2011) a partir de funções booleanas, o *perceptron* simples atua como um discriminador que divide o espaço de entrada em duas regiões, por meio de uma superfície de separação linear, desta forma, este tipo de neurônio está limitado a resolver apenas problemas lineares.

Para solução de problemas não lineares, é preciso incorporar funções de ativação (não-lineares) em cada neurônio, e a RNA é estruturada em camadas sucessivas, de modo que a resposta da camada mais externa da rede seja correspondente as respostas dos neurônios das camadas anteriores. A este tipo de

RNA de múltiplas camadas intermediárias dá-se o nome de *Perceptron* de Múltiplas Camadas (MLP – *Multilayer Perceptron*) (BRAGA *et al*, 2011).

Segundo Kovács (2002), os neurônios que recebem diretamente as entradas da rede constituem o que se chama de camada de entrada. Os neurônios que recebem como entradas as saídas da camada desta, constituem a segunda camada e assim sucessivamente até a camada final que é a camada de saída. As camadas internas são chamadas de camadas ocultas.

2.1.10.4.Aprendizado Supervisionado

Este tipo de aprendizado implica necessariamente na existência de um supervisor, que é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída calculada pela mesma, comparando-a com a saída desejada. É indicado para aplicações em problemas que se deseja obter um mapeamento entre padrões de entrada e saída (BRAGA *et al*, 2011).

2.1.10.5.Aprendizado Não Supervisionado

De acordo com Haykin (2001), para realizarmos a aprendizagem não supervisionada, podemos utilizar a regra de aprendizagem competitiva, como por exemplo, utilizando uma rede neural com duas camadas, uma de entrada e outra competitiva na qual os neurônios competem entre si, de acordo com uma regra de aprendizagem, pela oportunidade de responder às características contidas nos dados de entrada.

2.2.BIBLIOMETRIA

Costa (2010) descreve aplicação de um modelo de desenvolvimento bibliométrico denominado *webiblioming.* O modelo fornece ao pesquisador iniciante em uma área de conhecimento a seleção de um núcleo inicial de artigos para sua pesquisa bibliográfica.

O modelo apresentado por Costa (2010) considera a execução de algumas etapas: i) definição da amostra da pesquisa; ii) pesquisa na amostra com palavras-

chave; iii) identificação dos periódicos com maior número de artigos publicados sobre o tema; iv) identificação dos autores com maior número de publicações; v) levantamento da cronologia da produção; vi) seleção do núcleo de partida para a pesquisa.

O núcleo de partida deve ser contemplado por: artigos mais relevantes; identificação dos primeiros autores a escreverem sobre o tema; identificação dos últimos autores a escreverem sobre o tema; identificação dos textos mais relevantes em cada ciclo de maior produção.

Neste trabalho foi elaborado um levantamento bibliométrico adaptando o modelo proposto por Costa (2010), com o objetivo de auxiliar a seleção de um núcleo base de partida para a pesquisa. A seguir serão apresentadas informações sobre todos os passos realizados para a elaboração da pesquisa bibliográfica.

2.2.1.Levantamento Bibliométrico Inicial

Objetivando um estudo mais focado e detalhado a cerca do tema proposto neste trabalho, foram realizadas pesquisas a Base de Dados Científica *Scopus,* com acesso pelo Portal de Periódicos da Capes.

Inicialmente a pesquisa foi realizada buscando-se identificar o estado da arte na área de Simulação a Eventos Discretos (*Discrete Event Simulation*). A primeira busca realizada teve como objetivo responder sobre a possibilidade de simular padrões de comportamentos do trânsito para serem empregados nas linhas de produção das indústrias, uma vez que o trânsito pode ser considerado um ambiente propício e livre para coleta dos mais diversos tipos de dados a serem utilizados na simulação, diferentemente das linhas de produção.

Essa primeira busca ocorreu no dia 05 de fevereiro de 2013 utilizando a frase *“discrete event simulation”* que retornou 11.288 registros, cuja distribuição por tipo de produto pode ser visualizada no quadro 2.1.

Tipo de Publicação	**Quantidade de Registros**
Artigos de Conferências	5.663
Artigos	5.007
Indefinidos	198
Revisões	174

Revisões de Conferências	162
Artigos "*in Press"*	50
Livros	16
Editoriais	9
Pequena Pesquisa	4
Notas	2
Relatório	1
Errata	1
Relatório resumido	1

Quadro 2.1: Distribuição dos Registros Encontrados por Tipo de Documento
FONTE: Elaborado pelo autor

Ao realizar o refinamento dos registros encontrados filtrando apenas os artigos publicados em periódicos, foram obtidos 5.007 registros. O quadro 2.2 apresenta a relação dos periódicos com maior número de artigos publicados na área em estudo.

Título do periódico	
Simulation	85
International Journal of Production Research	5
ACM Transactions on Modeling and Computer Simulation	9
Journal of the Operational Research Society	8
Xitong Fangzhen Xuebao Journal of System Simulation	4
Simulation Modelling Practice and Theory	4
Computers and Industrial Engineering	4
IEEE Transactions on Automatic Control	2
European Journal of Operational Research	7
Discrete Event Dynamic Systems Theory and Applications	6

Quadro 2.2: Distribuição das Publicações por Periódicos
FONTE: Elaborado pelo autor

O quadro 2.3 traz a relação dos autores com maior número de artigos publicados na área da pesquisa em periódicos.

Autor	**Abr.**	**Qd.**
Zeigler	B.P.	46

Wainer	G.	24
Kim	T.G.	21
Paul	R.J.	20
Cassandras	C.G.	18
Azzaro-Pantel	C.	18

Quadro 2.3: Distribuição de Registro de Artigos por Autores
FONTE: Elaborado pelo autor

É possível observar que há uma tendência crescente em relação à quantidade de publicações na área ao longo do tempo, esta relação pode ser visualiza pela figura 2.8.

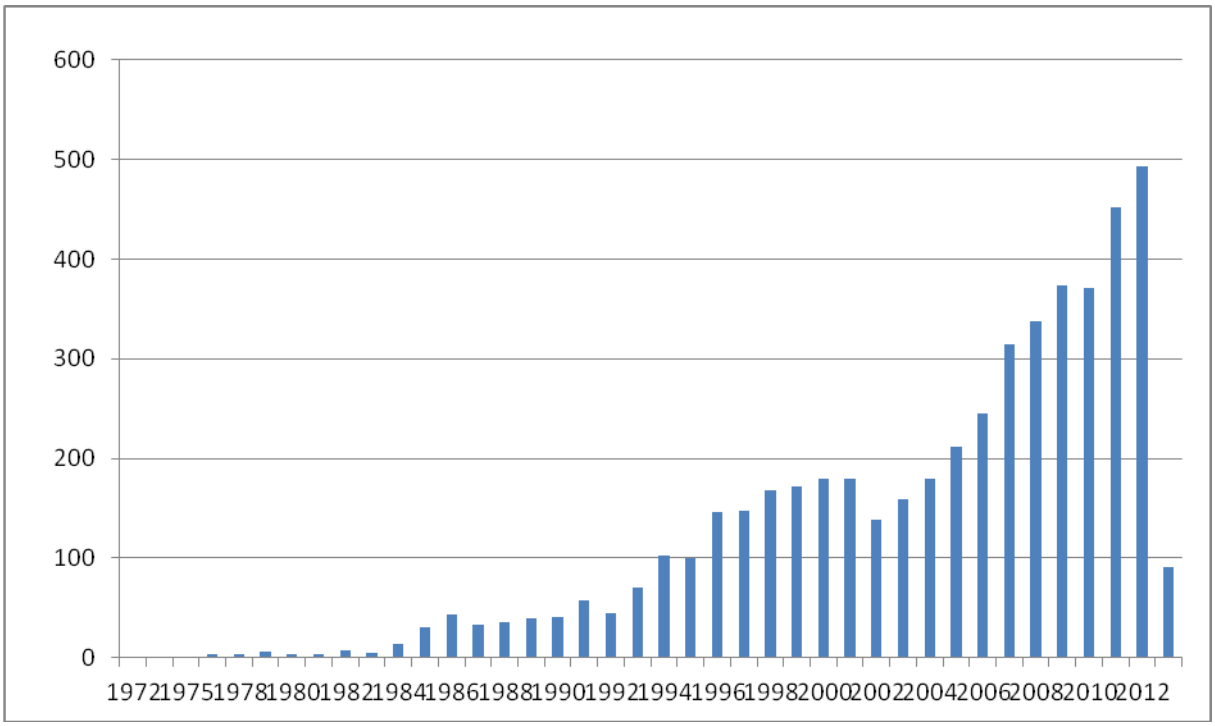

Figura 2.8: Distribuição da Quantidade de Artigos Publicados por Ano
FONTE: Elaborado pelo autor

O quadro 2.4 apresenta a distribuição de artigos publicados em cada país, demonstrando que mais de 35% do total de publicações dos periódicos estão nos Estados Unidos, vindo em seguida a China com mais de 10%, Reino Unido com 9%, o Canadá com aproximadamente 6% e os demais com um número bem inferior. O Brasil aparece na 17ª posição do *ranking* de publicações.

PAÍS	QD
Estados Unidos	1776
China	523
Reino Unido	453
Canadá	291
França	266
Alemanha	244
Itália	221
Holanda	154
Coréia do Sul	143
Espanha	110
Japão	110
Austrália	103
Taiwan	98
Índia	93
Suécia	90
Singapura	75
Brasil	58

Quadro 2.4: Distribuição de Artigos Publicados por País
FONTE: Elaborado pelo autor

Em relação à área em que os artigos são publicados, pode-se destacar a Engenharia aparecendo em primeiro lugar, correspondendo a mais de 50% do total das publicações. O quadro 2.5 apresenta a relação das 10 áreas com maior concentração de publicações.

ÁREA	QD
Engenharia	2593
Ciência da Computação	1942
Matemática	1267
Ciências da Decisão	730
Medicina	332
Negócios, Gestão e Contabilidade	312
Física e Astronomia	309
Bioquímica, Genética e Biologia Molecular	187
Ciências da Terra e Planetária	183
Ciências Social	178

Quadro 2.5: Distribuição de Artigos por Área
FONTE: Elaborado pelo autor

Refinando a pesquisa na área “Engenharia” foi possível encontrar 2593 registros. A figura 2.9 apresenta de forma bem resumida a relação dos 10 autores com maior número de publicações. É possível observar que Zeigler continua aparecendo como principal autor também nesta área.

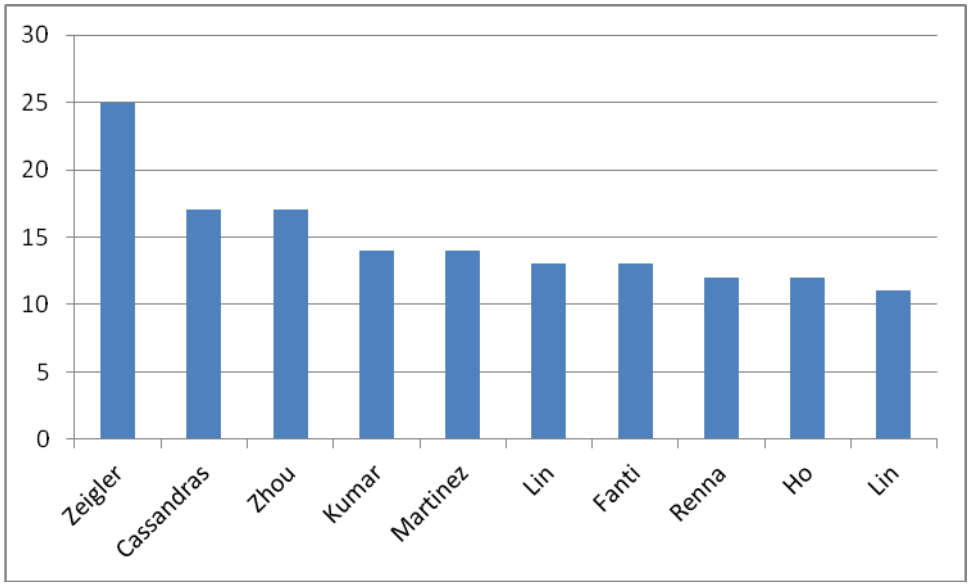

Figura 2.9: Distribuição da Quantidade de Artigos Publicados por Autor na Área de Engenharia
FONTE: Elaborado pelo autor

Na área “Engenharia” é possível observar o mesmo comportamento de aumento das publicações ao longo dos anos, como apresentado no figura 2.10.

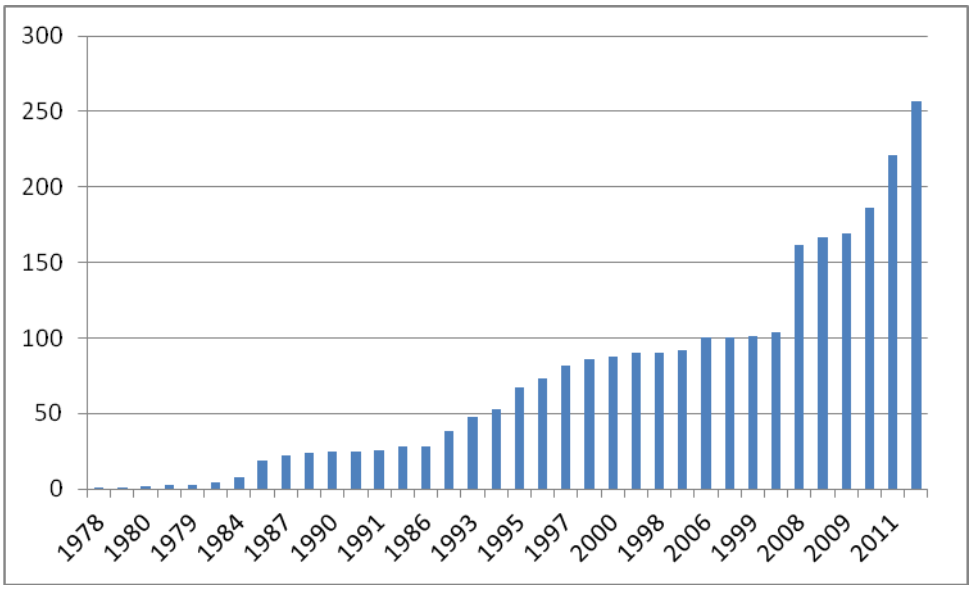

Figura 2.10: Distribuição da Quantidade de Artigos Publicados por Ano na Área de Engenharia.
FONTE: Elaborado pelo autor

Um refinamento ainda maior pôde retornar resultados mais específicos ao se filtrar dentro da área de Engenharia as expressões *“Simulation”* ou “*Discrete Event Dynamic Systems Theory and Applications”.* Neste caso, foi possível obter 196

registros (140 artigos relacionados ao termo "*Simulation*" e 56 artigos relacionados ao termo "*Discrete Event Dynamic Systems Theory and Applications"),* cujas publicações demonstram um comportamento um pouco flutuante a cada ano, conforme pode ser visualizado pela figura 2.11

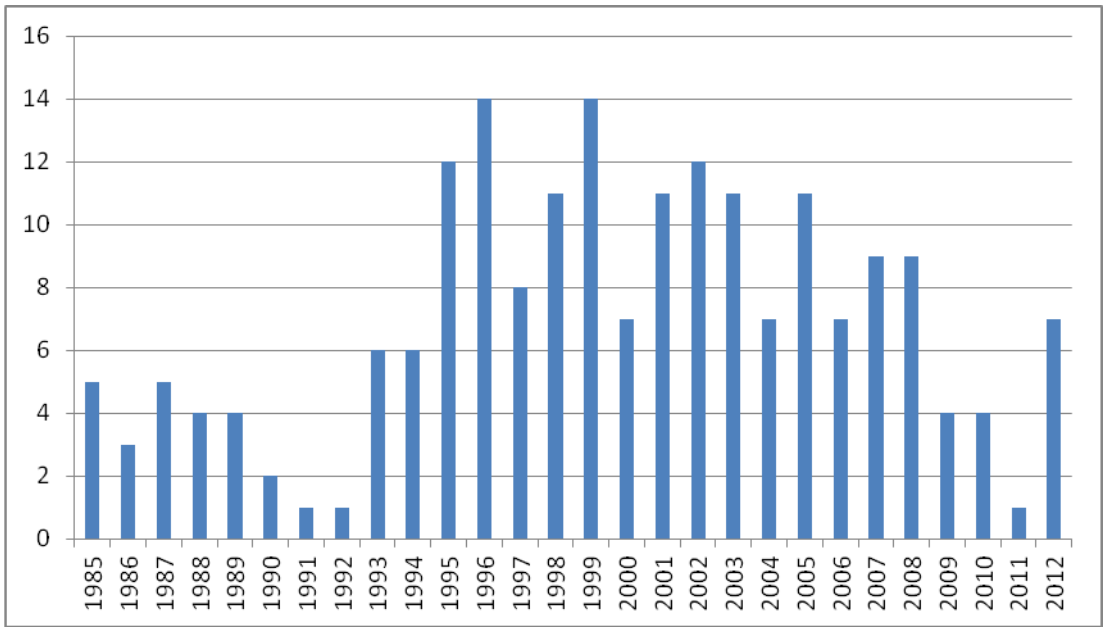

Figura 2.11: Distribuição da Quantidade de Artigos Publicados por Ano.
FONTE: Elaborado pelo autor

Iniciando-se no ano de 1985 com 04 publicações, a produção ganhou maior representatividade apresentando ciclos mais acentuados como pôde ser visto no Gráfico 2.4 e descritos abaixo:

- 1995-1996 (ápice)
- 1998-1999 (ápice);
- 2001-2003;
- 2005;
- 2007-2008.

Com o objetivo de se obter resultados ainda mais próximos dos objetivos da pesquisa, foram realizadas consultas com os termos envolvendo simulação a eventos discretos, trânsito e linhas de produção ou cadeias de suprimentos.

Ao pesquisar o termo *"discrete event simulation and transit and supply chain"* somente 01 resultado foi obtido, porém o registro não era de periódico, mas sim de

Conferência (*21st European Conference on Modelling and Simulation Simulations in United Europe Ecms*) da área de Matemática e do ano de 2007.

O mesmo ocorreu com o termo *"discrete event simulation and transit and production line"*, apontando somente uma ocorrência de artigo de conferência no ano de 2005.

A pesquisa foi alterada utilizando o termo *"discrete event simulation and traffic and supply chain"* , e esta retornou 14 publicações, sendo que somente 6 eram referentes a área de Engenharia e desses, somente 4 eram artigos de periódicos. A relação dos trabalhos obtidos nesta pesquisa é apresentada no Quadro 2.1.

Assi, T.-M., Rookkapan, K., Rajgopal, J., Sornsrivichai, V., Brown, S.T., Welling, J.S., Norman, B.A., Connor, D.L., Chen, S.-I., Slayton, R.B., Laosiritaworn, Y., Wateska, A.R., Wisniewski, S.R., Lee, B.Y. ***How influenza vaccination policy may affect vaccine logistics.*** (2012) Vaccine, 30 (30), pp. 4517-4523. http://www.scopus.com/inward/record.url?eid=2-s2.0-84862016836&partnerID=40&md5=6909caf3d6393e03b797aed56ffc76e6 DOCUMENT TYPE: Article SOURCE: Scopus
Byrne, P.J., Heavey, C., Ryan, P., Liston, P***. Sustainable supply chain design: Capturing dynamic input factors*.**(2010) Journal of Simulation, 4 (4), pp. 213-221. http://www.scopus.com/inward/record.url?eid=2-s2.0-78349285926&partnerID=40&md5=296b2d93eae130109822a4fe200a74bd DOCUMENT TYPE: Article SOURCE: Scopus
Ringhofer, C*. **A level set approach to modeling general service rules in supply chains*** (2010). Communications in Mathematical Sciences, 8 (4), pp. 909-930. Cited 1 time. http://www.scopus.com/inward/record.url?eid=2-s2.0-77956904021&partnerID=40&md5=d4c6b5b8a4c11254ba7add4d34d8d55b DOCUMENT TYPE: Article SOURCE: Scopus
Whitworth, M.H. ***Designing the response to an anthrax attack*** (2006). Interfaces, 36 (6), pp. 562-568. Cited 10 times. http://www.scopus.com/inward/record.url?eid=2-s2.0-3845393816&partnerID=40&md5=308ce6e2f6ee03b2bb979a47952cfdff DOCUMENT TYPE: Article SOURCE: Scopus

Quadro 2.6: Relação de Artigos Contendo os Termos Base da Pesquisa.
FONTE: Elaborado pelo autor

2.2.2.Conclusão do 1º Levantamento Bibliométrico

Embora o assunto Simulação a Eventos Discretos seja de amplo interesse pela comunidade acadêmica, como pôde ser visualizado pelos resultados das consultas na base de dados *Scopus*, não foi possível encontrar indexado nesta base um grande número de artigos periódicos ao se aplicar na busca todos os termos referentes ao objetivo dessa primeira consulta: verificar possibilidade de simular padrões de comportamentos do trânsito para serem empregados nas linhas de produção das indústrias.

Contudo, foi possível notar que com o passar do tempo o tema Simulação a Eventos discretos, tem sido objeto de estudo de muitos autores, já que tem havido um aumento crescente na quantidade de publicações ao longo dos anos.

Dentre os países que mais produzem nesta área é possível observar que os Estados Unidos lideram na publicação de artigos, enquanto os autores Zeigler e Cassandras aparecem com número expressivo de publicações para as diferentes alternativas de buscas empregadas, mostrando grande nível de especialização dos autores nesta área.

É possível concluir que esta análise bibliométrica permitiu dentre outras coisas, criar recortes importantes do conjunto de publicações indexadas na base de dados científica *Scopus*, permitindo um melhor plano de execução do levantamento bibliográfico na pesquisa e um melhor encaminhamento dos rumos do trabalho em sua fase inicial.

2.2.3.Segundo Levantamento Bibliométrico

Realizando uma segunda pesquisa na base de dados *Scopus*, no dia 17 de Abril de 2013, considerando o termo "*Discrete Event Simulation*", foram retornadas 11.452 publicações, ou seja, 164 publicações a mais que a pesquisa realizada em Fevereiro de 2013.

Relacionado a publicações em periódicos, observa-se que nos dois últimos meses o crescimento numérico foi de 96 artigos, já que nesta segunda pesquisa, foram obtidos 5.103 registros de artigos, dentre os quais 5.043 são artigos já publicados em periódicos e 60 artigos no prelo.

De Janeiro de 2013 até a data dessa pesquisa, foi possível encontrar o registro de 142 publicações em periódicos nesta área, o que representa um grande número de artigos publicados em menos de 4 meses. Mantendo-se a proporção mensal, é possível que no ano de 2013 a quantidade de publicações supere aos anos anteriores mantendo o crescimento com características exponenciais.

Fazendo-se outro conjunto de análises sobre o tema, verifica-se que a primeira ocorrência desse tipo de publicação foi no ano de 1972 com o artigo intitulado: *"Digital simulation of iodine kinetics during inhibition phase"*.

Dentre os 5.103 registos de artigos, o que recebeu maior número de citações (1052) é do ano de 1999, do periódico *Automatica*, intitulado "*Control of systems integrating logic, dynamics, and constraints*" dos autores Bemborad e Morari, conforme informações do Apêndice A.

Novamente com o objetivo de se verificar o andamento da produção de trabalhos envolvendo o tema Simulação a Eventos Discretos aplicados a problemas de tráfego urbano, foi realizada uma nova consulta na base de dados *Scopus* contendo os seguintes termos: *"Discrete Event Simulation AND Urban Traffic"*.

Como resultado da busca, foram obtidos 51 registros, sendo que somente 19 representam artigos científicos, enquanto 29 são publicações de eventos, 02 revisões e uma publicação indicada como "não definido". Limitando a busca por artigos publicados em periódicos, temos a lista que aparece no Apêndice B, onde são apresentadas informações como nomes dos autores, título da publicação, ano em que foi publicado, nome do periódico e a quantidade de citações que cada um dos trabalhos recebeu.

Da relação apresentada pela Tabela 2.7, o trabalho que apresentou maior número de citações (38), é o artigo intitulado: "*An urban traffic network model via couloured timed Petri nets"* dos autores Dotoli e Fantini publicado no ano de 2006.

Em relação ao grau de relevância quanto ao tema pesquisado, a base de dados *Scopus*, aponta o trabalho do ano de 2009, *"Hierarchical agente-based modelling: A Guadalajara city case study on urban traffic simulation*", do autor López-Neri, conforme pode ser visto no Apêndice C, que apresenta as 5 publicações mais relevantes para a consulta.

2.2.4.Núcleo de Partida para a Pesquisa Bibliográfica

O núcleo de partida para a elaboração do estudo bibliográfico deste trabalho pode ser consultado no Apêndice D e foi elaborado a partir de uma adaptação do modelo proposto por Costa (2010).

Inicialmente foram realizadas consultas mais abrangentes sobre o tema base de estudo, no qual foram selecionados alguns artigos de maior adequação.

Posteriormente a pesquisa foi sendo refinada e a cada refinamento foram sendo selecionadas publicações também com base no modelo de Costa (2010).

Além disso, foram selecionados trabalhos cujos conteúdos possuem aderência aos temas tratados, com base em troca de informações de pessoas que trabalham com o assunto.

2.3.ESTADO DA ARTE

Neste tópico serão apresentados trabalhos que representam o estado da arte para a pesquisa, considerando as seguintes questões:

a. Analogias entre Sistemas de Tráfego Urbano e Linhas de Produção;
b. Utilização de Redes Neurais Artificiais para Tomadas de Decisão em Modelos de Simulação a Eventos Discretos.

2.3.1.Analogias entre Sistemas de Tráfego Urbano e as Linhas de Produção

2.3.1.1. O Trabalho de Baptista e Rangel (2013)

Baptista e Rangel (2013) apresentam em seu trabalho a integração e comunicação em tempo real de um modelo de simulação a eventos discretos com um sistema de controle automático. Para tal, os autores lançaram mão da natureza dinâmica e estocástica de um cruzamento semaforizado do trânsito para desenvolvimento do modelo de simulação. O estudo pode auxiliar na tomada de decisão e no tratamento de dados de produção oriundos do chão de fábrica, neste caso, o modelo de simulação pode tratar basicamente de três elementos em uma análise: a via, o cruzamento e o carro. A via se assemelha a uma esteira, movimentando todos os carros com igual velocidade do começo ao fim do segmento. O cruzamento se assemelha a um nó onde os carros são distribuídos de acordo com a operação dos semáforos. O carro é a entidade que flui dinamicamente pelo modelo. Os autores esperam que a integração entre simulação a eventos discretos e sistemas de controle automático possam ser utilizados para análise de fluxo de veículos em áreas urbanas e também que situações comuns em indústrias,

como sistemas de transporte de produtos em esteiras em linhas de montagem, possam ser analisadas e testadas com a abordagem apresentada no trabalho.

2.3.1.2. O Trabalho de Ringhofer (2010)

Ringhofer (2010) desenvolveu seu trabalho com objetivo de modelar um conjunto de políticas de agendamento no âmbito das leis de conservação hiperbólicas para cadeias de suprimentos gerais. A justificativa está no fato de que nem todas as partes da cadeia são tratadas de forma idêntica, e muitas vezes não podem necessariamente serem processadas em sequência, por exemplo, quando mais de um produto é produzido ao mesmo tempo, ou quando cada parte deve ser entregue com uma data de vencimento ou momento que já tenha sido gasto no processo, sendo este fato bem relevante, uma vez que pode ocorrer do produto perecer e/ou ter seu valor diminuído ao longo do tempo.

Para se tomar a decisão sobre quais partes processar primeiro é necessário que o processo seja regido por alguma política ou regra de serviço.

Inicialmente o autor cita que os modelos de leis de conservação para cadeias e redes de suprimentos utilizam uma analogia com os modelos de fluxo de tráfego.

A analogia se baseia no fato de que as peças passando pela cadeia são comparadas aos veículos que viajam numa estrada virtual (a fase do processo), em que entram como matéria prima (fase $x = 0$) e deixam a cadeia de suprimentos como produtos acabados na fase $x=X$.

Cada estágio de evolução ao longo da cadeia é descrito por uma lei de conservação, cuja função de fluxo pode ser derivada a partir de heurísticas, teoria das filas ou modelos macroscópicos que descrevem o comportamento individual de cada parte.

O modelo geral considerado no artigo se baseia na função de prioridade que exige a escolha da prioridade da função (modelando a política de escalonamento) e da velocidade, que descreve o fluxo. Esses assuntos são tratados passo à passo no trabalho.

Além disso, o autor descreve que para escolher atributos e políticas para partes individuais é necessário escolher regras de serviço que dependem da aplicação e para tal, o mesmo trabalha considerando: o tempo do ciclo (tempo decorrido desde a entrada no sistema), tempo de cada data de vencimento (quando a peça deverá ser entregue e o tempo que resta para essa data) e o tipo da peça.

São utilizadas formulações matemáticas derivadas (leis de conservação hiperbólica) e teoria cinética para apresentação de cada situação a ser trabalhada no problema.

Foram desenvolvidos algoritmos também baseados em formulações matemáticas para a realização da simulação a eventos discretos. As simulações foram realizadas utilizando políticas de escalonamento FIFO (*First In First Out*) e em políticas mais complexas considerando situações determinísticas e também estocásticas.

Desta forma foi possível comparar os resultados levando em consideração a natureza do processo (determinístico ou estocástico) como também a influência das políticas de escalonamento aplicadas.

O sistema considerado no estudo é composto de quarenta processos em que foram definidos entre outras coisas, o tempo de *thoughput* e a capacidade de cada um.

Ao final o autor conclui que é possível que uma cadeia de suprimentos seja regida por uma classe geral de regras de serviços com base em atributos de priorização, tendo sido modelado por um conjunto de leis de conservação hiperbólica.

Este trabalho mostrou-se de grande relevância para a pesquisa uma vez que apresenta conceitos bem próximos dos conceitos base da pesquisa, como por exemplo, a analogia do fluxo nas cadeias de suprimentos e o trânsito, aplicações de politicas de escalonamento (FIFO e com base em prioridades), tempos de ciclo, tempos de chegadas, variáveis estocásticas entre outras.

2.3.1.3. O Trabalho de Helbing (2003)

Helbing (2003) apresenta informações importantes em que relaciona conceitos das redes de abastecimentos das linhas de produção com tráfego de veículos. Em seu trabalho, o autor afirma que recentemente outros trabalhos desenvolvidos por economistas, cientistas de tráfego, matemáticos e físicos tem salientado que métodos utilizados para o estudo da dinâmica de tráfego são também de uso potencial para o estudo das redes de abastecimento.

O autor apresenta uma relação matemática entre o Efeito Chicote, frequente em sistemas de abastecimento e o Páre e Siga dos sistemas de tráfego, mostrando que a analogia entre a cadeia de suprimentos e modelos de tráfego diz respeito a sua estrutura matemática, porém ressalta que a interpretação deve ser diferenciada e que esta relação pode dar dicas de como métodos utilizados com sucesso na investigação de modelos de tráfego, pode ser generalizada para o estudo de redes de abastecimento.

Ao concluir o trabalho o autor afirma que a razão para as semelhanças entre os sistemas de produção e de trânsito é a presença de entidades dinâmicas (pessoas, objetos), que interagem de um modo não linear, e a existência da competição por recursos limitados (como capacidade, tempo ou espaço).

Este trabalho contribui positivamente, uma vez que destaca diversas similaridades entre os sistemas, destacando que matematicamente todos podem também ser resolvidos da mesma forma, bastando a realização de uma interpretação adequada a cada situação.

2.3.2. Utilização de Redes Neurais Artificiais para Tomadas de Decisão em Modelos de Simulação a Eventos Discretos

2.3.2.1. O Trabalho de Silva *et al.* (2012)

O trabalho desenvolvido por Silva *et al.*(2012) teve como objetivo empregar RNAs para representar o comportamento de pessoas em modelos de simulação a eventos discretos em situações que é impossível determinar aspectos lógicos da tomada de decisão.

Segundo os autores, diversos pesquisadores vêm buscando formas de melhorar o desenvolvimento de modelos de simulação de forma a aproximá-los mais da realidade dos sistemas representados, que muitas vezes, dependem de ações de pessoas. Alguns destes pesquisadores afirmam que criar representações realísticas das decisões humanas é uma tarefa extremamente difícil.

A dificuldade em se programar estes modelos está relacionada a alta complexidade do pensamento humano e a necessidade de utilização de recursos não triviais de *softwares* de simulação.

Os autores afirmam que a maioria dos *softwares* de simulação comerciais não fornece um conjunto necessário de funções que seriam úteis para modelar o comportamento humano e, além disso, inviabiliza que novos recursos sejam desenvolvidos.

Desta forma, os autores desenvolveram um módulo com uma RNA treinada para substituir uma operação lógica de decisão tradicionalmente utilizada nos modelos de simulação discreta.

Este módulo foi denominado Módulo Inteligente, e se comunicava com o ambiente de desenvolvimento de modelos de simulação a eventos discretos Ururau. O JAVANNS foi utilizado para a criação e treinamento da RNA.

Os autores criaram um modelo de simulação hipotético, que permitiu a avaliação da simulação. Foram realizados experimentos que puderam demonstrar a a viabilidade de aplicação de RNA para representar decisões humanas em modelos de simulação a eventos discretos.

Foi possível observar uma grande flexibilidade em relação ao Ururau, que possibilitou a comunicação com o módulo inteligente desenvolvido.

2.3.2.2. O Trabalho de Bergmann *et al.* (2014)

Bergmann *et al.* (2014), utilizaram RNA em conjunto com um sistema de simulação. No caso do trabalho dos autores, a geração dos modelos ocorreu de forma automática, e se baseou na aquisição de dados armazenados em sistemas como ERP (*Enterprise Resource Planning*).

Segundo os autores, uma dificuldade encontrada nesta tarefa é a representação do comportamento dinâmico dos sistemas de manufatura. Diversas

informações sobre estes comportamentos, como estratégias de controle, frequentemente não são armazenadas nos sistemas de TI.

Neste caso é preciso a utilização de algoritmos específicos para extrair comportamentos complexos, e muitas vezes, é necessário um especialista em simulação para adicionar manualmente o comportamento detalhado ao modelo que foi gerado automaticamente.

A metodologia apresentada pelos autores se baseia em aproximar o comportamento dinâmico utilizando RNA, ao invés de tentar determinar representações exatas. Desta forma, eles delegaram a tomada de decisão à RNA, sempre que não foi possível representá-la no modelo devido a falta de conhecimento suficiente sobre o comportamento do sistema real.

Segundo os autores, as estruturas de dados e algoritmos para RNA normalmente não estão disponíveis em ambientes de simulação, ou em Sistemas de Informação (SI). Desta forma, foi necessário optar entre programá-las na linguagem do simulador, ou usar a interface disponível no *software* de simulação para ligar bibliotecas externas.

Visando um menor custo de implementação e validação, os autores decidiram utilizar bibliotecas externas para ligar o *software* de simulação *Plant Simulation* ao *Neural-Network Toolbox* integrado ao *software* Matlab.

Com este trabalho, os autores concluíram que podem utilizar RNAs para aproximação de regras de decisão diretamente na simulação. Além disso, afirmam que a integração de RNA é factível em todos os *softwares* de simulação que suportam interfaces de bibliotecas externas.

Segundo os autores, este trabalho representa um primeiro passo para utilização de RNA no contexto de geração de modelos de simulação de forma automática, uma vez que as abordagens existentes não conseguem reproduzir o comportamento dinâmico do sistema real.

2.4.CONCLUSÃO DA REVISÃO DA LITERATURA

A revisão da literatura permitiu dentre outras coisas:

i) Realizar um apanhado geral a respeito de simulação, diferenciando a Simulação de Monte Carlo, Simulação Contínua e Simulação a Eventos Discretos, que faz parte do escopo deste trabalho;

ii) Identificar as mais diversas áreas de aplicação da Simulação a Eventos Discretos;

iii) Descrever um breve relato histórico sobre o surgimento e a evolução da Simulação;

iv) Realizar um levantamento a respeito das LPPG que podem ser utilizadas na simulação, bem como das Bibliotecas e Ambientes de Simulação;

v) Relatar os problemas do trânsito e destacar a aplicação da Simulação nos sistemas de tráfego urbano;

vi) Apresentar conceitos e informações relevantes sobre as Redes Neurais Artificiais;

vii) Realizar um estudo bibliométrico a respeito da produção científica em torno do tema em estudo, podendo verificar diversas métricas relacionadas a autores, países de maior publicação, periódicos e áreas com maior concentração de trabalhos publicados, dentre outros;

viii) Destacar e relatar dados de alguns trabalhos representam grande relevância e contribuição para o estado da arte do estudo.

Desta forma, pode-se afirmar que a revisão da literatura permitiu que fossem coletados dados e informações que constituem a base para o desenvolvimento de um modelo de simulação a eventos discretos, com a utilização de Inteligência Computacional acoplado em módulo de decisão no software Ururau, que é o objetivo deste trabalho.

3. MATERIAIS E MÉTODOS

Para desenvolvimento deste trabalho a seguinte metodologia foi empregada:

Inicialmente foi realizada uma revisão bibliográfica partindo de uma análise de artigos de periódicos consultados na base de dados *Scopus* acessível pelo portal da CAPES. Além disso, artigos relacionados a área de Simulação a Eventos Discretos apresentados em congressos importantes também foram utilizados como fonte de consulta e pesquisa.

Após essa etapa inicial, o sistema a ser modelado foi definido e um modelo conceitual foi criado utilizando-se uma metodologia própria para modelagem conceitual.

Na etapa seguinte, o código fonte do Ururau foi estudado e se iniciou o processo de desenvolvimento dos recursos de RNA acoplados ao ambiente de simulação. Um *framework* para criação de redes neurais artificiais foi estudado e um módulo inteligente foi construído para possibilitar o treinamento de RNA aplicáveis aos mais diversos sistemas a serem modelados.

Após a etapa de desenvolvimento dos recursos de RNA, bem como a etapa de testes e avaliações, uma RNA foi treinada com base no sistema anteriormente definido.

Os resultados obtidos após a execução do modelo foram avaliados e os resultados e conclusões seguem descritos neste trabalho.

3.1.SISTEMAS DE CONTROLE DE TRÁFEGO INTELIGENTES

Ahmane *et al.* (2013) afirma que atualmente as estratégias controle de tráfego urbano tradicionais não podem satisfazer as exigências das cidades modernas. O aumento crescente dos congestionamentos em todo o mundo, não só diminui a eficiência do transporte, mas também aumenta a poluição atmosférica e consumo de combustível. Este fato levanta a questão de saber se as infraestruturas fornecidas correspondem à demanda de tráfego, além disso, é possível notar que investir em novas infraestruturas é cada vez mais caro.

Um esquema geral para controle de tráfego urbano é apresentado por Basile *et. al.* 2012, conforme pode ser visto na figura 3.1. A área urbana a ser controlada, que pode ser, por exemplo, duas intersecções semaforizadas, juntamente com os sensores e o atuador é visto como uma planta. Os sensores detectam diversas informações, que servirão de entradas para o subsistema Observador. O Observador possui o modelo detalhado do sistema de tráfego para controle da área urbana, cujas informações são selecionadas para que o Controlador realize as otimizações em tempo real AHMANE *et al.* (2013).

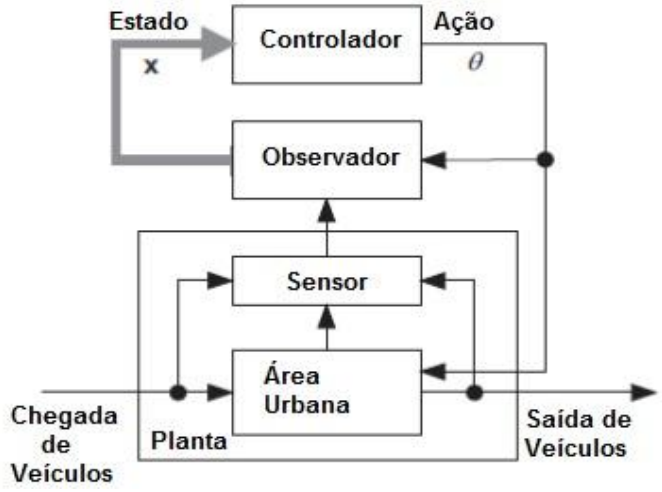

Figura 3.1: Esquema Geral para Controle de Tráfego Urbano.
Fonte: adaptado de BASILE *et. al.* 2012

De acordo com Timm e Kamat (2008) o planejamento urbano busca analisar o projeto de interseções, com o objetivo de otimizar o fluxo de tráfego e minimizar o tempo perdido. As vantagens dessa otimização está na redução do consumo de combustível, redução nas emissões de gases nocivos e redução do tempo perdido pelo impasse no tráfego.

Sistemas de Transporte Inteligentes são considerados a chave para aumentar a capacidade das infraestruturas viárias existentes (Ahmane *et al.* 2013).

Segundo Basile *et al.* 2012, modelos de simulação de tráfego rodoviário, sevem para diferentes propósitos, como por exemplo, para apoiar o planejamento das redes rodoviárias e configurações de semáforos, ou podem ser utilizados para o controle *on line* de sistemas complexos de orientação de tráfego ou para detecção de incidentes.

Para o controle do sistema de tráfego em tempo real, várias técnicas podem ser utilizadas, requendo que o sistema seja bem modelado. Dentre as técnicas de controle possíveis de serem aplicadas está a Rede Neural Artificial, que segundo Basile *et al.*(2012) pode obter uma boa precisão, porém há um custo na fase de treinamento da rede.

3.2.A RELAÇÃO ENTRE SISTEMAS DE PRODUÇÃO E SISTEMAS DE TRÂNSITO

Em recente trabalho, Baptista e Rangel (2013) analisaram o desempenho de um sistema de controle automático através de um modelo de simulação a eventos discretos. O modelo de simulação, no caso, representava o fluxo de veículos de uma via urbana. Os autores lançaram mão da semelhança entre os fenômenos dinâmicos e estocásticos associados ao fluxo de veículos nos cruzamentos e puderam avaliar o sistema de controle. A análise de tal sistema pôde então representar de forma análoga o comportamento existente em diversas situações típicas encontradas em linhas de montagens industriais. A principal diferença é que as vias urbanas são públicas e os autores puderam coletar os dados de forma livre.

O estudo de Ringhofer (2010) também se utiliza de analogias entre cadeias e redes de suprimentos e os sistemas de fluxo de tráfego. De acordo com o autor a analogia se baseia no fato de que as peças passando pela cadeia são comparadas

aos veículos que viajam numa estrada virtual (a fase do processo), em que entram como matéria prima e deixam a cadeia de suprimentos como produtos acabados.

De acordo com Barbosa *et al.* (2009), a formação de filas tem sido um fenômeno rotineiro na vida atual, ocorrendo nas mais diversas áreas como na indústria (por exemplo, uma peça aguardando para ser lixada ou polida), um avião aguardando para decolar, um programa de computador esperando para ser executado e também uma fila de pessoas aguardando algum serviço.

No caso dos sistemas de trânsito, o problema de formação de filas não é diferente. Segundo Moita e Almeida (2012) existem vários fatores que contribuem para o congestionamento nas vias, tornando-as quase que intrafegáveis principalmente nos horários de fluxo intenso devido à formação de filas. Quando a demanda é alta e o atendimento é deficitário, surgem as filas que praticamente estancam o fluxo do sistema.

Segundo Barbosa *et al. (*2009) muito do que se conhece atualmente sobre o fenômeno de formação de filas é atribuído ao engenheiro A. K. Erlang pela análise dos painéis de controle telefônico dinamarquês pela concepção da Teoria da Formação das Filas, tendo sido uma das melhores aplicações desta teoria a análise do fluxo de automóveis nas rodovias dos Estados Unidos, permitindo verificar o número de pistão, número de semáforos, entre outros, a fim de aperfeiçoar o fluxo de tráfego.

Bitran e Morabito (1995) acrescentam que a partir do trabalho de Erlang em 1917, outras aplicações têm aparecido em diversas áreas desde então, como por exemplo: comunicação, computação, transporte, produção, manutenção, biologia (redes neurais), saúde (modelos comportamentais), química e materiais (polimerização), entre outras aplicações abrangentes no uso de redes de filas para representar sistemas de manufatura.

No caso do trabalho de Bitran e Morabito (1995), os autores fazem um exame dos modelos de redes de filas abertas aplicados a sistemas de manufatura discretos (*job shop)* e descrevem que cada nó (estação) contém os seguintes elementos: (i) processo de chegada, (ii) processo de serviço e (iii) fila de espera. Os mesmos autores afirmam que os modelos de filas são motivados por casos em que o processo de chegada ou o processo de serviço, ou ambos são probabilísticos,

resultando numa fila de espera. O quadro 3.1 a seguir apresenta um resumo conceitual dos elementos de redes de filas aplicados a sistemas de manufatura.

Elemento	Descrição
Processo de Chegada	Intervalo de tempo entre chegadas, que no caso de ser probabilístico pode depender dos outros intervalos de tempo entre chegadas e /ou do processo de serviço ou consistir em intervalos entre chegadas independentes e identicamente distribuídos (idd).
Processo de Serviço	Tempo de processamento que também pode ser determinístico ou probabilístico. No caso de ser probabilístico, ele pode depender de outros tempos de processamento e/ou do processo de chegada, ou consistir de i.d.d (independente e identicamente distribuída).
Estações	Podem ter uma única máquina (servidor) ou várias máquinas. Uma máquina pode executar uma operação em *job* individual, ou em lotes de *jobs.* Cada máquina pode representar um conjunto de recursos como: operadores, ferramentas, etc.
Fila de Espera	Pode ter capacidade limitada ou ilimitada para o número de *jobs* na fila. A fila tem uma disciplina ou regra para ordenar os *Jobs* esperando por processamento. Exemplo: primeiro a chegar primeiro a ser servido, fila com prioridades, primeiro o de menor tempo de processamento etc.
Tipo da Fila = Aberta	Os *jobs* entram na rede, recebem processamento e um ou mais nós e eventualmente saem da rede. O número de *jobs* fluindo entre os nós é uma variável aleatória.

Quadro 3.1: Elementos de redes de filas aplicados a sistemas de manufatura
Fonte: Adaptado de BITRAN & MORABITO (1995).

A figura 3.2 ilustra bem o sistema dinâmico cujos elementos foram descritos no quadro 3.1.

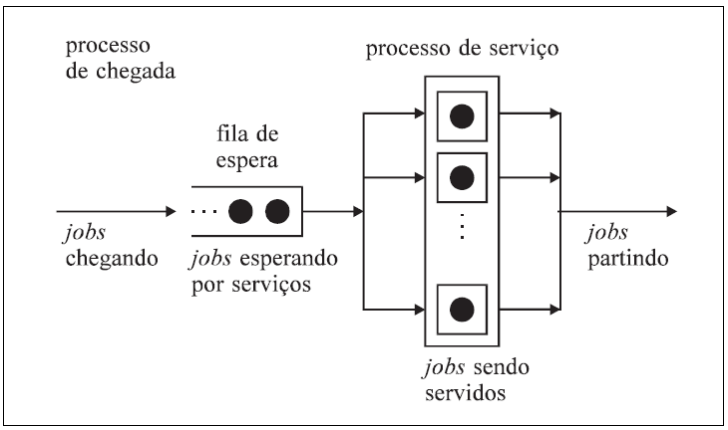

Figura 3.2: Uma Estação com Máquinas Idênticas e Fila Única.
Fonte: BITRAN & MORABITO (1995).

Embora o problema da formação das filas seja comum tanto em situações relacionadas ao trânsito quanto a problemas das linhas de produção, tem sido possível observar outros casos em que essa analogia também se aplica. Helbing (2003) apresenta informações importantes em que relaciona conceitos das redes de abastecimentos das linhas de produção com tráfego de veículos. Em seu trabalho, o autor afirma que recentemente outros trabalhos desenvolvidos por economistas, cientistas de tráfego, matemáticos e físicos tem salientado que métodos utilizados para o estudo da dinâmica de tráfego são também de uso potencial para o estudo das redes de abastecimento.

O trabalho de Helbing (2003) apresenta uma relação matemática entre o Efeito Chicote, frequente em sistemas de abastecimento e o Páre e Siga dos sistemas de tráfego, mostrando que a analogia entre a cadeia de suprimentos e modelos de tráfego diz respeito a sua estrutura matemática, porém ressalta que a interpretação deve ser diferenciada e que esta relação pode dar dicas de como métodos utilizados com sucesso na investigação de modelos de tráfego, pode ser generalizada para o estudo de redes de abastecimento.

Além disso, Helbing (2003) utiliza como analogia o trânsito de pedestres e linha de produção de semicondutores e ao concluir o trabalho o autor afirma que a razão para as semelhanças entre os sistemas de produção e de trânsito é a presença de entidades dinâmicas (pessoas, objetos), que interagem de um modo não linear, e a existência da competição por recursos limitados (como capacidade, tempo ou espaço).

Nagatani e Helbing (2008) apresentam um modelo matemático para cadeia de suprimentos envolvendo políticas de ordens de entrega com base em níveis de estoque e o relaciona com modelo macroscópico de trânsito.

É interessante notar que embora Bitran e Morabito (1995) tenham realizado seu trabalho utilizando métodos exatos e aproximados, os mesmos ressaltam que é possível utilizar a simulação e outras técnicas relacionadas para tal, sendo esta a abordagem que permite maiores considerações das situações reais. Muitas das características dos elementos destacados na tabela 1 apresentam relação com elementos dos modelos de Simulação a Eventos Discretos *(Discrete Event Simulation* - DES).

Ingalls (2008) diz que a Simulação a Eventos Discretos é o processo de concepção de um modelo dinâmico de um sistema real também dinâmico com a finalidade de compreender o comportamento do sistema ou de avaliar várias estratégias para a operação do mesmo.

De acordo com White Jr e Ingalls (2009), existe uma estrutura básica de componentes na simulação de eventos discretos, que estão compilados conforme Quadro 3.2.

Elemento	Descrição
Entradas, saídas e estados	Entradas são ações que provocam alterações no estado do sistema e refletem nas saída(s).
Entidades e Atributos	São as entidades dinâmicas que percorrem o fluxo do sistema (realizam as entradas no sistema). Atributos são características de uma determinada entidade que são exclusivas a mesmas.
Atividades e Eventos	Atividades são processos e lógica na simulação. Os eventos são condições que ocorrem em um ponto no tempo que provocam uma mudança no estado do sistema. Uma entidade interage com atividades para criar eventos. Em simulação existem três tipos principais de eventos: atrasos, filas e lógica.
Recursos	Representam alguma coisa em simulação que possui uma capacidade restrita. Exemplos: trabalhadores, máquinas, cruzamentos de trânsito.
Variáveis Globais	Os valores de uma variável global estão disponíveis para toda a simulação em todos os momentos e pode rastrear qualquer coisa de interesse
Gerador de Números Aleatórios	É uma rotina de software que gera um número aleatório entre 0 e 1. Este número é nas distribuições aleatórias.
Relógio e Calendário	O relógio é uma variável global que carrega o valor do tempo atual na simulação. O calendário é uma lista de eventos que estão programados para ocorrer no futuro, ou seja, em momentos de clock mais tarde do que a hora atual.
Coletor de Estatísticas	É uma parte da simulação que recolhe estatísticas sobre as condições (como o número de unidades da capacidade de um recurso em uso), ou o valor de variáveis globais, ou estatísticas de desempenho determinados com base em atributos da entidade.

Quadro 3.2: Estrutura básica dos componentes na Simulação de Eventos Discretos. Fonte: Adaptado de WHITE JR & INGALLS (2009).

É comum em sistemas de trânsito a utilização de simuladores que segundo Moita e Almeida (2012) auxiliam na identificação de fatores que influenciam negativamente o tráfego de veículos que transitam em determinado local, possibilitando a avaliação da necessidade de uma grande intervenção de natureza física ou apenas uma intervenção operacional e acrescentam que com surgimento

do computador na década de 50, a modelagem de filas pôde ser analisada pelo ângulo da simulação, onde se procurou imitar o funcionamento do sistema real.

Souza e Ribeiro (2004) ao analisarem os impactos causados no tráfego por alterações na rede viária utilizando micro simulação definiram que neste caso, os veículos são introduzidos na rede viária de forma aleatória, oriundos de um nó de entrada, trafegam pelos *links* existentes e, finalmente são absorvidos pelo nó de destino.

Baptista e Rangel (2011) afirmam que em uma via controlada por semáforo um dos fatores que mais influencia a fluidez do tráfego veículos, além dos acidentes e dos reparos na via, é o mau sincronismo entre os temporizadores dos semáforos, fazendo com que o congestionamento ocorra principalmente, nos pontos críticos, onde há interseção entre vias cujos semáforos encontram-se desregulados ou dessincronizados. Neste caso, a simulação torna-se um instrumento valioso na modelagem computacional deste sistema real sem que o mesmo sofra qualquer tipo de perturbação, pois os estudos e experiências são realizados no computador.

Moita e Almeida (2012) relatam que o uso da simulação como ferramenta tem contribuído para aperfeiçoar processos, já que possibilita prever as consequências que as mudanças na estrutura trarão a um determinado sistema.

3.3.A METODOLOGIA DA SIMULAÇÃO

A metodologia da simulação apresentada por Banks *et al.* (2010), Chwif e Medina (2010) chama atenção para algumas etapas que devem ser seguidas. A figura 3.3 apresenta o caminho a ser percorrido na simulação, que envolve uma primeira fase denominada “concepção”, que está relacionada a necessidade de se entender claramente o sistema a ser simulado e seus objetivos, bem como, ser realizada a coleta de dados de entrada para alimentação do modelo.

Na fase seguinte, a “implementação”, o modelo conceitual, produto da fase anterior, é convertido em um modelo computacional por meio da utilização de alguma linguagem de programação para uso geral (C, Pascal, *etc*) ou de um simulador. Neste momento também deve ser realizada a verificação e validação do modelo. A última fase é a “análise”, em que o modelo computacional está pronto para a realização dos experimentos.

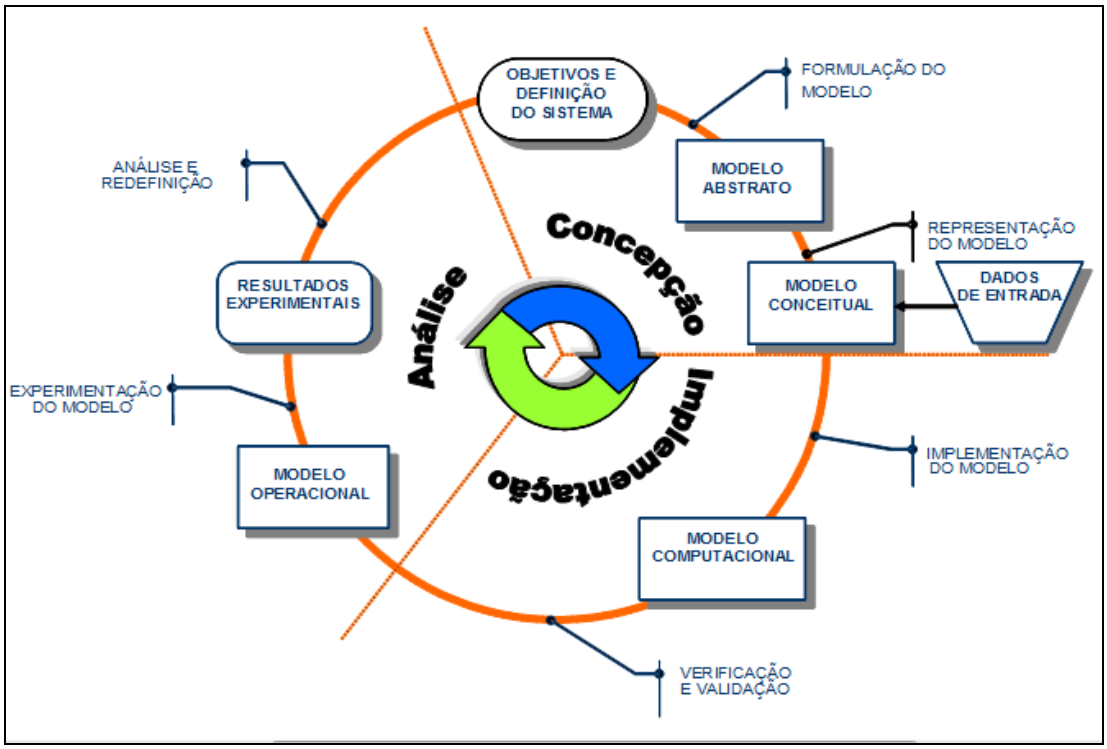

Figura 3.3: Metodologia de Simulação.
Fonte: CHWIF e MEDINA (2010)

Chwif e Medina (2010) lembram que as etapas apresentadas na Figura 3.3 não devem representar uma sequencia linear, mas um conjunto de iterações possibilitando diversas realimentações no processo à medida que o entendimento do problema for sendo alterado.

3.3.1.Fase de Concepção

3.3.1.1.Modelagem dos dados de entrada

Para Chwif e Medina (2010) uma das etapas mais importantes na construção de um estudo de simulação é a modelagem dos dados de entrada. O estudo da modelagem de dados pode ser resumido em três etapas: coleta de dados, tratamento de dados e Inferência. É nesta última etapa que será possível obter modelos probabilísticos que permitirão inferir as propriedades de um dado fenômeno aleatório.

LAW (2012) afirma que quase todos os sistemas do mundo real contêm uma ou mais fontes de aleatoriedade e que para realizar uma simulação usando entradas aleatórias temos que especificar suas distribuições de probabilidade.

De acordo com Chwif e Medina (2010) a modelagem de dados é facilitada quando as seguintes condições são válidas:

i) o processo de entrada de dados pode ser representado por uma sequência de variáveis aleatórias independente e identicamente distribuída (i.i.d), isto é, todas as variáveis pertencentes à sequencia têm a mesma distribuição de probabilidades e são mutuamente independentes entre si;

ii) a distribuição das variáveis aleatórias pode ser aproximada por um modelo probabilístico conhecido;

iii) os dados estão disponíveis de modo que seus parâmetros possam ser estimados.

Há uma série de armadilhas que podem minar o sucesso de um estudo de simulação, dentre os quais Law (2012) cita duas: i) substituição de uma distribuição de probabilidade de entrada por sua média; ii) utilização de uma distribuição errada.

Diante do exposto, são apresentados dois métodos que auxiliam a encontrar a melhor representação para os dados randômicos: a) buscar dentre as distribuições teóricas padrão (por exemplo, exponencial, lognormal ou Weibull) aquela que mais se adere ao conjunto de dados; b) utilizar uma distribuição empírica construída a partir dos dados (este método também possui desvantagens).

3.3.1.2.Criação do modelo conceitual

Robinson (2006) define modelo conceitual como sendo a abstração de um modelo de sistema real ou proposto.

Leal *et. al.* (2008) diz que sistema a ser simulado é analisado pelo analista de simulação na fase inicial de um projeto de simulação. O analista registra-o em sua mente de forma abstrata, e após deve registrar na forma de um modelo conceitual.

Após a elaboração do modelo abstrato em nossa mente, devemos coloca-lo no papel através de alguma técnica adequada de representação de modelos de simulação conceitual (CHWIF e MEDINA, 2010).

Para Montevechi, *et.al.* (2010) diversas técnicas de modelagem de processos tem sido utilizadas em projetos de simulação. Muitas dessas técnicas não foram desenvolvidas utilizando a mesma lógica utilizada em modelos de simulação, fato que contribui para que não haja um suporte adequado à programação.

A técnica IDEF-SIM é baseada em uma linguagem de modelagem que utiliza símbolos para representar elementos dos modelos de simulação, cuja principal característica é a semelhança da sua lógica de aplicação com a lógica utilizada na simulação a eventos discretos.

3.3.1.3.Fase da Implementação

3.3.1.3.1.Modelo de Simulação

Chwif e Medina (2010) afirmam que a etapa de implementação do modelo computacional nem sempre é fácil e direta, pois depende dos conhecimentos do analista em relação ao *software* de simulação utilizado.

3.3.1.3.2.Verificação e Validação

De acordo com Sargent (2013) tanto os desenvolvedores, como usuários e gestores que utilizam modelos de simulação para auxiliar na tomada de decisão, bem como as pessoas que são afetadas por essas decisões, preocupam-se com o fato de que o modelo de simulação e seus resultados estejam corretos.

Segundo Chwif e Medina (2010) para que um estudo de simulação seja bem sucedido é fundamental o processo de Validação e Verificação (V&V), que embora na metodologia descrita pela figura 1 esteja logo após a construção do modelo computacional, na realidade este processo deve acompanhar todo o ciclo de vida do projeto, sendo portanto um processo contínuo.

A terminologia na área de Verificação e Validação (V&V) não é padrão. Klejnen (1995) utiliza as seguintes definições: a verificação determina se o programa simulador executa conforme o esperado, e a validação está preocupada em determinar se o modelo de simulação conceitual é uma representação exata do sistema em estudo.

Chwif e Medina (2010) também definem verificação relacionando-o ao modelo computacional, em que seria possível realizar a pergunta: “será que estamos desenvolvendo corretamente o modelo?”. Já a validação está relacionada ao modelo conceitual, em que a pergunta que pode ser feita é a seguinte: “será que estamos desenvolvendo o modelo correto?”.

Sargent (2013) afirma que há quatro abordagens básicas para avaliar se um modelo de simulação é válido, sendo que cada uma das abordagens requer que a equipe de desenvolvimento realize a verificação e validação como parte do processo de desenvolvimento do modelo. Em linhas gerais, as abordagens descritas por Sargent (2013) são as relatadas a seguir:

- Decisão subjetiva da equipe de desenvolvimento realizada com base nos resultados de vários testes e avaliações;
- Participação do usuário do modelo na equipe de desenvolvimento de forma que o mesmo possa validá-lo aumentando assim a sua credibilidade;
- Verificação e validação independente, em que nem a equipe de desenvolvimento, nem o usuário participam diretamente do processo de verificação e validação. Esta abordagem é normalmente empregada em desenvolvimento de modelos de simulação em grande escala, cujo desenvolvimento geralmente envolve várias equipes;
- Utilização de um modelo de pontuação. Pontos ou pesos são determinados subjetivamente ao realizar vários aspéctos do processo de validação e então são combinados para determinar a pontuação da categoria e a pontuação global. Esta abordagem é raramente utilizada na prática. O autor afirma que não acredita neste tipo de abordagem citando como justificativa para tal entre outras coisas, o alto grau de subjetividade empregada.

Há algumas técnicas ou procedimentos que podem ser utilizadas para facilitar o processo de verificação (CHWIF e MEDINA (2010), KLEIJNEN (1995)):

- Implementação modular / verificação modular: Esta técnica está relacionada com a implementação do modelo em partes e após a execução somente desta parte. Se a parte testada estiver correta,

então é possível passar para a parte seguinte, e assim sucessivamente;

- Verificação intermediária das saídas ou Simulação manual: Permite que o analista tenha um conhecimento prévio do comportamento do modelo, de forma a ter uma maior percepção sobre a correspondência entre o modelo computacional e o modelo conceitual. Essa técnica torna-se inviável para modelos muito grandes;
- Comparação dos resultados finais da simulação utilizando valores constantes ou simplificados: Em uma simulação pode-se utilizar um valor médio que represente todas as distribuições de probabilidades envolvidas no modelo e utilizar este valor médio como uma constante determinística, realizando assim uma simulação determinística. Embora os resultados não sejam os reais, neste caso é possível comparar os resultados do modelo com os resultados de uma planilha de cálculo e observar se os valores estão corretos;
- Animação gráfica: A animação gráfica permite que seja possível visualizar a dinâmica do modelo que representa o mundo real. Em casos em que ocorre alguma alteração dessa representação é possível detectar erros;
- Revisão em grupo: É uma técnica em que uma pessoa ou um grupo de pessoas verifica se o modelo está funcionando adequadamente (CHWIF e MEDINA, 2010).

Em relação a validação do modelo, Sargent (2013) recomenda que ao menos as 8 etapas a seguir sejam realizadas:

i) Acordo entre todas as partes envolvidas (desenvolvedores, clientes e usuários), de forma a especifiar a abordagem básica e um conjunto mínimo de técnicas de validação que serão utilizados no processo de validação;

ii) Especificar a precisão das variáveis de saída do modelo de simulação o quanto antes;

iii) Testar sempre que possível os pressupostos e teorias subjacentes ao modelo de simulação;

iv) Em cada iteração do modelo realizar a validação com o modelo conceitual;

v) Em cada iteração do modelo estudar o comportamento do modelo computadorizado;

vi) Na última iteração, realizar comparaçãos entre o modelo de simulação e o comportamento dos dados de saída do sistema para pelo menos alguns conjutos de condições experimentais e preferencialmente para vários conjuntos;

vii) Desenvolver a documentação de validação para inclusão na documentação do modelo de simulação;

viii) No caso do modelo de simulaçao ser utilizado por um período de tempo, deve ser criado um cronograma para revisão periódica da validade do modelo.

3.4.METODOLOGIA PARA SIMULAÇÃO COM INTELIGÊNCIA COMPUTACIONAL

Aperfeiçoando o trabalho de Silva *et al.* (2012), o objetivo deste trabalho foi incorporar ao Ururau um módulo capaz de executar uma RNA diretamente no modelo de simulação, sem a necessidade de comunicação externa, e assim, acrescentar esta nova funcionalidade diretamente ao seu código.

Desta forma, tornou-se possível representar partes de processos modelados, onde a decisão é tomada com base em fatores diferentes dos testes lógicos ou porcentagens, que tenham haver com conhecimentos e experiências prévias do decisor.

Para que não fosse necessário “reinventar a roda”, foram pesquisados e avaliados *frameworks* para aplicação de RNA em modelos de SED. Isso permitiu a realização de um estudo mais aprofundado sobre recursos e tecnologias já consolidadas.

A princípio o que se buscou ao realizar a pesquisa, foi encontrar algum *framework* em Java puro que permitisse a criação, treinamento e execução de uma RNA. A necessidade de ser um *framework* Java se deu ao fato do Ururau ter sido desenvolvido originalmente nesta linguagem.

Ao buscar o *framework* mais adequado ao projeto, os seguintes fatores foram considerados como tendo grande importância:

a) Interoperabilidade – capacidade de funcionar em vários sistemas operacionais;

b) *Framework* Java – compatibilidade com o código do Ururau existente ;
c) Possuir código fonte aberto – possibilidade de expansão e melhorias; uso acadêmico;
d) Ser gratuito – uso acadêmico;
e) Possuir facilidade de acesso a informações, atualizações, comunidade engajada – indicativo de melhorias, atualizações, expansão; crescimento; uso acadêmico.

A incorporação do módulo inteligente ao Ururau possibilita, entre outras coisas, que a RNA tenha seus parâmetros configurados no momento da criação do modelo, e pela GUI.

Observe ainda, que o usuário também tem a opção de poder manipular a RNA nas camadas do núcleo, por meio de linhas de código em Java.

Com base nos critérios preestabelecidos, e nas pesquisas realizadas, o *Encog* foi o *framework* escolhido para o desenvolvimento do módulo inteligente no Ururau.

A etapa seguinte à escolha do *framework,* foi a realização de testes para saber se a ferramenta seria realmente adequada. Para tal, foram realizados testes na ferramenta, e inicialmente, duas principais perguntas foram feitas:

- O *Encog* funciona sem falhas ou inconsistências junto ao Ururau?
- O módulo implementado com o *Encog* está gerando os resultados corretos?

Para responder a primeira pergunta, foi realizado um teste, chamado teste de acoplamento. O objetivo deste teste foi identificar possíveis falhas e conflitos entre o código existente do Ururau, e o código do *framework Encog.* Nesta etapa também foi possível fazer pequenos ajustes nos códigos, e verificar se o Ururau continuaria “rodando” sem a ocorrência de problemas de incompatibilidades. A princípio, o objetivo foi que os resultados obtidos no teste fossem analisados a partir de um modelo bem simples.

A etapa seguinte foi chamada de teste de funcionamento, que tem relação com a segunda pergunta. Neste caso, não bastaria que os códigos do Ururau e do *Encog,* já unificados, não gerassem conflitos, mas que os resultados obtidos durante a execução de modelos de simulação utilizando RNA, gerassem os resultados

corretos. Para isso, foi importante partir de algum modelo já validado e realizar comparações em relação aos resultados obtidos.

Os próximos subitens descrevem os passos dos testes descritos.

3.4.1.Construção do Modelo Computacional

A proposta deste trabalho é acoplar o módulo contendo a RNA a toda estrutura do Ururau, permitindo que já na camada de GUI, o modelador possa configurar uma RNA como um componente de decisão do ambiente de simulação, conforme pode ser visto na figura 3.4.

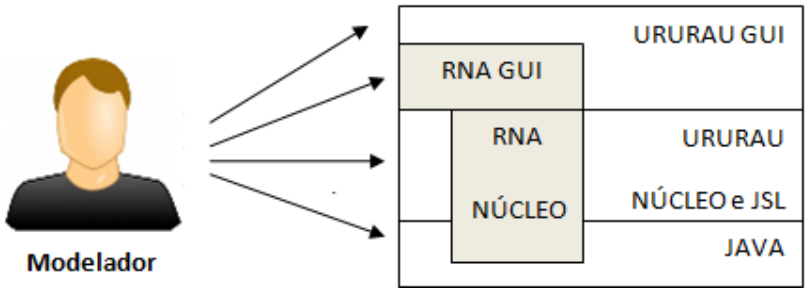

Figura 3.4: Módulo RNA acoplado ao URURAU.
Fonte: Elaboração própria.

A incorporação do módulo inteligente ao Ururau torna desnecessária a comunicação externa por meio de comunicação TCP, conforme foi descrito por Silva *et al.* 2012. Desta forma, representação da arquitetura do trabalho desenvolvido pode ser observada pela figura 3.5.

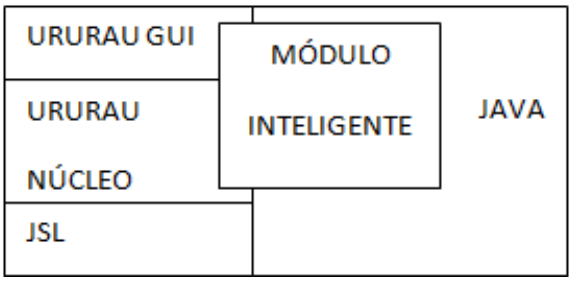

Figura 3.5: Adaptação do Ururau.
Fonte: Elaborado pelo autor

3.4.2.Frameworks de Redes Neurais Artificiais Java

Para o desenvolvimento do Módulo Inteligente, optou-se pela utilização de um *framework* Java para criação de RNAs. Desta forma, três dos principais *frameworks* não comerciais foram pesquisados e são destacados neste trabalho:

- JOONE *for Artificial Intelligence Programming* (JOONE, 2013);
- NEUROPH *Java Neural Network Framework* (NEUROPH, 2013);
- ENCOG *Neural Networks for Java* (ENCOG, 2013).

3.4.2.1.JOONE *for Artificial Intelligence Programming*

O Joone é um *framework* escrito em Java puro para construir e executar aplicações de Inteligência Artificial baseadas em Redes Neurais. Pode ser executado em qualquer plataforma e é compatível com vários sistemas operacionais como: Linux, Mac OSX, Windows 2000, Windows XP, SUN Solaris, MARRONE (2007).

De acordo com Marrone (2007) esta ferramenta está licenciada sob a LGPL (*GNU Lesser General Public License*), desenvolvida pela comunidade Joone, e pode ser utilizada tanto por entusiastas quanto por usuários profissionais.

Ele é composto por um motor central, um editor de GUI e um ambiente de treinamento distribuído. O Joone é aberto para que novos módulos ou novas arquiteturas sejam criadas a partir de seus componentes de base.

Marrone (2007) afirma que o Joone possibilita que RNAs sejam criadas em uma máquina local, treinadas em ambiente distribuído e executado em qualquer dispositivo. O mesmo é desenvolvido em componentes que possuem algumas características básicas, tais como, persistência, *multithreading*, serialização e parametrização, o que garante a escalabilidade, confiabilidade e expansibilidade.

A figura 3.6 apresenta o exemplo de uma GUI do Joone com dados de uma RNA XOR.

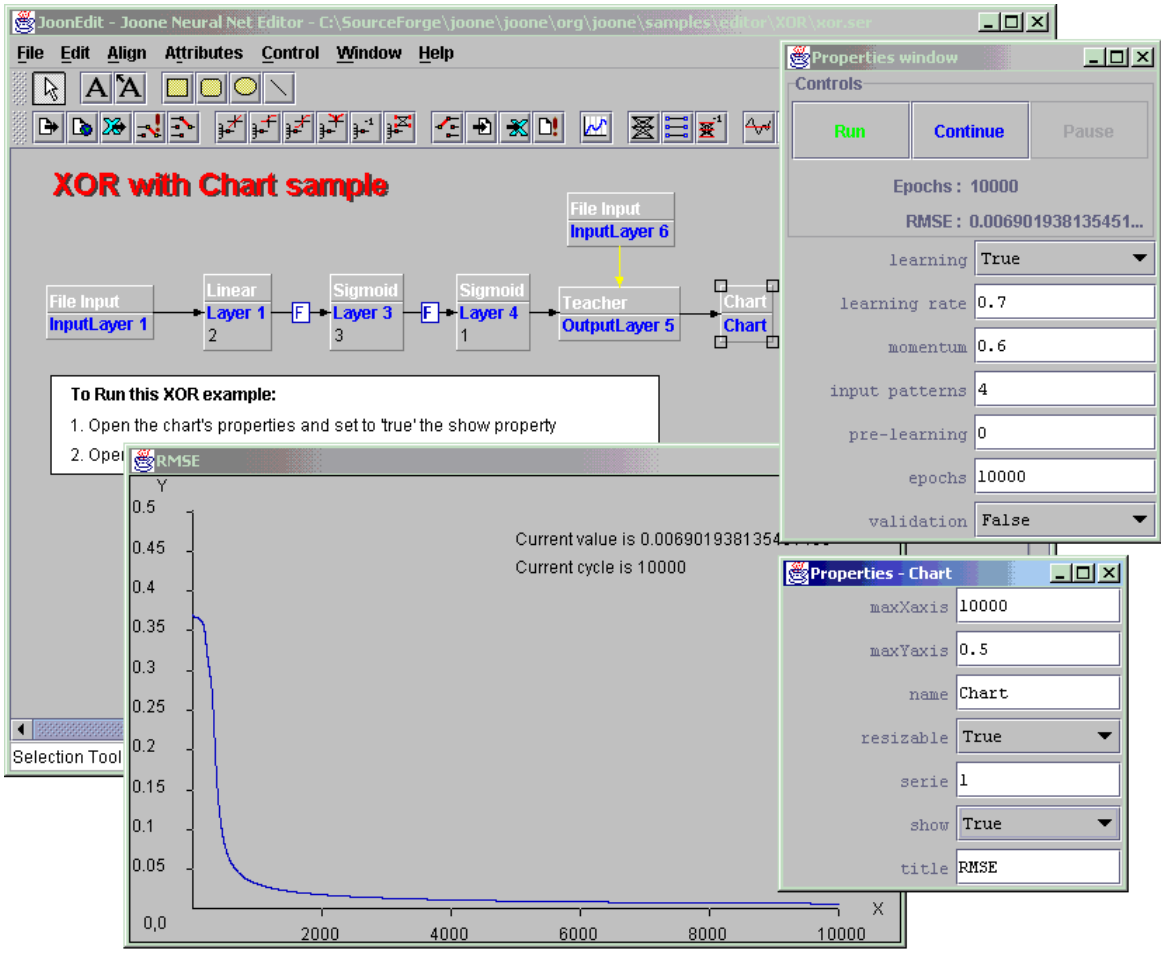

Figura 3.6: Exemplo da Interface Gráfica do *Framework* Joone.
Fonte: JOONE (2013).

3.4.2.2.Neuroph *Java Neural Network Framework*

O Neuroph é um *framework* para desenvolvimento de RNAs que consiste em uma biblioteca Java e um editor de redes neurais chamado *Neuroph Studio* (SEVARAC & KOPRIVICA 2013).

É um projeto *open source* hospedado no repositório *SourceForge* e desde a versão 2.4 é licenciado sob Apache 2.0. Versões anteriores estão sob a licença LGPL3 (SOURCEFORGE, 2013).

A figura 3.7 representa o exemplo de uma tela no *Neuroph Studio* para a criação de uma RNA.

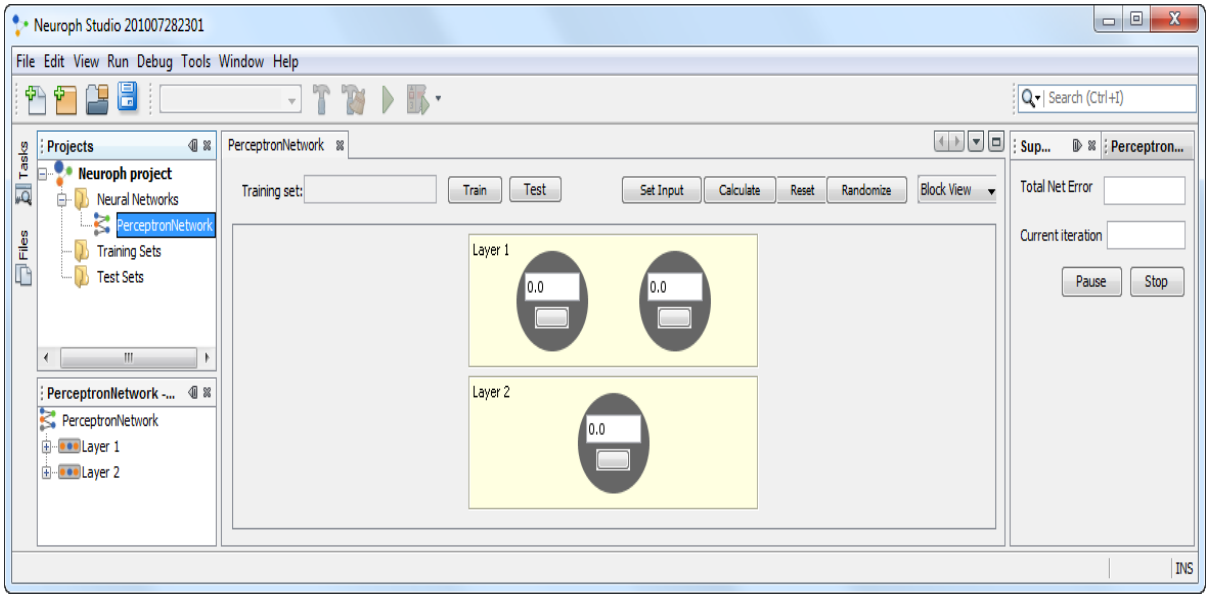

Figura 3.7: Exemplo da Interface Gráfica do Framework Neuroph.
Fonte: SOURCEFORGE(2013).

3.4.2.3.Encog Neural Networks for Java

O Encog é um framework de Inteligência Artificial Java e .Net. Inicialmente foi criado para suportar somente RNA, contudo com o tempo foi se expandindo para trabalhar com aprendizagem de máquina em geral (HEATON, 2011). De acordo com Heaton (2011), o Encog faz uso de diversas bibliotecas de terceiros para adicionar funcionalidades necessárias, sem “reinventar a roda”.

É completamente gratuito e aberto, sendo desenvolvido por uma comunidade de programadores, tendo como líder Jeff Heaton. É multi plataforma, funcionando em máquinas Windows, MAC e Linux (HEATON, 2013).

Segundo Heaton (2014) o Encog está sob a licença Apache License 2.0, que é muito similar à licença LGPL (*Lesser General Public License*).

O Encog possui uma aplicação gráfica conhecida como *Encog Workbench* que permite realizar diversas tarefas de aprendizagem de máquina, sem que seja necessário escrever código Java ou C#. Ele é escrito em Java, mas gera arquivos que podem ser usados com qualquer *framework* Encog.

A figura 3.8 apresenta uma tela da aplicação gráfica *Encog Workbench.*

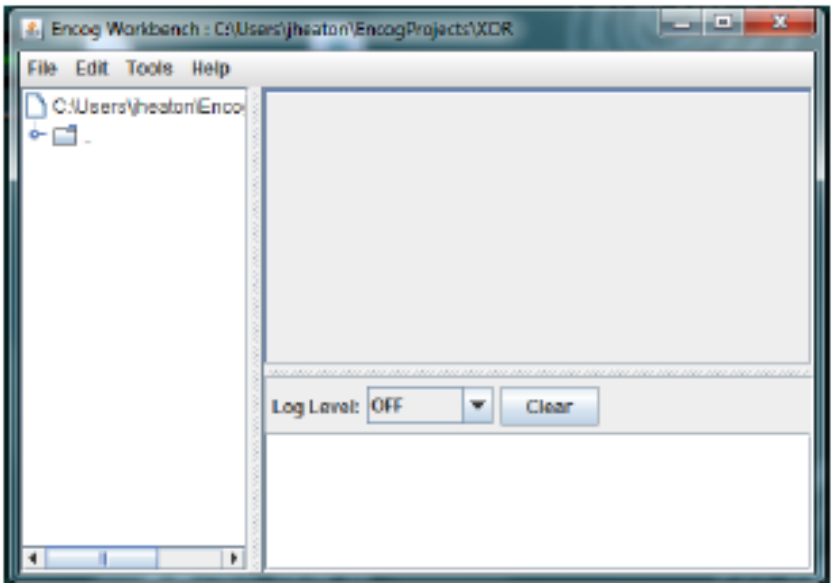

Figura 3.8 Exemplo do Encog Workbench.
Fonte: (HEATON, 2011).

3.4.3.Comparação dos *Frameworks* de RNAs Java

Baptista & Dias (2012) afirmam que as informações contidas em artigos e sites que tratam das ferramentas de RNA são suficientes para que se possa tomar uma decisão a respeito daquela que melhor se encaixa ao problema a ser resolvido. Para uma escolha mais criteriosa de uma ferramenta RNA Java, foi realizada uma pesquisa visando à determinação de alguns critérios para comparação das mesmas.

Matviykiv & Faitas (2013) realizaram uma análise com o objetivo de escolher a melhor ferramenta para o desenvolvimento de uma RNA para classificação espectral. Os autores testaram o Encog 3.1, o Joone 2 RC1, o Neuroph 2.6 e também a ferramenta FANN 2.2, que não é objeto deste estudo. Ao final da análise, foi possível concluir que o Encog apresentou melhores resultados e maior facilidade de utilização.

Baptista & Dias (2013) apresentaram dados sobre um grande número de ferramentas que podem ser utilizadas para criação de RNA. O trabalho desenvolvido pelos autores levou em consideração os seguintes aspectos: Sistema Operacional, Requisitos mínimos de *hardware* e *software*, Licença, Arquitetura da rede e Algoritmo de treinamento.

O quadro 3.3 traz a relação das arquiteturas de redes que podem ser desenvolvidas por cada *framework.* É perceptível a superioridade do Encog neste

quesito, uma vez que possibilita a criação de RNAs com diversas arquiteturas diferentes, contribuindo para uma maior aplicabilidade nos mais variados tipos de problemas.

Frameworks	Arquitetura de Rede
Encog	*Adaline Linear Neuron; Adaptive Resonance Theory; Bidirectional Associative Memory; Boltzmann Machine; Counter-Propagation Neural Network; Elman Network; Hopfield Network; Jordan Recurrent; Kohonen Networks; Multilayer Perceptron; Neuro evolution of Augmenting Topologies; Radial Basis Function; Self-Organizing Map.*
Joone	*Feed-Forward Neural Network; Kohonen Networks; Modular Neural Network; Principal Component Analysis; Time-Delay Neural Network*
Neuroph	*Adaline Linear Neuron; Bidirectional Associative Memory; Competitive Neural Network; Hebbian Network; Hopfield Network; Kohonen Networks; Maxnet; Multilayer Perceptron; Radial Basis Function.*

Quadro 3.3: Arquitetura de Rede

Fonte: Adaptado de Baptista & Dias (2013).

Baptista & Dias (2013) afirmam que a arquitetura de uma RNA está relacionada com a estrutura e o tipo da rede, e que é uma das decisões mais importantes no momento da escolha da ferramenta. Algumas redes são mais apropriadas para alguns tipos de tarefas do que outros, e desta forma, cada ferramenta abrange apenas um subconjunto de todas as arquiteturas disponíveis.

Outro aspecto importante e destacado por Baptista & Dias (2013) são os algoritmos de treinamento. O Quadro 3.4, apresenta os resultados encontrados pelos autores em relação as ferramentas em estudo.

Frameworks	Algoritmos de Treinamento
Encog	*Backpropagation; Conjugate Gradient; Competitive Learning; Genetic Algorithm; Levenberg–Marquardt.*
Joone	*Backpropagation; Batch Training; Conjugate Gradient Descent; Delta-bar-Delta; Resilient Propagation.*
Neuroph	*Backpropagation; Competitive Learning.*

Quadro 3.4: Algoritmo de Treinamento.

Fonte: Adaptado de Baptista & Dias (2013).

O Encog e o Neuroph apresentaram a mesma quantidade de algoritmos de treinamento, embora com diferenças entre ambos. Somente o algoritmo

Backpropagation pode ser utilizado por todos os *frameworks.* Embora não esteja listado no trabalho de Baptista & Dias (2013), o algoritmo de treinamento *Resilient Propagation*, também está disponível no Encog 3, de acordo com informações apresentadas por Heaton (2010).

O Codeproject (2010) realizou uma comparação entre o Neuroph, o Encog e o Joone. Para tal, foi criada uma RNA *feedforward* para reconhecer uma operação XOR.

As versões testadas e comparadas de cada *framework* são as seguintes: Encog v2.4, Neuroph v2.4 e Joone v2 RC1. O computador utilizado foi um Dell Studio XPS 8000, com processador Intel Core i7 860@ 2.8ghtz. O processador é *quadcore* com *hyperthreading.*

Segundo o autor, a ideia da realização dos testes se baseou na observação de que o *framework* Joone apresenta uma alta complexidade no código fonte. O objetivo é verificar se outro *framework* é capaz de realizar ciclos de treinamento de forma mais rápida fazendo uso do recurso *multicore* das máquinas modernas.

O Quadro 3.5 apresenta algumas informações importantes sobre a RNA criada para os testes:

Dados da RNA	
Neurônios de Entrada	10
Neurônios de Saída	10
Neurônios da Camada Oculta	20
Função	TANH
Conjunto de Treinamento	100 elementos
Iteração	50
Método de Treinamento	*backpropagation* com momento

Quadro 3.5: Dados de criação da RNA.

Fonte: Codeproject (2010 a)

A figura 3.9 apresenta a comparação do tempo total de processamento das RNA criadas utilizando cada um dos *frameworks*. O Encog e o Joone oferecem recursos *multithread,* que possibilita o processamento paralelo beneficiado, por exemplo, pela tecnologia *multicore* presente nos computadores atuais. O Neuroph não apresenta essa funcionalidade. O recurso *multithread* está representado pelas letras MT e ST quando não tiver o recurso.

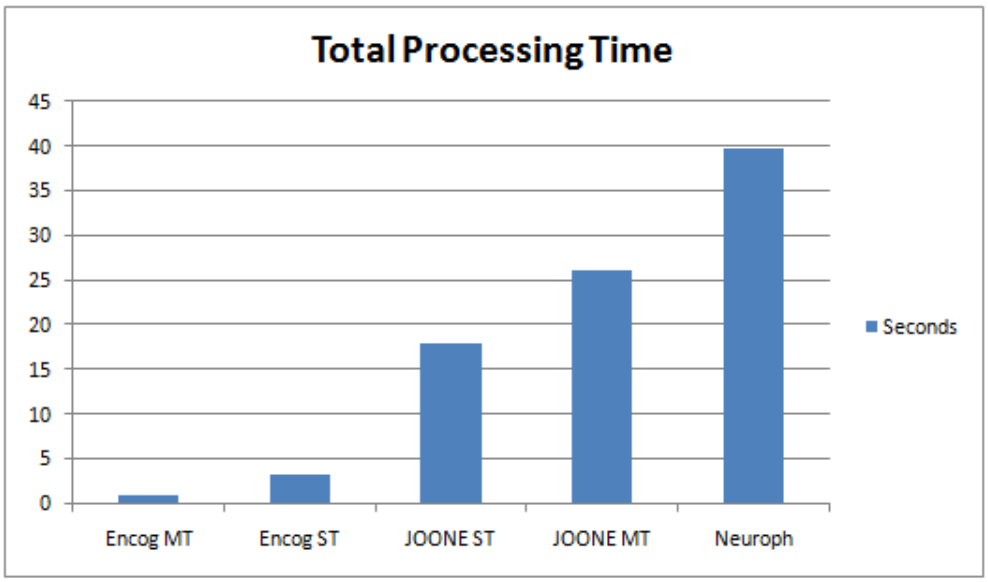

Figura 3.9: Tempo total de Processamento da RNA.
Fonte: Codeproject (2010a)

De acordo com o resultado dos testes realizados por Codeproject (2010a) e apresentados na figura 3.9, é possível notar que o Encog teve os menores tempos de processamento, tanto em modo *multithread* quanto em modo *monothread.*

O Joone mostrou um desempenho inferior trabalhando em modo *multithread,* o que significa que com uma arquitetura desenvolvida utilizando múltiplas *threads,* o processamento da RNA pode levar mais tempo, mesmo sendo utilizada uma máquina *multicore.*

A RNA criada utilizando o Neuroph foi a que gastou maior tempo para ser processada.

O quadro 3.6 apresenta um paralelo em que se observam os tipos de RNAs suportadas por cada *framework.* O Joone é a opção que apresenta uma menor capacidade de implementação dos diversos tipos de RNAs .

O Neuroph e o Encog são opções mais robustas neste sentido, contudo o Encog ainda consegue implementar 3 opções de RNAs a mais que o Neuroph (*Art1, Boltzmann Machine, Elman SRN, Recurrent SOM)*. Este por sua vez não é capaz de implementar apenas a RNA do tipo *NeuroFuzzy Perceptron.*

	JOONE	Neuroph	Encog
ADALINE		*	*
ART1			*
BAM		*	*
Boltzmann Machine			*
CPN		*	*
Elman SRN	*		*
Perceptron	*	*	*
Hopfield		*	*
Jordan SRN		*	*
NEAT		*	*
RBF		*	*
Recurrent SOM			*
Kohonen/SOM		*	*
NeuroFuzzy Perceptron		*	
Hebbian Network		*	*

Quadro 3.6: Tipos de Redes Neurais Artificiais.
Fonte: Codeproject (2010).

Em relação às funções de ativação que os *frameworks* possibilitam implementar, os três oferecem um bom número de opções, contudo mais uma vez o Encog apresenta uma maior variedade de funções disponíveis, não sendo capaz de implementar somente a função trapezoide, conforme pode ser visualizado no quadro 3.7.

	JOONE	Neuroph	Encog
Sigmoid	*	*	*
HTAN	*	*	*
Linear	*	*	*
SoftMax	*		*
Step	*	*	*
Bipolar/Sgn			*
Gaussian		*	*
Log	*	*	*
Sin	*		*
Logarithmic	*		*
Ramp		*	*
Trapezoid		*	

Quadro 3.7: Funções de Ativação.
Fonte: Codeproject (2010).

Quanto às técnicas responsáveis pela geração da aleatoriedade dos dados da RNA, o quadro 3.8 apresenta que o Encog permite implementar quatro delas

(*Range, Gaussian, Fan-In e Nguyen-Widrow*), enquanto o Neuroph somente 1 (*Range*) e o Joone 2 (*Range e Fan-In*).

	JOONE	Neuroph	Encog
Range	*	*	*
Gaussian			*
Fan-In	*		*
Nguyen-Widrow			*

Quadro 3.8: Técnicas de Aleatoriedade.
Fonte: Codeproject (2010).

Para o treinamento das RNA, são utilizadas técnicas que possuem características próprias e adequadas a tipos de problemas e situações diversas. O quadro 3.9 traz a relação dessas técnicas com os *frameworks* que foram testados por Codeproject (2010).

É possível notar que o Encog possibilita a implementação de praticamente todas as diferentes técnicas de treinamento listadas.

	JOONE	Neuroph	Encog
Annealing		*	*
Auto Backpropagation		*	*
Backpropagation	*	*	*
Binary Delta Rule		*	
Resilient Prop	*		*
Hebbian Learning		*	*
Scaled Conjugate Grd			*
Manhattan Update			*
Instar/Outstar		*	*
Kohonen	*	*	*
Hopfield		*	*
Levenberg Marquardt (LMA)			*
Genetic		*	*
Instar		*	*
Outstar		*	*
Adaline			*

Quadro 3.9: Técnicas de Aleatoriedade.
Fonte: Codeproject (2010).

A partir dos testes realizados, o Codeproject (2010) apresentou algumas conclusões cujas mais relevantes estão relatadas aqui.

No geral, o Encog e o Neuroph apresentaram melhores resultados que o Joone, porém o *framework* que apresentou superioridade na maioria dos testes foi o Encog.

Foram necessárias apenas 18 iterações para o treinamento da rede utilizando o Encog, enquanto o Neuroph realizou 613 iterações e o Joone mais de 5000 iterações.

Talvez essa diferença na quantidade de iterações em relação aos outros dois *frameworks*, esteja relacionado com o fato do Encog utilizar GPU *(Graphics Processing Unit,* ou Unidade de Processamento Gráfico*)* para aumentar sua velocidade de treinamento.

3.4.4.Decisão de Projeto

As conclusões obtidas com os trabalhos de Baptista & Dias (2013), Matviykiv & Faita (2013) e Codeproject (2010 a-b) possibilitaram observar que o *Encog* apresentou melhores resultados em relação à execução da RNA e, também, na maioria dos quesitos da comparação.

Durante as buscas e pesquisas realizadas, foi possível observar que o *Encog* possui uma grande gama de informações disponibilizadas sob forma de tutoriais, fóruns, listas de discussões e livros publicados, o que, em termos FOSS, garante uma maior perspectiva de continuidade, crescimento e desenvolvimento de novos recursos e melhorias da ferramenta.

Por meio deste estudo de pesquisa envolvendo os três *frameworks,* foi possível observar que o *Encog* se apresentou como o *framework* mais completo em termos de funcionalidades, e também o que possuía mais informações disponíveis, desta forma, optou-se por escolher o Encog para a realização dos testes no Ururau.

3.5.O SISTEMA DE TRÂNSITO EM ESTUDO

O sistema escolhido para estudo e aplicação dos experimentos, é composto por um cruzamento em que o controle é realizado por um guarda de trânsito.

Neste sistema, guarda é quem toma a decisão. Ele se baseia em diversas variáveis do trânsito, como quantidade de veículos em cada via, número de

pedestres aguardando para atravessar, tempo necessário para a fluidez do trânsito, dentre outros. A figura 3.10 apresenta uma visão aérea do cruzamento.

Figura 3.10: Visão Aérea do Cruzamento em estudo.
Fonte: Google Maps.

A figura 3.11 apresenta o sistema de trânsito real que foi utilizado. É importante perceber a existência de uma faixa de pedestres, que corta a via de maior fluxo, chamada neste trabalho de via principal. A via de menor fluxo é chamada aqui de via secundária.

Este cruzamento está localizado em uma área movimentada da cidade de Campos dos Goytacazes, RJ, próximo a áreas bastante populares, como o mercado municipal e o camelódromo.

Figura 3.11: Sistema de trânsito real utilizado no estudo

As figuras 3.12 e 3.13 mostram o momento em que o guarda atua como um controlador de trânsito.

Figura 3.12: Fechamento para veículos.

O guarda toma a decisão de fechar a via principal para o fluxo de veículos (figura 3.12), e abre a mesma para o fluxo de pedestres (figura 3.13).

Figura 3.13: Abertura para pedestres.

A figura 3.14 apresenta um grupo de pessoas aguardando o momento de fechamento do trânsito para fluxo de veículos na via principal, para realizarem a travessia.

Figura 3.14: Pedestres aguardando

A figura 3.15 mostra os pedestres atravessando a via sobre a faixa, com a presença do guarda de trânsito realizando o controle das vias.

Figura 3.15: Pedestres atravessando a via principal.

3.5.1.Modelo Conceitual do Sistema

Para construção do modelo foi seguida a metodologia apresentada por Chwif e Medina (2010), composta por três grandes etapas: concepção ou formulação do modelo, implementação, e análise dos resultados. Para tal, observaram-se as advertências descritas por Banks e Chwif (2010).

Além disso, para a verificação e validação, foi seguida adicionalmente a metodologia proposta por Sargent (2013). O modelo foi testado apenas após a checagem completa da validade do mesmo.

O desenho esquemático do cruzamento pode ser visualizado pela figura 3.16.

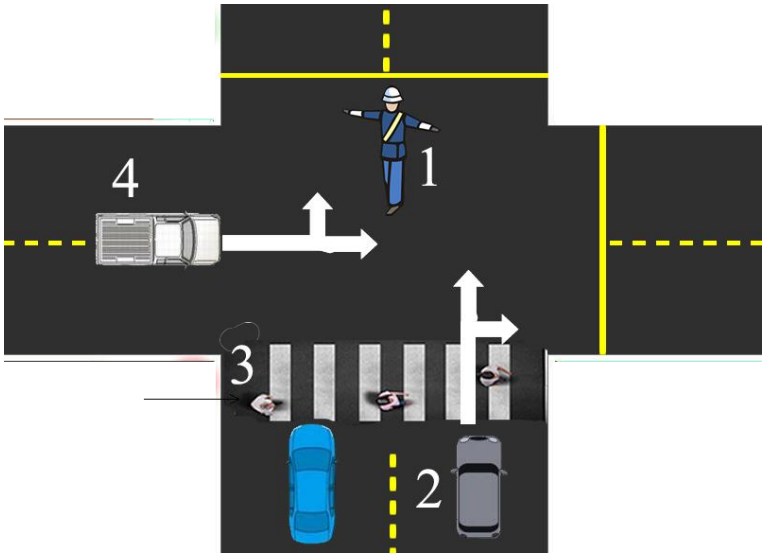

Figura 3.16: Esquema do sistema controlado por um guarda de trânsito

No sistema modelado são apresentados alguns elementos que podem ser destacados: (1) o guarda de trânsito que realiza o controle da via, (2) a via principal que recebe maior fluxo de veículos, e que possui a faixa para travessia de pedestres (3). A via secundária (4) corta a via principal e também possui um grande número de pessoas devido a proximidade com áreas comerciais populares da cidade.

4. RESULTADOS E DISCUSSÃO

4.1. TESTE DE ACOPLAMENTO DA RNA COM O *SOFTWARE* URURAU

O primeiro teste realizado com o Encog teve como objetivo verificar se o mesmo possui um bom acoplamento com o Ururau em termos de compatibilidade entre os códigos.

Para tal, o Encog foi incorporado ao núcleo do Ururau. Para essa incorporação entre códigos, foi utilizado o IDE (*Integrated Development Environment)* Netbeans 7.0.1 e a linguagem de programação Java.

Neste primeiro teste foi criada uma RNA para o problema XOR (ou exclusivo), conforme modelo conceitual apresentado na figura 4.1.

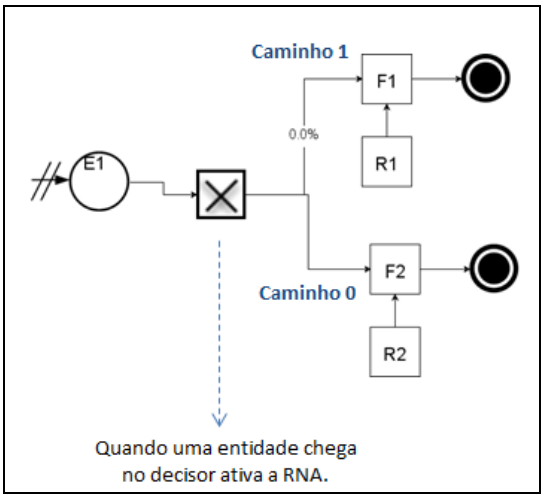

Figura 4.1: Modelo Conceitual com uma rede XOR no Ururau.

O referido modelo está representado em linguagem IDEF-SIM (Montevechi *et al*., 2010). É importante citar que o Ururau utiliza componentes gráficos semelhantes aos elementos desta linguagem. Essa característica torna a conversão do modelo conceitual em modelo de simulação, com um entendimento mais fácil e direto.

Para a realização do teste, foi utilizado um computador Intel *Core* i5 2.6Ghz, 4Gb de memória RAM e sistema operacional *Windows* 7.

No problema modelado, quando uma entidade encontra o operador de decisão, a RNA é ativada e a mesma consulta os valores dos neurônios de entrada.

A tabela verdade da função XOR funciona como demonstrada na Tabela 4.1:

Tabela 4.1: Tabela Verdade da Função XOR.

Neurônios de Entrada		Neurônio de Saída
N1	**N2**	**S**
0	0	0
0	1	1
1	0	1
1	1	0

A arquitetura da RNA criada neste teste é apresentada na figura 4.2, em que os neurônios de entrada obtêm dados resultantes de expressões processadas pelo Ururau, como por exemplo: tamanho da fila, tempo na fila, testes condicionais, entre outros.

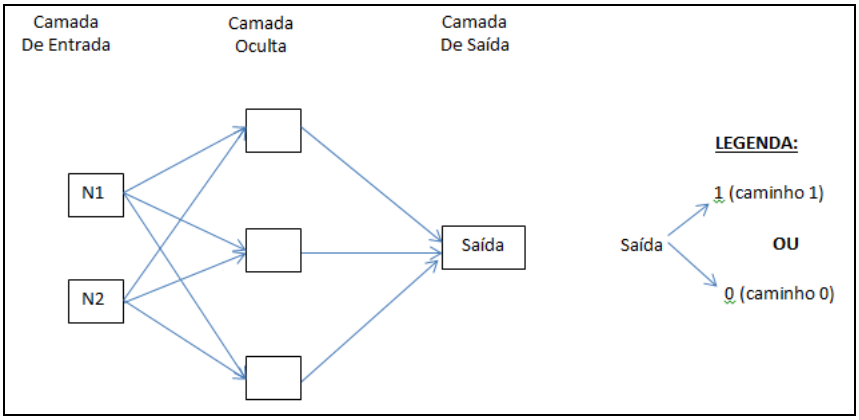

Figura 4.2: Exemplo da RNA XOR implementada no teste.

No exemplo representado pela figura 4.1, F1 representa a quantidade de entidades na fila do processo F1, e F2 representa a quantidade de entidades na fila do processo F2.

O caminho a ser percorrido no modelo pela entidade é definido segundo a decisão da própria RNA e não mais por critérios pré-definidos. Após a execução da RNA cada entidade segue seu percurso no modelo de acordo com a decisão tomada pela mesma.

Os neurônios da camada de entrada foram configurados de forma que receberão o valor binário 1, no caso de existirem entidades aguardando na fila dos processos F1 e F2, e receberão valor binário 0 no caso de não haver entidades aguardando na fila.

Por se tratar de um problema binário, convencionou-se que no caso do valor de saída da RNA ser 0, a entidade é encaminhada para o caminho 0 do modelo, e no caso da saída da RNA ser o valor 1, a entidade será encaminhada para o caminho 1 do modelo (figura 4.2).

Neste caso, supondo-se, por exemplo, que em um dado momento da simulação F1 seja igual a 1, ou seja, há uma entidade aguardando na fila F1, e F2 seja igual a 0, ou seja, não há entidades aguardando na fila F2, temos que:

N1= “F1>0”, resulta 1 na entrada do primeiro neurônio (N1);

N2= “F2>0”, resulta 0 na entrada do segundo neurônio (N2).

Desta forma temos as seguintes entradas nos neurônios: N1 = 1 e N2 = 0. Como a RNA implementa um XOR, temos a SAÍDA = 1. Na simulação a entidade segue pelo CAMINHO 1.

Com a realização deste primeiro teste foi possível observar que o Encog funcionou sem problemas de incompatibilidades com o código do Ururau. A RNA criada também apresentou os resultados esperados para os cenários possíveis testados.

Contudo, visando uma maior confiabilidade em relação ao funcionamento do Módulo Inteligente acoplado ao Ururau, optou-se pela realização de um novo teste.

4.2.TESTE DE FUNCIONAMENTO

Objetivando verificar o funcionamento do novo módulo inteligente (acoplado ao Ururau), decidiu-se criar um modelo de simulação tanto no módulo acoplado quanto no módulo inteligente desenvolvido por Silva *et al.* (2012), de modo a permitir a comparação de resultados.

Como o modelo de simulação proposto Silva *et al.* (2012), já estava devidamente validado, decidiu-se implementá-lo em ambos módulos inteligentes. Contudo, é importante destacar que se optou por utilizar uma massa de dados diferente e após compará-las.

A figura 4.4 apresenta as etapas da simulação como módulo inteligente se comunicando por meio de socket TCP, e as respectivas ferramentas necessárias para a realização das mesmas.

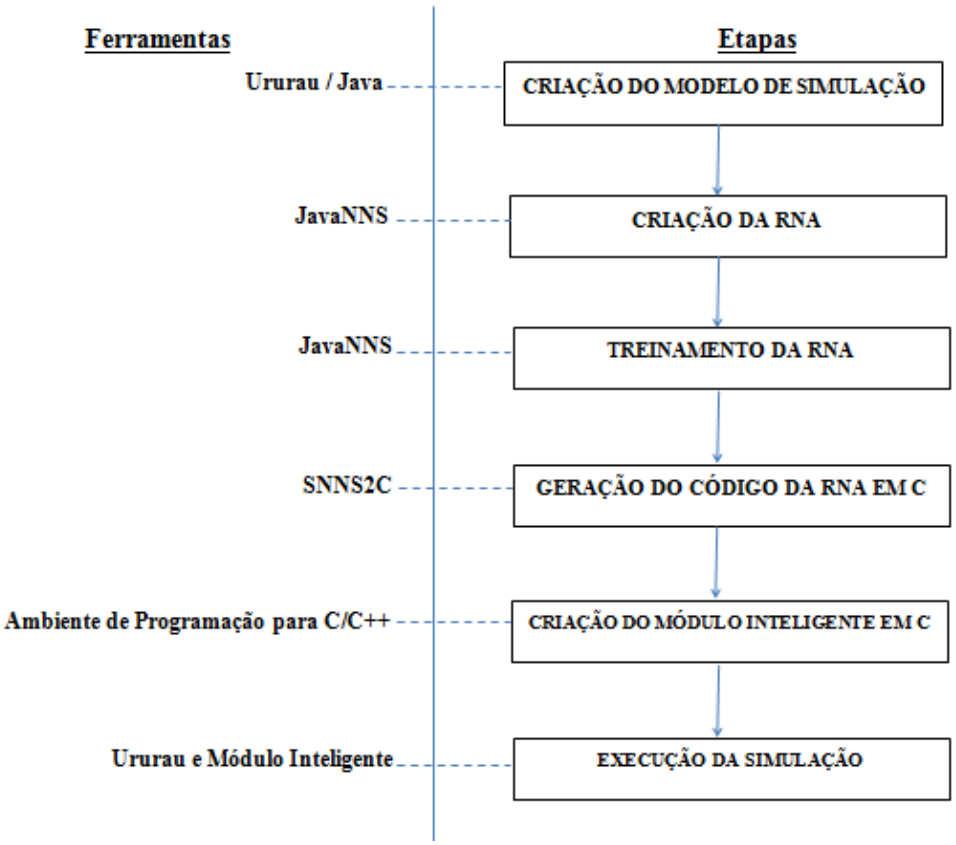

Figura 4.4: Ferramentas e Etapas da Simulação Inteligente.

Conforme pode ser observado na figura 4.4, a primeira etapa a ser realizada é a criação do modelo de simulação, utilizando-se o Ururau.

O modelo desenvolvido por Silva *et. al* (2012) para a realização dos testes pode ser visualizado na figura 4.5.

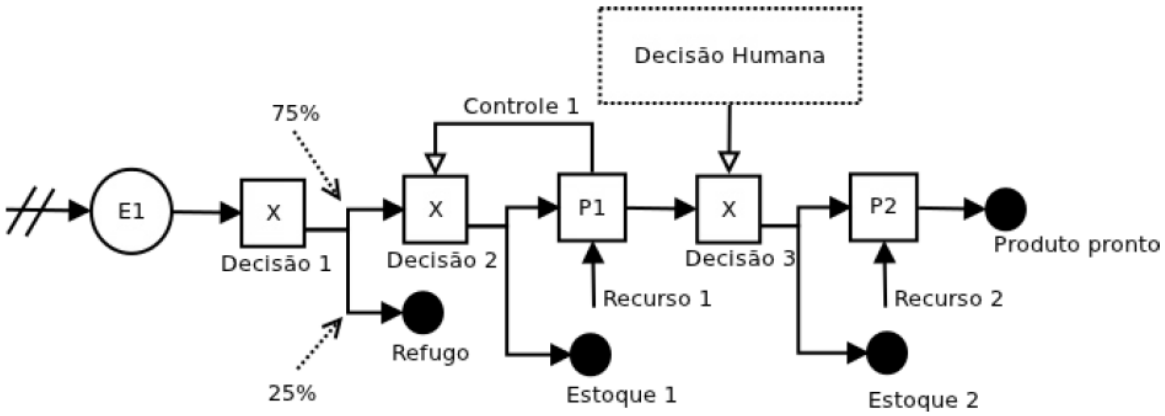

Figura 4.5: Modelo Conceitual em IDEF-SIM com "decisão 3" usando RNA.
Fonte: Silva *et al.* (2012).

Após a criação do modelo de simulação é necessário a criação da RNA, que neste caso foi realizada utilizando o JavaNNS.

Para a construção desta RNA, Silva *et al.* (2012) utilizou três neurônios na primeira camada, sendo um para cada parâmetro, em que se definiu que o "operador" tomaria suas decisões com base nos três parâmetros: tamanho da fila P2, tempo de vida da entidade E1 e turno (figura 4.5). Além disso, foram criados três neurônios na camada oculta e um neurônio na camada de saída. Após a RNA criada, é necessário que a mesma seja treinada. O treinamento também foi realizado utilizando-se a ferramenta JavaNNS.

No trabalho proposto por Silva *et al.* (2012) foi necessário que após o treinamento da RNA fosse gerado um código da mesma, na linguagem de programação C.

Depois da geração do código em C da RNA, o mesmo foi acoplado ao Módulo Inteligente, que implementou comunicação com o Ururau, usando *sockets* TCP.

Quando o modelo no Ururau foi executado foram enviados para o Módulo Inteligente os dados utilizados para a tomada de decisão. Na sequencia, a RNA definiu a decisão que foi enviada de volta para o Ururau.

A figura 4.6 apresenta a nova estrutura desenvolvida, em que todas as etapas ocorrem diretamente no Ururau.

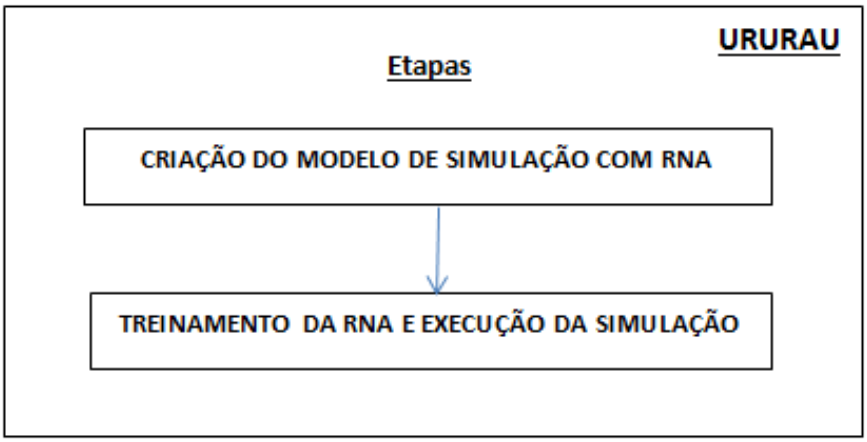

Figura 4.6: Etapas da Simulação Inteligente

As etapas da simulação ficam reduzidas em apenas duas, quando se utiliza o Encog ao invés do JavaNNS para criação e treinamento da rede.

Uma vez criada em JavaNNS, a RNA precisa ser exportada para uma outra ferramenta capaz de gerar o código fonte da mesma, enquanto que o Encog permite que a RNA seja criada a partir de código Java compatível com o código do Ururau. No momento da criação do modelo de simulação é possível informar os dados de criação da RNA.

A figura 4.7 apresenta um trecho de código onde o modelo de simulação é gerado utilizando o Encog acoplado ao Ururau.

É possível notar que as informações referentes ao módulo com a decisão inteligente fazem parte diretamente do código do modelo de simulação. Desta forma, mudanças na RNA não ocasionam mudanças na estrutura do módulo, possibilitando um acoplamento mais simples e menos trabalhoso para o modelador. A partir do

Ururau também é possível inserir os dados de treinamento para a RNA, e logo após iniciar a execução do modelo.

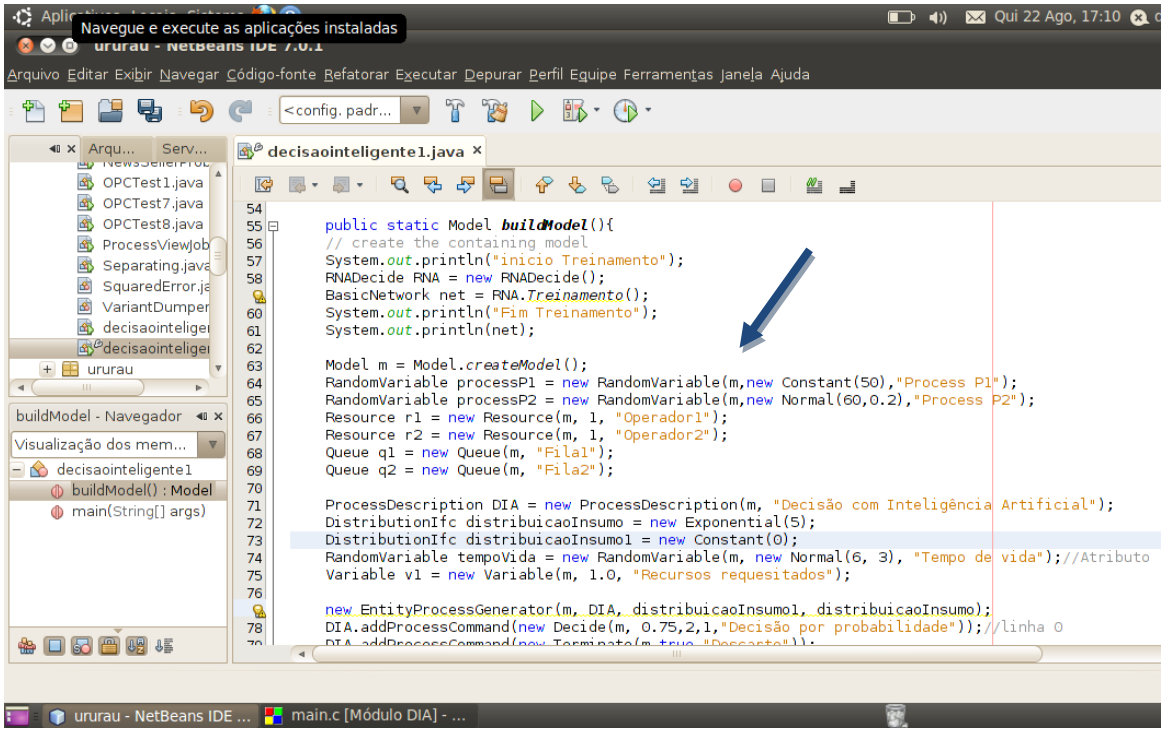

Figura 4.7: Modelo criado utilizando Encog acoplado ao URURAU

Utilizando o modelo de Silva, *et al.* (2012), uma massa de dados foi submetida à testes, nos dois módulos inteligentes. Os dados dos experimentos foram comparados conforme a tabela 4.2.

Tabela 4.2: Resultados dos Experimentos 1 e 2.

Parâmetro	Experimento 1	Experimento 2
Ferramenta de RNA	**JavaNNS**	**Encog**
Tempo médio das entidades na fila P2	233,67 min	235,22min
Quantidade média de entidades na fila P2	12,01 un	12,20 un
Recurso R2 ocupado	96,48%	96,86%
Decisões com RNA	2116 unidades	
Decisões divergentes entre o experimento 1 e 2	10 unidades	
Percentual de decisões divergentes	0,47%	

Usando uma nova massa de dados implementou-se o modelo de simulação de duas maneiras diferentes, usando o JavaNNS e acoplando a biblioteca Encog diretamente no Ururau. Comparando os resultados da simulação da rede neural JavaNNS (Silva *et al.* ,2012) e com Encog observa-se que das 2116 decisões, as RNAs tomaram a mesma decisão em 2106 vezes.

Em relação ao modelo de simulação observou-se uma pequena variação entre os valores de “Tempo médio das entidades na fila P2”, “Quantidade média de entidades na fila P2” e “Recurso R2 ocupado” foram obtidos nos dois experimentos. Contudo é difícil afirmar se tais variações são consequência das decisões divergentes entre as RNAs, que ocorrem em 0,47% das vezes, ou da aleatoriedade dos sistemas estocásticos.

4.3.SIMULAÇÃO COM INTELIGÊNCIA COMPUTACIONAL NO URURAU

4.3.1.Módulo decisor do Ururau

Para possibilitar a configuração da RNA no nível de interface gráfica do usuário, foi preciso realizar adaptações no código do Ururau, para tal foi utilizado o IDE Netbeans 7.0.1 e a linguagem de programação Java. Desta forma, o módulo decisor foi adaptado, conforme a figura 4.8.

Ao se fazer a opção de utilizar o módulo decisor do Ururau como uma RNA com 2 caminhos, é necessário informar alguns parâmetros de configuração da rede.

O primeiro campo a ser preenchido é o caminho de localização do arquivo .csv que contém os dados de treinamento da RNA. O campo seguinte diz respeito ao algoritmo que será utilizado para treinar a rede.

O campo “Expressão do neurônio 1”, deverá receber a expressão que permite a coleta dos dados do avaliador de expressões do Ururau, durante o tempo de simulação. Esses dados serão utilizados como entradas do neurônio 1.

O mesmo ocorre com o campo “Expressão do neurônio 2”, cujos dados obtidos do avaliador de expressões durante o tempo de simulação, servirão como entradas para o neurônio 2, e assim sucessivamente, até “Expressão do neurônio 5”.

A tela de configuração da RNA foi criada para uma rede com 5 neurônios de entrada, para atender as necessidades deste trabalho, porém, pode ser adaptada para receber qualquer valor, dependendo do modelo que for desenvolvido.

Os campos seguintes, dizem respeito a quantidade de neurônios na camada oculta e o número de neurônios na camada de saída, que neste caso é o valor 1.

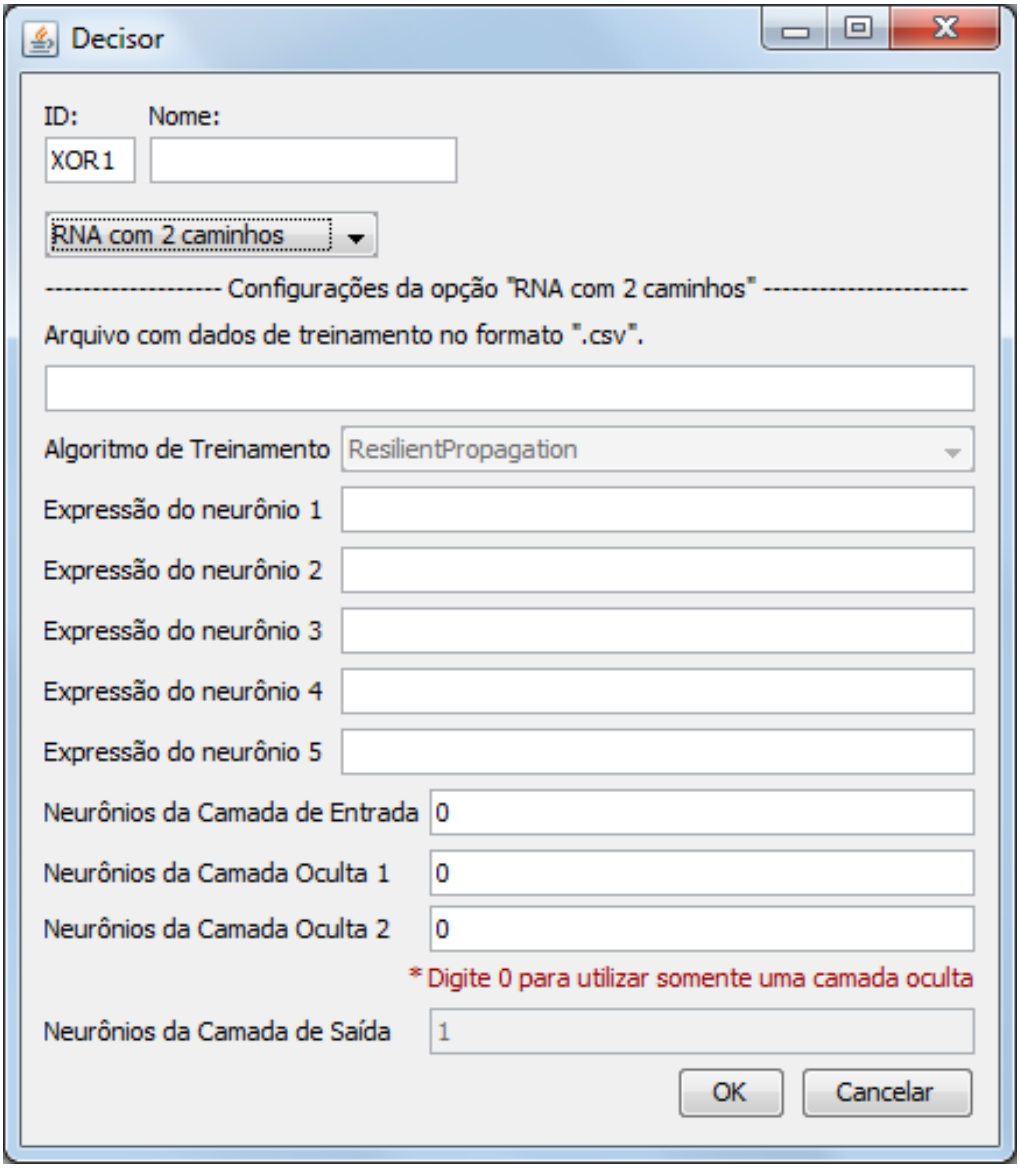

Figura 4.8: Tela de configuração do módulo decisor RNA.

4.3.2.Modelo de Simulação com RNA no Ururau

O modelo de simulação criado é composto por uma parte que é responsável pelo fluxo de veículos e pedestres nas vias, e por outra parte que é responsável pelo controle do cruzamento.

Os dados que alimentam o modelo foram obtidos por meio de medições realizadas no local do cruzamento. Após coletados e tratados, os dados obtidos foram submetidos ao *software Input Analyzer*, que gerou as funções mais aderentes a cada conjunto de dados. As funções obtidas serão demonstradas ao longo deste tópico.

O controle do semáforo é realizado pela RNA, que no Ururau está disponível por meio do módulo decisor RNA destacado na figura 4.9. Esta figura apresenta o modelo de simulação do sistema em estudo.

A primeira parte do modelo representa as vias principal e secundária, sendo que na via principal há o fluxo de veículos, e uma faixa de travessia de pedestres, cujo fluxo de pessoas também foi representado no modelo. Um recorte detalhado de cada uma das vias modeladas será apresentado nesta seção de forma a tornar o entendimento mais claro.

A segunda parte do modelo representa a lógica de controle do cruzamento sendo realizada por um guarda de trânsito. Nesta parte, as decisões do guarda em relação à abertura ou fechamento da via é representado pelo módulo decisor RNA, que na figura 4.9 encontra-se destacado.

O fluxo de veículos na via principal é representado pela figura 4.10. O módulo “Criar” (E1), é responsável pela geração das entidades que percorrem o modelo, que são criadas de acordo com uma taxa de chegada previamente definida por uma função estatística. Conforme destacado anteriormente, as funções foram obtidas utilizando a ferramenta *Input Analyzer,* com base nos dados coletados no sistema real.

L1 corresponde ao módulo “segurar” do Ururau, que como o nome diz, é capaz de segurar as entidades em uma fila, enquanto a condição for verdadeira. Neste caso, enquanto a variável “variavelsemaforo” for igual a 1, a via principal estará fechada para o trânsito de veículos.

C1 é responsável pela atribuição da variável “variavelcarrorua1”, que será incrementada toda vez que uma entidade veículo percorrer o sistema durante a simulação. Esta informação é útil para a geração de relatórios de simulação e realização de testes.

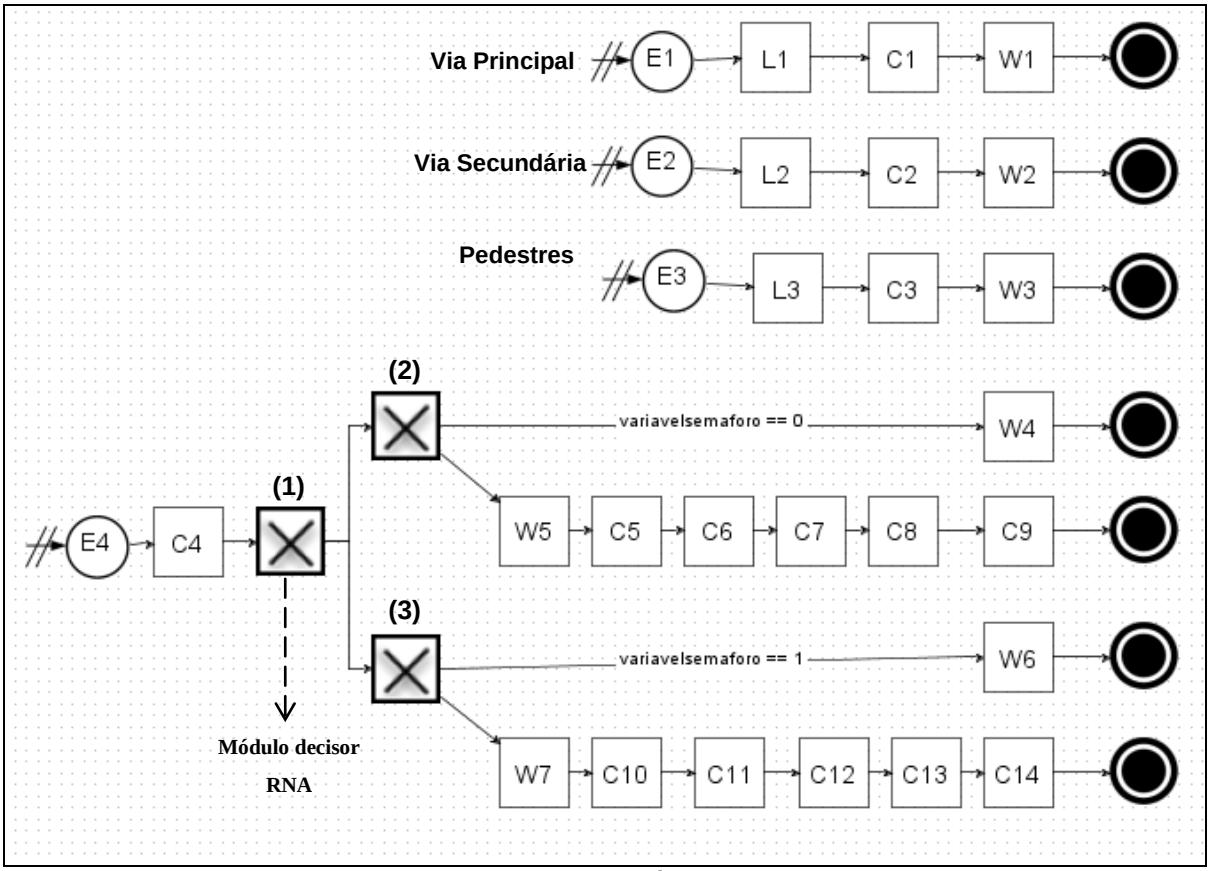

Figura 4.9: Modelo de Trânsito no Ururau

W1 é o módulo que possibilita que os dados da contagem dos veículos sejam escritos em um arquivo texto, enquanto o modelo é executado.

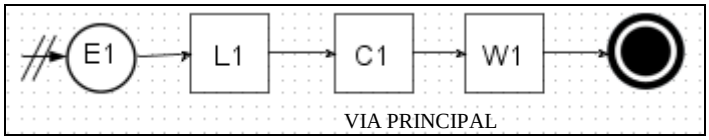

Figura 4.10: Fluxo de veículos na via principal.

Os parâmetros de configuração desta parte do modelo seguem conforme quadro 4.1.

Módulo	Nome	Descrição	Dados de Configuração
Criar	E1	Responsável pela criação de entidades.	T. Chegadas: LOGN(2.78, 3.84) T. Primeira Chegada: 2 Máx. Chegadas: Infinito
Segurar	L1	Segura as entidades em uma fila, até que a condição seja satisfeita.	Expressão: variavelsemaforo==1
Atribuição	C1	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua1 Valor: variavelcarrorua1+1
Escrever	W1	Escreve em arquivo valores da de uma variável	Nome da variável: variavelcarrorua1 Arquivo: saída.txt
Terminar	-	Termina a lista de comandos	-

Quadro 4.1: Módulos do Ururau com descrição e dados de configuração da via principal.

A parte do modelo que representa o fluxo de veículos da via secundária é muito semelhante ao da via principal (figura 4.11).

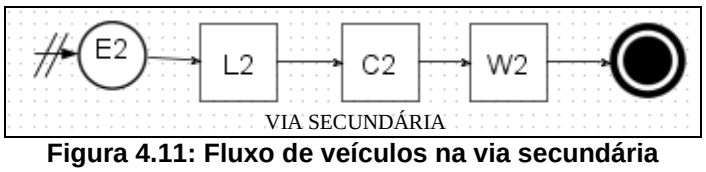

Figura 4.11: Fluxo de veículos na via secundária

Neste caso, o que diferencia ambas as vias são os parâmetros de configuração do modelo, que se encontram registrados no quadro 4.2.

Módulo	Nome	Descrição	Dados de Configuração
Criar	E2	Responsável pela criação de entidades.	T. Chegadas: LOGN(3.5, 2.5) T. Primeira Chegada: 2 Máx. Chegadas: Infinito
Segurar	L2	Segura as entidades em uma fila, até que a condição seja satisfeita.	Expressão: variavelsemaforo==0
Atribuição	C2	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua2 Valor: variavelcarrorua2+1
Escrever	W2	Escreve em arquivo valores da de uma variável	Nome da variável: variavelcarrorua2 Arquivo: saída2.txt
Terminar	-	Termina a lista de comandos	-

Quadro 4.2: Módulos do Ururau com configuração da via secundária.

A figura 4.12 representa o fluxo de pedestres no sistema. Neste caso, as entidades que percorrem o modelo seguem o comportamento das pessoas que fazem a travessia da via principal.

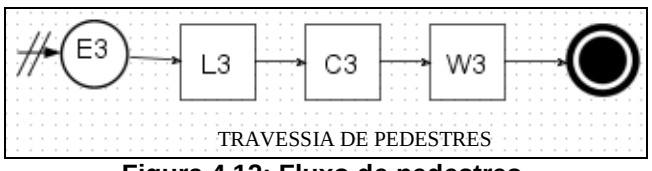

Figura 4.12: Fluxo de pedestres.

Os parâmetros de configuração desta parte do modelo podem ser visualizados no quadro 4.3.

Módulo	Nome	Descrição	Dados de Configuração
Criar	E3	Responsável pela criação de entidades.	T. Chegadas: LOGN(1.4, 1.32) T. Primeira Chegada: 2 Máx. Chegadas: Infinito
Segurar	L3	Segura as entidades em uma fila, até que a condição seja satisfeita.	Expressão: variavelsemaforo==0
Atribuição	C3	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelpessoa Valor: variavelpessoa+1
Escrever	W3	Escreve em arquivo valores da de uma variável	Nome da variável: variavelpessoa Arquivo: saída3.txt
Terminar	-	Termina a lista de comandos	-

Quadro 4.3: Módulos do Ururau com configuração para trânsito de pedestres.

A figura 4.13 mostra um recorte mais detalhado da parte do modelo de simulação responsável por realizar o controle do cruzamento de trânsito. Nesta parte do modelo está o decisor RNA que vai atuar como o guarda de trânsito, e decidirá o momento de abertura e fechamento de cada uma das vias. O Apêndice E mostra os módulos utilizados com as respectivas descrições e dados.

O módulo E4 que é responsável por criar entidades para o modelo, gera uma entidade a cada segundo simulado. Desta forma, a entidade que percorre esta parte do modelo é a representação de uma unidade de tempo, em segundos.

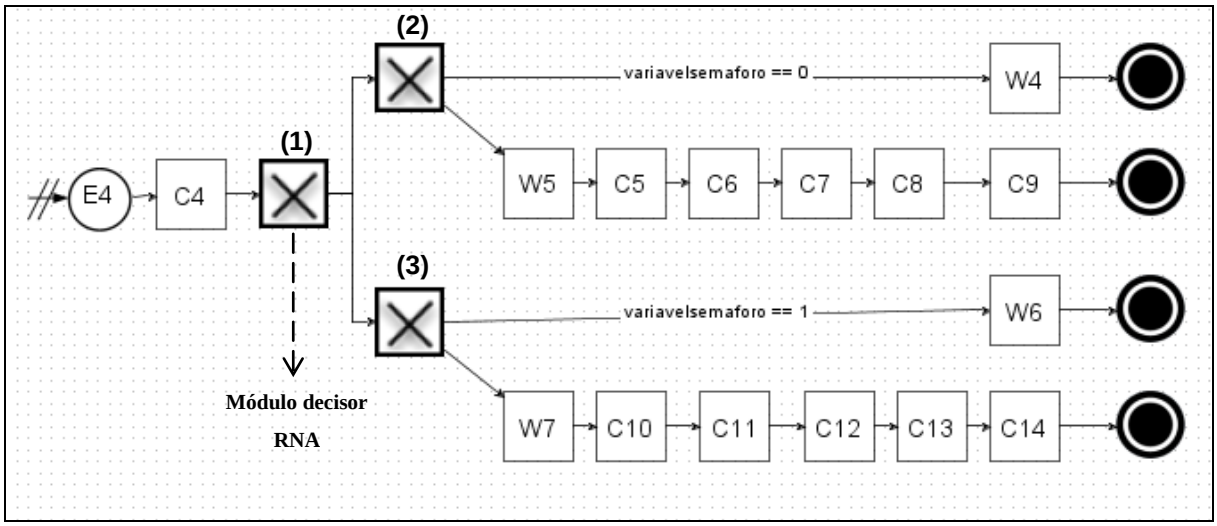

Figura 4.13: Controle do Cruzamento

Analisando o sistema real, percebeu-se que os guardas de trânsito só começam a verificar a necessidade de abrir ou fechar uma das vias, após um tempo mínimo, desta forma o módulo C4 atribui uma variável “tempo” ao modelo que é responsável por atuar como um contador, fundamental para garantir o sincronismo em todo o modelo de simulação.

Toda vez que uma entidade chega ao módulo decisor RNA (1) a mesma é acionada recebendo como dados de entrada:

i) tamanho da fila de veículos da via principal;
ii) tamanho da fila de veículos da via secundária;
iii) tamanho da fila de pedestres;
iv) valor da variável “tempo”;
v) valor da variável “variavelsemaforo”.

Os módulos decisores (2) e (3) são utilizados para detectar a situação das vias no momento anterior a decisão da RNA. Essa informação é importante para permitir que as variáveis “tempo” e “variavelsemaforo” sejam devidamente atualizadas. A variável “variavelsemaforo” é fundamental para integrar a parte do modelo responsável pelo controle de abertura e fechamento das vias, com a parte do modelo responsável pelos fluxos de veículos e pedestres.

A tela que é aberta para a configuração da RNA no módulo decisor pode ser visualizados na figura 4.14.

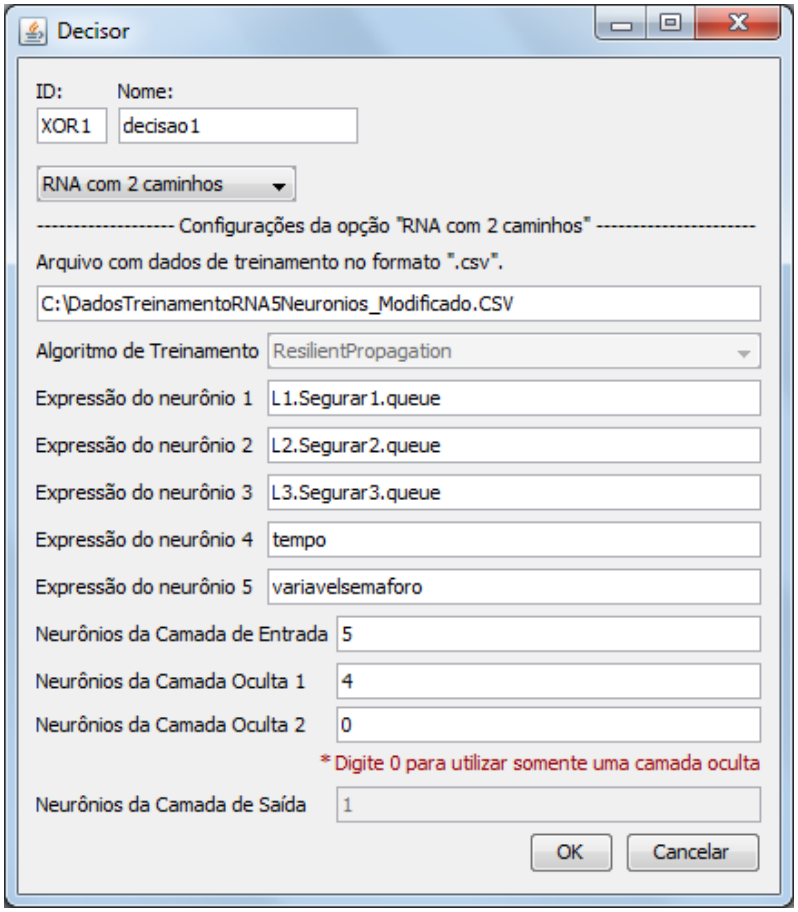

Figura 4.14: Tela de configuração do decisor RNA para modelo de trânsito.

O algoritmo de treinamento utilizado é o *Resilient Propagation Training.*

Algoritmos de treinamento com propagação utilizam treinamento supervisionado, em que é necessário informar valores de entradas e saídas esperadas. Uma vantagem em relação ao uso do *Resilient Propagation Training* é o fato do mesmo não requerer a utilização de parâmetros, como taxa de aprendizagem, que muitas vezes é difícil de determinar (HEATON, 2011).

As expressões de entrada dos neurônios 1, 2 e 3 correspondem aos dados das filas que serão formadas em L1, L2 e L3, respectivamente. Já as expressões de entrada dos neurônios 4 e 5 estão relacionadas com os valores das variáveis “tempo” e “variavelsemaforo”.

A RNA criada para o modelo em questão utiliza 4 neurônios na camada oculta e 1 neurônio na camada de saída.

4.3.1.Execução do Modelo de Simulação com RNA no Ururau

Durante o tempo de execução do modelo de simulação, a RNA é ativada sempre que uma entidade precisa tomar uma decisão sobre sua rota. A figura 4.15 representa um esquema em que é possível acompanhar como ocorre essa tomada de decisão pela RNA no Ururau.

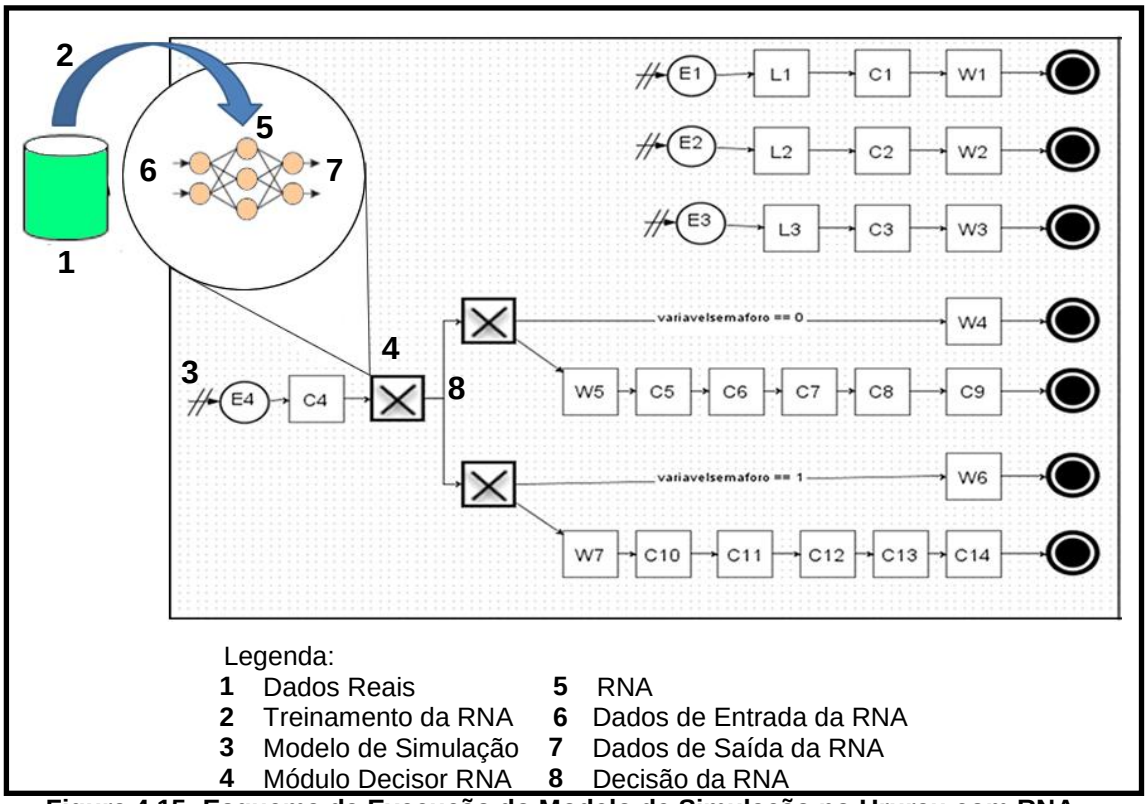

Figura 4.15: Esquema da Execução do Modelo de Simulação no Ururau com RNA.

Os dados coletados dos sistemas reais modelados (1) são utilizados para o treinamento da RNA (2), que ocorre no momento que antecede o início da simulação.

No instante posterior, as entidades iniciam seu percurso ao longo do modelo de simulação (3). Toda vez que uma entidade chega até o módulo decisor (4) a RNA (5), previamente treinada, é ativada.

Os dados que alimentam os neurônios de entrada da rede (6) são dados do próprio modelo, obtidos dinamicamente no momento em que a RNA é ativada. Estes dados podem ser os mais diversos, obtidos do avaliador de expressões do Ururau,

como por exemplo, o tamanho de uma determinada fila, o tempo em que uma entidade está aguardando na fila, ou até mesmo o resultado de uma expressão.

Como saída (7), a RNA apresenta a decisão (8) sobre qual caminho a entidade deve percorrer. Essa decisão é devolvida ao modelo, que retoma a iteração enquanto a simulação estiver sendo executada.

4.3.1.1.Resultados da Execução do Modelo

Para validar o modelo, foram observados os dados do sistema de trânsito real, e posteriormente, estes foram comparados com o resultado da simulação. A simulação foi executada utilizando um computador Intel *Core* i5 2.6 Ghz, 4Gb de memória RAM e sistema operacional *Windows* 7.

Foram realizadas duas rodadas com o modelo de simulação, sendo cada uma delas com uma replicação apenas. As rodadas foram chamadas de Simulação A e Simulação B, respectivamente. A tabela 4.3 apresenta informações que demonstram os resultados obtidos.

Com a análise dos resultados da execução do modelo, foi possível observar que o valor do erro da RNA possui influência no resultado obtido com a simulação, uma vez que se pôde perceber que na Simulação B, onde o erro da RNA foi menor, os valores médios obtidos se aproximaram mais dos valores médios do sistema real para maior parte dos parâmetros observados.

Tabela 4.3: Resultados da Execução do Modelo de Simulação com RNA.

	Real		**Simulação A**		**Simulação B**	
Parâmetro	**Média**	**IC (95%)**	**Média**	**IC (95%)**	**Média**	**IC (95%)**
T.M.A.P (s)	55,33	[47,29; 63,37]	49,11	[46,88; 51,33]	52,33	[50,06; 54,60]
T.M.A.S (s)	40,12	[34,16; 46,08]	40,98	[31,57; 50,38]	40,93	[39,71; 42,16]
Q.M.P.A.C (un)	28,77	[18,47; 39,07]	28,81	[22,10; 35,50]	30,57	[28,18; 32,95]
Q.M.V.P.P.C (un)	21,57	[13,85; 29,28]	18,65	[16,64; 20,66]	20,73	[19,91; 21,55]
Erro da RNA			**0,020558**		**0,012998**	

Legenda:

T.M.A.P (s) : Tempo Médio de Abertura da Via Principal (em segundos)

T.M.A.S (s) : Tempo Médio de Abertura da Via Secundária (em segundos)

Q.M.P.A.C (un) : Quantidade Média de Pedestres atravessando a via por Ciclo (unid.)

Q.M.V.P.P.C (un) : Quantidade Média de Veículos percorrendo a via Principal por ciclo (unid.)

Neste ponto é importante citar que os valores médios obtidos estão bem próximos para os parâmetros observados em todas as situações apresentadas, porém em alguns casos pôde-se notar uma diferença maior entre o Intervalo de Confiança (IC) da medição do sistema real com a Simulação A e a Simulação B.

Um dos motivos para esta diferença pode ser o fato das medições das simulações possuírem uma quantidade de observações maior que do sistema real, devido à facilidade de geração de dados automaticamente no sistema simulado.

Além disso, alguns fatores podem ter influência sobre estes dados, como por exemplo, o fato de que em algumas vezes a decisão do guarda não reflete nenhum padrão pré-estabelecido. Foi possível observar que em alguns momentos enquanto faz o controle do cruzamento, o guarda de trânsito se distrai conversando com alguém que o aborda na rua, e sendo assim, este não toma nenhuma decisão por um período de tempo bem superior ao que vinha sendo executado.

Da mesma forma, em alguns momentos os veículos que se acumulam na via têm dificuldade de se dispersarem em função de haver algum impedimento, como por exemplo, um ônibus que realiza o desembarque de passageiros fora do ponto, ou pedestres que atravessam fora da faixa.

Todas as variáveis do sistema real quando analisadas, demonstram uma forte interligação entre si, uma vez que por se tratar de um cruzamento, sempre que uma via estiver aberta para o fluxo de veículos, a outra deverá estar fechada.

Ao tentar reproduzir estes comportamentos atípicos no modelo de simulação utilizando RNA, a massa de dados de treinamento da rede recebem valores que não estão dentro de nenhum padrão, o que pode influenciar tanto no erro da RNA, quanto própria decisão tomada por ela.

Por outro lado, tentar retirar esses pontos discrepantes da massa de dados da simulação, significa modificar a curva da função que representa a taxa de chegada das entidades no modelo, ou seja, o modelo de simulação passa a não representar o sistema real. Por estes motivos, optou-se por manter os dados coletados conforme o sistema real, e analisar o seu impacto, observando o erro da RNA.

A figura 4.16 permite observar os resultados obtidos em relação ao tempo médio de abertura das vias principal e secundária no sistema de trânsito real e no sistema simulado (Simulação A e Simulação B).

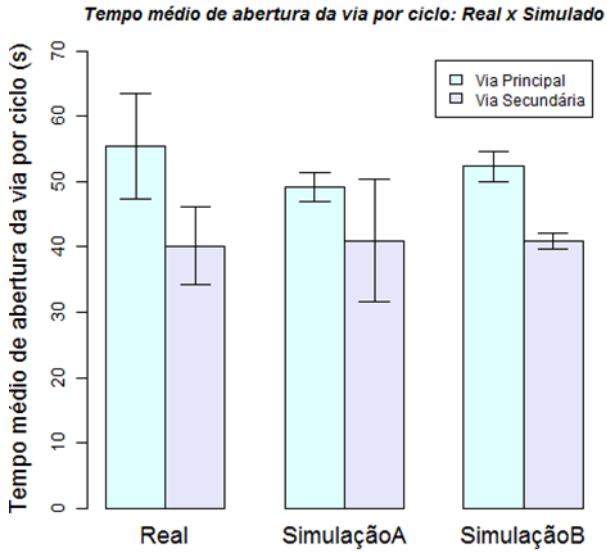

Figura 4.16: Comparação do tempo médio de abertura das vias no sistema real e no simulado.

Os tempos obtidos na simulação B se aproximam um pouco mais do sistema de trânsito real, corroborando com o fato da RNA ter apresentado um menor valor de erro durante a simulação, esta informação pode ser melhor consultada na tabela 4.3.

Um comportamento semelhante pode ser observado pela figura 4.17, que apresenta uma comparação do tempo médio de abertura das vias no sistema real e no simulado.

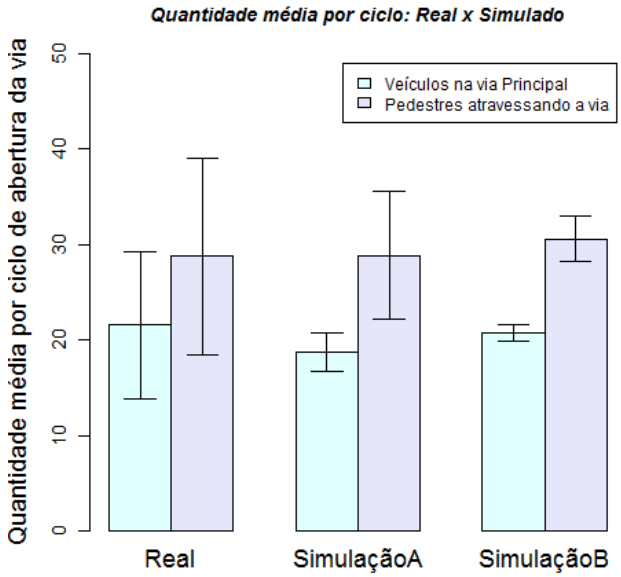

Figura 4.17: Comparação da quantidade média de veículos e pedestres no sistema real e no simulado.

4.3.1.2. Cenários da Simulação

Após a validação do modelo, dois novos cenários foram criados com o objetivo de observar se RNA responderia bem aos estímulos para cada um destes cenários.

Cenário 1: Aumento da taxa de chegada de veículos na via principal – Observou-se que há aumento expressivo do fluxo de veículos percorrendo a via principal em horários de pico, que correspondem aos horários de entrada e saída de alunos. Esta via possui escolas ao longo dela, e ainda é uma das formas de acesso a outras instituições de ensino que se localizam próximo da mesma. Desta forma, visando representar um aumento no fluxo de veículos na via principal, a taxa de chegada de veículos foi duplicada. Foi utilizada a mesma função.

Cenário 2 – Aumento da taxa de chegadas de pedestres - Por se tratar de uma área onde estão localizados o mercado municipal, o camelódromo e diversas lojas populares, a região onde o cruzamento em estudo está compreendido é trafegada por um grande número de pedestres. Em datas comemorativas como Natal e Dia das Mães, percebe-se um aumento considerável no número de pessoas que buscam a região para realizarem suas compras por um preço mais acessível. Para representar este cenário, a função que representa a taxa de chegada de pedestres para atravessar a via principal foi duplicada. Neste caso, também se manteve a mesma função que representa o sistema real.

A tabela 4.4 trás as informações a respeito do sistema real, e dos dois cenários criados.

Tabela 4.4: Resultados da Execução do Modelo de Simulação nos Cenários 1 e 2.

	Real		Cenário 1		Cenário 2	
Parâmetro	**Média**	**IC (95%)**	**Média**	**IC (95%)**	**Média**	**IC (95%)**
T.M.A.P (s)	55,33	[47,29; 63,37]	63,15	[57,64; 68,66]	45,40	[43,59; 47,21]
T.M.A.S (s)	40,12	[34,16; 46,08]	39,72	[38,65; 40,49]	43,96	[37,26; 50,66]
Q.M.P.A.C (un)	28,77	[18,47; 39,07]	28,58	[27,29; 29,87]	62,48	[52,79; 72,16]
Q.M.V.P.P.C (un)	21,57	[16,42; 34,72]	43,27	[38,90; 47,65]	16,65	[15,52; 17,79]
Erro da RNA			**0,015810**		**0,016282**	

Legenda:

T.M.A.P (s) : Tempo Médio de Abertura da Via Principal (em segundos)

T.M.A.S (s) : Tempo Médio de Abertura da Via Secundária (em segundos)

Q.M.P.A.C (un) : Quantidade Média de Pedestres atravessando a via por Ciclo (unid.)

Q.M.V.P.P.C (un) : Quantidade Média de Veículos percorrendo a via Principal por ciclo (unid.)

Nota-se que ao aumentar a taxa de chegada de veículos na via 1 (cenário 1) a RNA responde aumentando o tempo médio de abertura da mesma, o que possibilita uma maior quantidade de veículos percorrendo a via por ciclo. O efeito desta decisão tem algum impacto sobre a via 2, que tem seu tempo médio levemente reduzido se comparado ao sistema real. A quantidade média de pedestres atravessando a via não sofreu grande impacto.

A figura 4.18 ajuda na visualização dos dados referentes à quantidade de veículos trafegando nas vias para os cenários 1 e cenário 2.

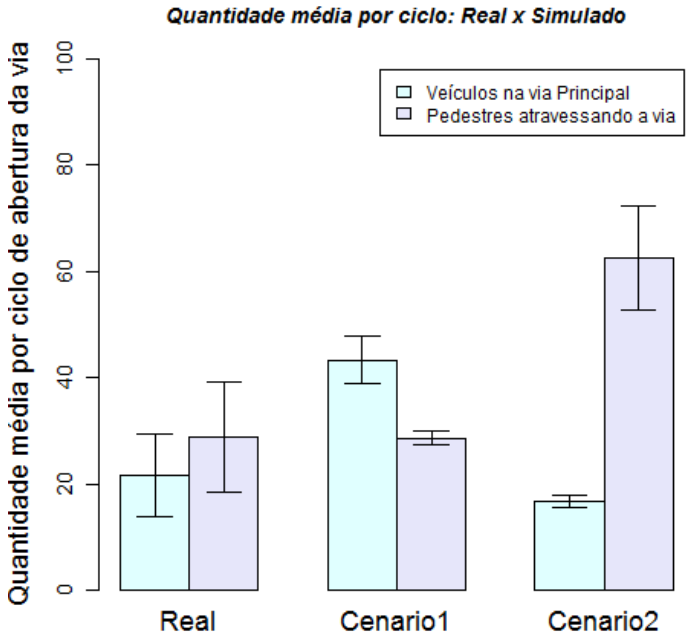

Figura 4.18: Comparação da quantidade média de veículos e pedestres em diferentes cenários

Nota-se que no cenário 2, em que se tem um aumento do número de pedestres para atravessar a via 1, que a RNA toma decisões diferentes. Neste caso, o que se pôde perceber foi que a RNA optou por diminuir o tempo de abertura da via 1, diminuindo também quantidade média de veículos percorrendo a via principal, conforme pode ser visualizado na figura 4.19. A quantidade de pedestres atravessando a via aumentou consideravelmente neste cenário.

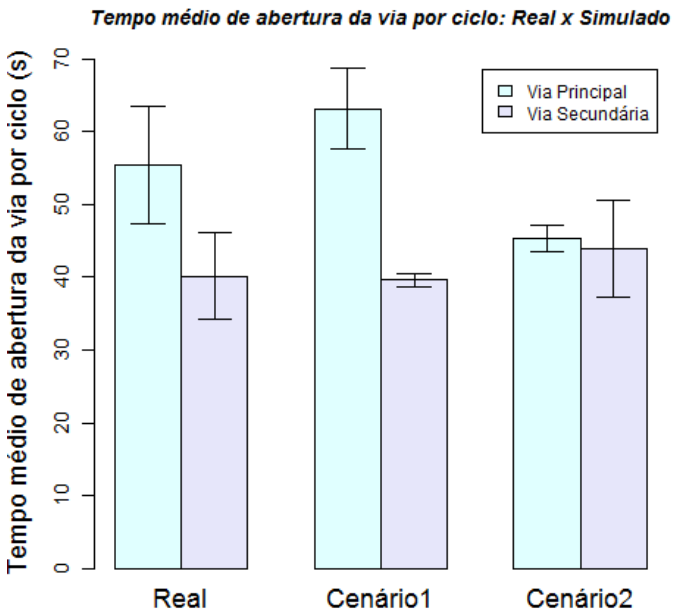

Figura 4.19: Comparação do tempo médio de abertura da via por ciclo em diferentes cenários

5. CONSIDERAÇÕES FINAIS

5.1.CONCLUSÕES

Foi possível observar que os sistemas de trânsito possuem uma forte relação com as linhas de produção das indústrias, se considerarmos as semelhanças existentes entre eles, tornando-se desta forma um ambiente propício para aquisição de dados de forma livre para realização de estudos e analogias.

Contudo é importante ressaltar que em alguns casos, o comportamento do guarda de trânsito se difere de uma situação típica de linha de produção, uma vez que em ambiente industrial, o operador deve obedecer a padrões de conduta pré-estabelecidos para operar o sistema. Embora essa também seja uma verdade para um guarda de trânsito, o ambiente aberto e livre das ruas, possibilita que muitas vezes esse comportamento seja influenciado, e não reflita os padrões pré-definidos.

Relacionado às ferramentas, as pesquisas bibliográficas apontaram o Encog como o *framework* que atendeu aos requisitos estabelecidos para a elaboração deste trabalho. O resultado do teste de acoplamento permitiu validar o funcionamento do módulo inteligente acoplado ao modelo de simulação. Além disso, o *framework* Encog se mostrou adequado para a criação das RNAs, sem apresentar erros de funcionamento e incompatibilidades com o código do Ururau.

Em relação ao teste comparativo realizado com o módulo proposto e o apresentado pela literatura, foi possível observar que os valores dos resultados

encontrados foram compatíveis, mostrando que foi possível executar a RNA de maneira confiável acoplada ao *software* Ururau.

O *software* Ururau se apresentou como uma ferramenta gratuita e de código fonte aberto, com grande potencial acadêmico. Até a redação deste trabalho, ainda não havia sido encontrado na literatura, um *software* de simulação capaz de executar diretamente uma RNA em tempo de simulação.

Da mesma forma, pôde-se considerar pioneira possibilidade de configurar uma RNA a partir da interface gráfica do ambiente de simulação, e se obter os dados que alimentam os neurônios de entrada da RNA em tempo de execução.

O modelo criado com dados obtidos do sistema de trânsito real foi capaz de representar um cruzamento controlado por um guarda. Neste caso, a RNA conseguiu tomar decisões semelhantes a este, considerando um intervalo de confiança de 95%.

Foi possível notar que a RNA fornece uma aproximação em relação ao comportamento real do guarda de trânsito. Contudo, ela é treinada novamente a cada simulação, o que gera resultados próximos, porém diferentes devido a natureza aleatória do treinamento da rede. Este comportamento pôde ser verificado nas Simulações A e B cujos dados foram apresentados na tabela 4.3.

Considerando os resultados obtidos nos dois cenários testados, pôde-se observar que a RNA correspondeu ao esperado uma vez que tomou decisões distintas quando apresentadas situações diferentes das que vinham sendo realizadas.

5.2.LIMITAÇÕES DO TRABALHO

Embora a utilização do Ururau tenha permitido o desenvolvimento do módulo com a RNA acoplada ao seu código de maneira satisfatória, foi possível observar que o *software* ainda apresenta algumas limitações para representar modelos de trânsito.

Uma das limitações percebida foi a dificuldade de se integrar a parte do modelo que representa o controle das vias (realizado pelo guarda de trânsito), com a parte do modelo que representa o fluxo de veículos nas mesmas.

A utilização de variáveis permitiu que essa integração fosse possível, porém o módulo “segurar” que foi utilizado para representar o controle da via que é fechada ou aberta para o fluxo de veículos e pessoas, possui uma característica de dispersão da fila que não é compatível com a realidade do sistema real, já que a mesma é desfeita de uma só vez. Para minimizar os impactos desta característica sobre a RNA, foi necessário realizar uma discretização dos dados utilizados para treinamento da rede, o que demandou um grande esforço.

Como o modelo de trânsito trabalha com ciclos (aberto e fechado), a observação dos resultados da simulação também não se demonstrou uma tarefa trivial, uma vez que o relatório gerado pelo Ururau trás informações sobre valores obtidos ao final da simulação, porém para se validar o funcionamento do modelo trânsito, é necessária a observação desses valores ciclo a ciclo.

A possibilidade de se coletar informações sobre o modelo de simulação em arquivos de texto utilizando o módulo *“writer”* colaborou muito na coleta de dados para análise de resultados, porém a necessidade de se ter um módulo para cada tipo de dado coletado, pode dificultar o entendimento do modelo como um todo, uma vez que há um aumento expressivo de módulos *“writer”* no modelo, em casos em que é necessário coletar diferentes tipos de informação para análise.

5.3.TRABALHOS FUTUROS

Como sugestão de trabalhos futuros pode-se disponibilizar no Ururau a possibilidade de se configurar RNAs utilizando diversos algoritmos de treinamento e outras arquiteturas de redes.

Um estudo de aplicação de RNAs em modelos de simulação envolvendo diversas replicações pode contribuir para observação e análise da relação entre o erro da RNA e o resultado obtido com a simulação do modelo.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMANE, M.; *et al.* Modeling and controlling an isolated urban intersection based on cooperative vehicles. In: **Transportation Research.** Part C 28, p. 44-62. 2013. Disponível em: <http://hal.inria.fr/docs/00/80/80/76/PDF/Ahmane-TRC.pdf>. Acessado em: 20 de agosto de 2013.

BANKS, J.; CHWIF, L. Warnings about simulation. In: **Journal of Simulation** .1-13.2010.

BANKS, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2010. **Discrete-Event System Simulation**. 5th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.

BAPTISTA, R. C. T. ; RANGEL, J. J. A. . Simulação a Eventos Discretos de uma Via Semaforizada com Controle Automatizado em Tempo Real. In: **Revista Produção Online,** v. 13, p. 290-317, 2013. Disponível em:< http://www.producaoonline.org.br/rpo/article/view/1149/994>. Acessado em: 10 de Março de 2013.

BAPTISTA, R. C. T.; RANGEL, J. J. DE A. Modelo de Simulação para Análise de desempenho de um Cruzamento Viário Semaforizado. Expansão com Qualidade e Interface com o Mercado. In: **Anais do ENCONTRO MINEIRO DE ENGENHARIA DE PRODUÇÃO (EMEPRO)**. São João *del* Rei: 28 de Maio de. 2011.

BARBOSA, R. A., RODRIGUES, T. L., ALMEIDA, R. P., ESPINDOLA, J. R., & MOREIRA, D. V. Modelagem e Análise do Sistema de Filas de Caixas de Pagamento de uma Drogaria: uma aplicação da Teoria das Filas**.** In: **Anais do ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO (29)**. Salvador, Bahia, Brasil. 06 à 09 de outubro de 2009.

BASILE F.; CHIACCHIO P.; TETA, D. A hybrid model for real time simulation of urban traffic. In: **Control Engineering Practice.** V. 20, p. 123-137. 2012.

BERGMANN, S; STELZER,S; STRASSBURGER, S. On the use of artificial neural networks in simulation-based manufacturing control. In: **Jornal of Simulation.** V.8, p.76-90. 2014.

BITRAN, G. R.; MORABITO, R. Um Exame dos Modelos de Redes de Filas Abertas Aplicadas a Sistemas de Manufatura Discretos: Parte 1. In: **Gestão & Produção**. V2, n.2, p.192-219. Ago 1995. Disponível em: http://www.dep.ufscar.br/docentes/morabito/g&p95a.pdf. Acessado em: 10 de fevereiro de 2013.

BRAGA, A. P.; CARVALHO, A. P. L. F.; LUDERMIR, T. B. **Redes Neurais Artificiais Teorias e Aplicações.** 2 Ed. Rio de Janeiro: LTC, 2011.226p.

CHWIF, L.; MEDINA, A. C.; **Modelagem e Simulação de Eventos Discretos: Teorias & Aplicações**. 3 Ed. São Paulo: Edição do Autor, 2010.

CODEPROJECT. **Benchmarking and Comparing Encog, Neuroph and JOONE Neural Networks.** 03 de Junho de 2010 (a).Disponível em: <http://www.codeproject.com/Articles/85487/Benchmarking-and-Comparing-Encog-Neuroph-and-JOONE>. Acessado em: 28 de Junho de 2013.

CODEPROJECT. **Comparing Neural Networks in Neuroph, Encog and JOONE.** 02 de Junho de 2010 (b).Disponível em: http://www.codeproject.com/Articles/85385/Comparing-Neural-Networks-in-Neuroph-Encog-and-JOO. Acessado em: 28 de Junho de 2013.

COSTA, H.G. Modelo para webibliomining: proposta e caso de aplicação. **Revista da FAE.** Curitiba, v. 13, n.1, p.115-126, jan./jun.2010. Disponível em: http://www.unifae.br/publicacoes/v.13_01-2010.pdf. Acessado em: 30 de janeiro de 2013.

DOTOLI, M., FANTI, M.P. An urban traffic network model via coulored timed Petri nets. In: **Control Engineering Practice.** V.14, p. 1213-1229.2006. Disponível em: www.researchgate.net/.../72e7e52084bdf7da15.pdf. Acessado em: 10 de fevereiro de 2013.

HAYKIN, S. **Redes Neurais Artificiais Princípios e Prática**. 2 Ed. Porto Alegre: Bookman, 2001. 900p.

HEATON, J. **Introduction to Encog 2.5** . Heaton Research, Inc. 108p. 2010. Disponível em: <http://www.Heatonresearch.com/Encog.> Acessado em: 15 de Julho de 2013.

HEATON, J. **Heaton Research Copyright, Trademark and License Information.** Heaton Research, Inc. Disponível em: http://www.heatonresearch.com/legal/copyright. Acesso: 18 de Junho de 2014.

HEATON, J. **Programming Neural Networks with Encog3 in Java**. Heaton Research, Inc. St. Louis, MO, USA. 2011.

HELBING, D. ***Modelling and Optimization of production processes:*** **lessons fron traffic dynamics**. Working Paper. Out.2003. Disponível em: http://www.santafe.edu/research/working-papers/abstract/dacaed19ae05872fce780032a1f12456. Acessado em 15 de Janeiro de2012.

GUIDORIZZI, M. DOS S. ; SANTOS, A.P. B; OLIVEIRA, A.B.; LEONARDI, F. Simulação como Apoio a Tomada de Decisão para a Solução de Problemas Causados pelos Gargalos Formados no Trânsito. A Engenharia de Produção e o Desenvolvimento Sustentável: Integrando Tecnologia e Gestão. In: **Anais do ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO (29).** Salvador, BA, Brasil: 09 de outubro de. 2009.

INGALLS, R. G. Introduction to Simulation. In: WINTER SIMULATION CONFERENCE. **Proceedings of the 2008 Winter Simulation Conference.** Miami. 2008.

JOONE. **An Object Oriented Neural Engine.** 2013. Disponível em: <http://sourceforge.net/projects/joone/>. Acessado em: 03 de Junho de 2013.
KLEIJNEN, J.P.C. Theory and Methodology: Verification and Validation of Simulation Models. In: **European Journal of Operational Research.** V. 82.145-162. 1995.

KOVÁCS, Z.L. **Redes Neurais Artificiais Fundamentos e Aplicações**. 3 Ed. São Paulo: Editora Livraria da Física.174p. 2002.

LAW. A.M. A Tutorial on How to Select Simulation Input Probability Distribuitions. In: WINTER SIMULATION CONFERENCE. **Proccedings of the 2012 Winter Simulation Conference.** Berlim – Germany. 2012.

LEAL, F.; ALMEIDA, D.A.; MONTEVECHI, J.A.B. Uma Proposta de Técnica de Modelagem Conceitual para a Simulação através de Elementos do IDEF. **In: Anais do SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL (SBPO-XL).** João Pessoa, PB. 02 à 05 de setembro de 2008.

LÓPEZ-NERI, E. Hiearchical Agent-Based Modelling: A Guadalajara City Case Study on Urban Traffic Simulation. In:**Transport and Telecommunication.** V. 10, n1, 2009.

MOITA, M. H. V.; ALMEIDA, E. S.. Aplicação de Simulação para Obtenção de Soluções ao Tráfego em Rotatória da Cidade de Manaus. In: **Jornal of Transporte Literature**.V6, n.1, pp.93-109.Jan 2012.

MARRONE, P. **Joone Java Object Neural Engine. The Complete Guide: All you need to know about Joone.** 2007. Disponível em: <http://www.joone.org.>. Acessado em: 12 de Março de 2013.

MATVIYKIV,O.M; Faitas, O.I., **Data Classification of Spectrum Analysis Using Neural Network**. Lviv Oblast, Ucrânica: Polytechnic National University, 2013.

MONTEVECHI, J.A.B.; et al. Conceptual modeling in simulation projects by mean adapted IDEF: An application in a Brazilian tech company. In: WINTER SIMULATION CONFERENCE. **Proceedings of the 2010 Winter Simulation Conference**, Baltimore 2010.

MUGNELA, B. S.; NETTO, M. L. GenPolis - Prototipagem e aplicação de uma ferramenta especializada para otimização via algoritmos genéticos de planos fixos de sinalização semafórica em sub-redes urbanas. Cidades Inteligentes. Anais In: **SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO (8).** São Paulo - SP - Brasil: 18 de maio de. 2012. Disponível em: <http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=11307>. Acessado em: 4 Julho de 2012.

NAGATANI, T., HELBING, D. Stability Analyis and stabilization strategies por linear supply chains. In: **Elsivier Science**. Journals. 2008. Disponível em: http://arxiv.org/pdf/cond-mat/0304476v2.pdf. Acessado em: 15 de Março de 2013.

PANTUZA JR., G.; SOUZA, M. J. F. Um Modelo de Simulação no Arena para o Sequenciamento e Redução do Tempo das Viagens dos Caminhões em uma Mina a Céu Aberto. Sustentabilidade na Cadeia de Suprimentos. Anais In: **SIMPÓSIO DE ENGENHARIA DE PRODUÇÃO (19)**. Bauru, Sp, Brasil: 7 nov. 2011. Disponível em: <http://www.simpep.feb.unesp.br/anais_simpep.php?e=6>. Acessado em: 26 de Julho de 2012.

PEIXOTO, T.A.; RANGEL, J.J.A.; MATIAS, I. O. Usando o JSL para Simulação de Monte Carlo. In: **GEPROS – Gestão da Produção, Operações e Sistemas.** Ano 7, n4 Out-Dez/2012.

PEIXOTO, T.A.; Rangel, J.J.A.; Matias, I. O.; Montevechi, J.A.B.; Miranda, R.C. (2013). Ururau – Um Ambiente para Desenvolvimento de Modelos de Simulação a Eventos Discretos.In: **PODes - Revista Eletrônica Pesquisa Operacional para o Desenvolvimento.** V.5, n.3, p.373-405. Disponível em: http://www.podesenvolvimento.org.br/inicio/index.php?journal=podesenvolvimento&page=article&op=view&path[]=216&path[]=229. Acessado em:20 de Fevereiro de 2013.

RINGHOFER, C. A level set approach to modeling general service rules in supply chains.In: **Communications in Mathematical Sciences**. 909-930. 2010.

ROBINSON, S. Conceptual Modeling for Simulation: Issues ad Research Requirements. In: WINTER SIMULATION CONFERENCE. **Proceedings of the 2006 Winter Simulation Conference.** Monterey, CA. 792-800, 2006.

ROSSETI, M. D. Java Simulation Library (JSL): An Open-Source Object-Oriented Library for Discrete-Event Simulation in Java. In: **International Journal of Simulation and Process Modelling**, V. 4, N. 1, p.69-87, 2008.

SARGENT, R. G. (2013) Verification and Validation of Simulation Models. **Journal of Simulation**, v. 7, p. 12-24.

SWAIN, J. J. Discrete Event Simulation Software: New Frontiers in Simulation, In: **OR/MS Today - INFORMS**, V. 34, N. 5, p.32-43, October 2007.

SILVA, D.V.C.; et al. Modelos de Simulação a Eventos Discretos com Aspectos de Decisão Humana: Uma Aplicação com o Ururau. In: **PODes - Revista Eletrônica Pesquisa Operacional para o Desenvolvimento.** V.4, n.3, p.339-355, set-dez . Rio de Janeiro, 2012.

SOUSA, D.L.M., RIBEIRO, P.C.M. Análise dos Impactos Causados no Tráfego por Alterações na Rede Viária utilizando Micro Simulação. In: CONGRESSO DE PESQUISA E ENSINO EM TRANSPORTES**. Anais do Congresso de Pesquisa e Ensino em Transportes (ANPET-18).** Hotel Jurerê Beach Village, Florianópolis SC. 2004.

TIMM, B.W.; KAMAT, V.R. General-purpose construction simulation and visualization tools for modelling and animating urban vehicular traffic operations. In:**Electronic Journal of Information Technology in Construction.** ITcon V. 13, p.564. 2008.

WHITE JÚNIOR, K. P.; INGALLS, R. G. Introduction to Simulation. In: WINTER SIMULATION CONFERENCE. ***Proceedings of the 2009 Winter Simulation Conference.*** Hilton Austin Hotel, Austin, TX . 2009.

ZUBEN, F. J. V. Uma caricatura funcional de redes neurais artificiais. Learning and Nonlinear Models. **Revista da Sociedade Brasileira de Redes Neurais**, V. 1, N. 2, p.- 66-76, 2003. Retirado de :< http://www.deinfo.uepg.br/~ivo/rna_aplicadas_agricultura/Artigos%20RN/Artigo%20Revista%20Uma%20caricatura%20funcional%20das%20RNA.pdf>. Acessado em:17 de janeiro de 2013.

APÊNDICE A - RELAÇÃO DOS 10 ARTIGOS RELACIONADOS À SIMULAÇÃO A EVENTOS DISCRETOS COM MAIOR NÚMERO DE CITAÇÕES

Autores	Título	Ano	Periódico	Citações
Bemporad A.; Morari M.	*Control of systems integrating logic, dynamics, and constraints*	1999	*Automatica*	1052
Gibson M.A.; Bruck J.	*Efficient exact stochastic simulation of chemical systems with many species and many channels*	2000	*Journal of Physical Chemistry*	565
Buyya R.; Murshed M.	*GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing*	2002	*Concurrency Computation Practice and Experience*	498
Shan Z.W.; Mishra R.K.; Syed Asif S.A;. Warren O.L.; Minor A.M.	*Mechanical annealing and source-limited deformation in submicrometre- diameter Nicrystals*	2008	*Nature Materials*	264
Jensen K.; Kristensen L.M.; Wells L.	*Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*	2007	*International Journal on Software Tools for Technology Transfer*	264
Ree R.H.; Smith S.A.	*Maximum likelihood inference of geographic range evolution by dispersal, local extinction , and cladogenesis*	2008	*Systematic Biology*	243
Anisimova M.; Nielsen R.; Yang Z.	*Effect of recombination on the accuracy of the likelihood method for detecting positive selection at amino acid sites*	2003	*Genetics*	236
Jun J.B.; Jacobson S.H.; Swisher J.R.	*Application of discrete-event simulation in health care clinics: A survey*	1999	*Journal of the Operational Research Society*	227
Hiskens I.A.; Pai M.A	*Trajectory sensitivity analysis of hybrid systems*	2000	*IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*	216
Wang X.; Merzenich M.M.; Beitel R.; Schreiner C.E.	*Representation of a species-specific vocalization in the primary auditory cortex of the common marmoset: Temporal and spectral characteristics*	1995	*Journal of Neurophysiology*	207

FONTE: Elaborado pelo autor

APÊNDICE B - RELAÇÃO DOS 19 ARTIGOS DE PERIÓDICOS RELACIONADOS A DES E TRÁFEGO URBANO

Continua

Autores	Título	Ano	Periódico	Citações
Vandebona Upali, Richardson Anthony J.	*Simulation of Transit Rout Operations*	1985	*Transportation Research Record*	3
Lu Kang Hsin, Jones Jeff, Kerridge Jon	*Modelling saturated traffic networks using massively parallel computing techniques*	1994	*Traffic Engineering and Control*	1
Di Febbraro A., Recagno V., Sacone S.	*INTRANET: A new simulation tool for intermodal transportation systems*	1996	*Simulation Practice and Theory*	2
Lee J.-K., Lee J.-J.	*Discrete event modeling and simulation for flow control in an automated highway system*	1997	*Transportation Research Part C: Emerging Technologies*	2
Le H., Young W.	*Modelling shopping centre traffic movement (1): Model validation*	1998	*Transportation Planning and Technology*	4
Davidson Alejandra, Wainer Gabriel	*Specifying truck movement in traffic models using cell-DEVS*	2000	*Proceedings of the IEEE Annual Simulation Symposium*	0
Sadoun B.	*Optimizing the operation of a toll plaza system using simulation: A methodology*	2005	*Simulation*	5
Vazquez-Abad F.J., Zubieta L.	*Ghost simulation model for the optimization of an urban subway system*	2005	*Discrete Event Dynamic Systems: Theory and Applications*	1
Stella F., Vigano V., Bogni D., Benzoni M.	*An integrated forecasting and regularization framework for light rail transit systems*	2006	*Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*	2
Dotoli M., Fanti M.P.	*An urban traffic network model via coloured timed Petri nets*	2006	*Control Engineering Practice*	38
Lu Y.-J., Dai H.-P.	*Hybrid Petri net modeling for urban traffic network*	2007	*(Engineering Science)*	5

Conclusão

Autores	Título	Ano	Periódico	Citações
Timm B.W., Kamat V.R.	*General-purpose construction simulation and visualization tools for modelling and animating urban vehicular traffic operations*	2008	*Electronic Journal of Information Technology in Construction*	3
Huang Y.-S., Su P.-J.	*Modelling and analysis of traffic light control systems*	2009	*IET Control Theory and Applications*	5
Correia G., Viegas J.M.	*A conceptual model for carpooling systems simulation*	2009	*Journal of Simulation*	4
Wainer G., Liu Q.	*Tools for graphical specification and visualization of DEVS models*	2009	*Simulation*	6
Lopez-Neri E.	*Hiearachical agent-based modelling: A Guadalajara city case study on urban traffic simulation*	2009	*Transport and Telecommunication*	0
Mahla I., Ovalle R.	*A new hybrid dynamic metropolitan train model [Un nuevo modelo dinámico híbrido de tren metropolitano]*	2010	*Ingeniare*	0
Basile F. Chiacchio P., Teta D.	*A hybrid model for real time simulation of urban traffic*	2012	*Control Engineering Practice*	2
Ahmane M., Abbas-Turki A., Perronnet F., Wu J., El Moudni A., Buisson J., Zeo R.	*Modeling and controlling an isolated urban intersection based on cooperative vehicles*	2013	*Transportation Research Part C: Emerging Technologies*	0

FONTE: Elaborado pelo autor

APÊNDICE C - RELAÇÃO DAS 5 PUBLICAÇÕES DE PERIÓDICOS MAIS RELEVANTES PARA A PESQUISA APONTADOS PELA BASE SCOPUS.

Autores	Título	Ano	Revista	Citações
Lopez-Neri E.	*Hiearachical agent-based modelling: A Guadalajara city case study on urban traffic simulation*	2009	*Transport and Telecommunication*	0
Timm B.W., Kamat V.R.	*General-purpose construction simulation and visualization tools for modelling and animating urban vehicular traffic operations*	2008	*Electronic Journal of Information Technology in Construction*	3
Basile F., Chiacchio P., Teta D.	*A hybrid model for real time simulation of urban traffic*	2012	*Control Engineering Practice*	2
Dotoli M., Fanti M.P.	*An urban traffic network model via coloured timed Petri nets*	2006	*Control Engineering Practice*	38
Huang Y.-S., Su P.-J.	*Modelling and analysis of traffic light control systems*	2009	*IET Control Theory and Applications*	5

FONTE: Elaborado pelo autor

APÊNDICE D - ARTIGOS QUE COMPÕEM O NÚCLEO DE PARTIDA PARA A PESQUISA BIBLIOGRÁFICA

Continua

Autores	Título	Ano	Revista	Observações
Bemporad A.; Morari M.	Control of systems integrating logic, dynamics, and constraints	1999	Automatica	Mais Citado (DES)
Gibson M.A.; Bruck J.	Efficient exact stochastic simulation of chemical systems with many species and many channels	2000	Journal of Physical Chemistry	Mais Citado (DES)
Jun J.B.; Jacobson S.H.; Swisher J.R.	Application of discrete-event simulation in health care clinics: A survey	1999	Journal of the Operational Research Society	Mais Citado (DES)
Hiskens I.A.; Pai M.A	Trajectory sensitivity analysis of hybrid systems	2000	IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications	Mais Citado (DES)
Wainer G., Liu Q.	Tools for graphical specification and visualization of DEVS models	2009	Simulation	Mais Citado (DES + *Transit*)
Huang Y.-S., Su P.-J.	Modelling and analysis of traffic light control systems	2009	IET Control Theory and Applications	Mais Citado (DES + *Transit*)
Vandebona Upali, Richardson Anthony J.	Simulation of Transit Rout Operations	1985	Transportation Research Record	Mais Antigo (DES + *Transit*)
Lu Kang Hsin, Jones Jeff, Kerridge Jon	Modelling saturated traffic networks using massively parallel computing techniques	1994	Traffic Engineering and Control	Mais Antigo (DES + *Transit*)
Di Febbraro A., Recagno V., Sacone S.	INTRANET: A new simulation tool for intermodal transportation systems	1996	Simulation Practice and Theory	Mais Antigo (DES + *Transit*)
Le H., Young W.	Modelling shopping centre traffic movement (1): Model validation	1998	Transportation Planning and Technology	Mais Antigo (DES + *Transit*)
Lopez-Neri E.	Hiearachical agent-based modelling: A Guadalajara city case study on urban traffic simulation	2009	Transport and Telecommunication	Maior Relevância DES + *Transit*)

Conclusão

Autores	Título	Ano	Revista	Observações
Timm B.W., Kamat V.R.	General-purpose construction simulation and visualization tools for modelling and animating urban vehicular traffic operations	2008	Electronic Journal of Information Technology in Construction	Maior Relevância (DES + *Transit*)
Basile F., Chiacchio P., Teta D.	A hybrid model for real time simulation of urban traffic	2012	Control Engineering Practice	Maior Relevância (DES + *Transit*)
Dotoli M., Fanti M.P.	An urban traffic network model via coloured timed Petri nets	2006	Control Engineering Practice	Maior Relevância e Mais Citado (DES + *Transit*)
Huang Y.-S., Su P.-J.	Modelling and analysis of traffic light control systems	2009	IET Control Theory and Applications	Maior Relevância (DES + *Transit*)
Ahmane, M., Turki, A.A., Perronnet, F., Wu, J., Moudni, A.E., Buisson, J., Zeo, R.	Modeling and controlling an isolated urban intersection base on cooperative vehicles	2013	Transportation Research	Mais Recente (DES + *Transit*)
Ringhofer, C.	A level set approach to modeling general service rules in supply chains	2010	Communications in Mathematical Sciences	(DES + *Traffic +suplly chain*)
Byrne, P.J., Heavey, C., Ryan, P., Liston, P.	Sustainable supply chain design: Capturing dynamic input factors.	2010	Journal of Simulation	(DES + *Traffic +supply chain*)

FONTE: Elaborado pelo autor

APÊNDICE E - MÓDULOS DO URURAU COM DESCRIÇÃO E DADOS DE CONFIGURAÇÃO

Continua

Módulo	Nome	Descrição	Dados de Configuração
Criar	E4	Responsável pela criação de entidades.	T. Chegadas: Const. (1s) T. Prim. Chegada: 1 Máx. Chegadas: Infinito
Atribuição	C4	Adiciona uma variável ao modelo.	Tipo: Variável Nome: tempo Valor:tempo+1
X (Decisor)	X1	Desvia a execução de uma função.	Tipo: RNA com 2 caminhos Arquivo de Treinamento: C:\DadosTreinamentoRNA5Neuronios_ Modificado.CSV Algoritmo: ResilientPropagation Exp. Neurônio 1: L1.Segurar1.queue Exp. Neurônio 2: L2.Segurar2.queue Exp. Neurônio 3: L3.Segurar3.queue Exp. Neurônio 4: tempo Exp. Neurônio 5: variavelsemaforo Nº Neurônios Camada Entrada: 5 Nº Neurônios Camada Oculta:4 Nº Neurônios Camada Saída:1
X (Decisor)	X2	Desvia a execução de uma função.	Tipo: 2 caminhos por condição Condição: variavelsemaforo ==0
X (Decisor)	X3	Desvia a execução de uma função.	Tipo: 2 caminhos por condição Condição: variavelsemaforo ==1
Escrever	W4	Escreve em arquivo valores da de uma variável	Nome da variável: tempo Arquivo: semaforo1.txt
Escrever	W5	Escreve em arquivo valores da de uma variável	Nome da variável: tempo Arquivo: semaforo2.txt
Atribuição	C5	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua1 Valor:0
Atribuição	C6	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua2 Valor:0
Atribuição	C7	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelpessoa Valor:0

Conclusão

Módulo	Nome	Descrição	Dados de Configuração
Atribuição	C8	Adiciona uma variável ao modelo.	Tipo: Variável Nome: tempo Valor:0
Atribuição	C9	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelsemaforo Valor:0
Escrever	W6	Escreve em arquivo valores da de uma variável	Nome da variável: tempo Arquivo: semaforo3.txt
Atribuição	C10	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua1 Valor:0
Atribuição	C11	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelcarrorua2 Valor:0
Atribuição	C12	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelpessoa Valor:0
Atribuição	C13	Adiciona uma variável ao modelo.	Tipo: Variável Nome: tempo Valor:0
Atribuição	C14	Adiciona uma variável ao modelo.	Tipo: Variável Nome: variavelsemaforo Valor:1
Terminar	-	Termina a lista de comandos	-

FONTE: Elaborado pelo autor

APÊNDICE F – ARTIGO APRESENTADO NO IX ENCONTRO MINEIRO DE ENGENHARIA DE PRODUÇÃO - EMEPRO 2013

F.1.MODELO DE SIMULAÇÃO PARA ANÁLISE DO FLUXO DE VEÍCULOS EM VIA URBANA

Resumo

O objetivo deste trabalho é descrever um modelo de simulação para análise do fluxo de veículos em um cruzamento de trânsito semaforizado. Este modelo considera o tempo de abertura e fechamento dos semáforos compreendidos entre as vias que compõem o sistema. O modelo foi implementado em ARENA® para a execução da simulação. Os resultados mostraram que o sistema pode ser considerado análogo ao sistema em linhas de produção industrial. Com isso, pôde-se criar um ambiente simulado a partir de uma grande quantidade de dados reais sem a necessidade de interferir em ambientes industriais.

Palavras-chave: Cruzamento; Semáforos; Linhas de Produção, Simulação a Eventos Discretos.

1. Introdução

Em recente trabalho, Baptista e Rangel (2013) analisaram o desempenho de um sistema de controle automático através de um modelo de simulação a eventos discretos. O modelo de simulação, no caso, representava o fluxo de veículos de uma via urbana. Os autores lançaram mão da semelhança entre os fenômenos dinâmicos e estocásticos associados ao fluxo de veículos nos cruzamentos e puderam avaliar o sistema de controle. A análise de tal sistema pôde então representar de forma

análoga o comportamento existente em diversas situações típicas encontradas em linhas de montagens industriais. A principal diferença é que as vias urbanas são públicas e os autores puderam coletar os dados de forma livre.

Tanto no trânsito como na indústria, a formação de gargalos e filas tem sido comum. Guidorizzi *et al.* (2009) afirmam que esses problemas são similares em ambas as áreas, e que desta forma, é possível fazer uma analogia direta para o problema, para as variáveis e também para as soluções.

Helbing (2003) apresenta informações em que relaciona conceitos das redes de abastecimentos das linhas de produção, com tráfego de pedrestres também aplicáveis ao tráfego de veículos. O autor utiliza como analogia o trânsito de pedestres e linha de produção de semicondutores e, ao concluir o trabalho, afirma que a razão para as semelhanças entre os sistemas de produção e de tráfego é a presença de entidades dinâmicas (pessoas, objetos), que interagem de um modo não linear, e a existência da competição por recursos limitados (como capacidade, tempo ou espaço).

Nagatani e Helbing (2008) apresentam um modelo matemático para cadeia de suprimentos envolvendo políticas de ordens de entrega com base em níveis de estoque e o relaciona com modelo macroscópico de trânsito.

O estudo de Ringhofer (2010) também se utiliza de analogias entre cadeias e redes de suprimentos e os sistemas de fluxo de tráfego. De acordo com o autor a analogia se baseia no fato de que as peças passando pela cadeia são comparadas aos veículos que viajam numa estrada virtual (a fase do processo), em que entram como matéria prima e deixam a cadeia de suprimentos como produtos acabados.

Considerando os aspectos da simulação, utilizando-se a analogia descrita por Mugnela e Netto (2012), o segmento de rua (link) se assemelha a uma esteira, movimentando todos os carros com igual velocidade do começo ao fim do segmento, onde são empilhados. O cruzamento (nó), que pode conter um semáforo ou não, é responsável por distribuir os carros dos segmentos, e o carro é a partícula básica de carga do sistema.

Com base nos trabalhos de Ringhofer (2010), Nagatani e Helbing (2008) e Helbing (2003), é possível afirmar que o trânsito possui padrões de comportamentos

que poderiam ser estudados para serem empregados em linhas de produção dos sistemas de manufatura.

Além disso, é possível verificar que a simulação a eventos discretos é aplicável em ambos os casos com o objetivo de se obter resultados e realizar análises de forma mais acessível, segura e dinâmica.

Desta forma, o objetivo deste trabalho é utilizar um cruzamento semaforizado do trânsito como fonte de dados para simulação e identificação de possíveis gargalos, análogos aos existentes também nas linhas de produção. Para tal, foi escolhido um cruzamento semaforizado na cidade de Campos dos Goytacazes, RJ, em que foram realizadas diversas medições de tempo e fluxo de veículos a fim de que essas informações fossem utilizadas na concepção e implementação do modelo de simulação.

2. Descrição do Sistema

A Figura 1 apresenta o sistema de trânsito objeto desde trabalho.

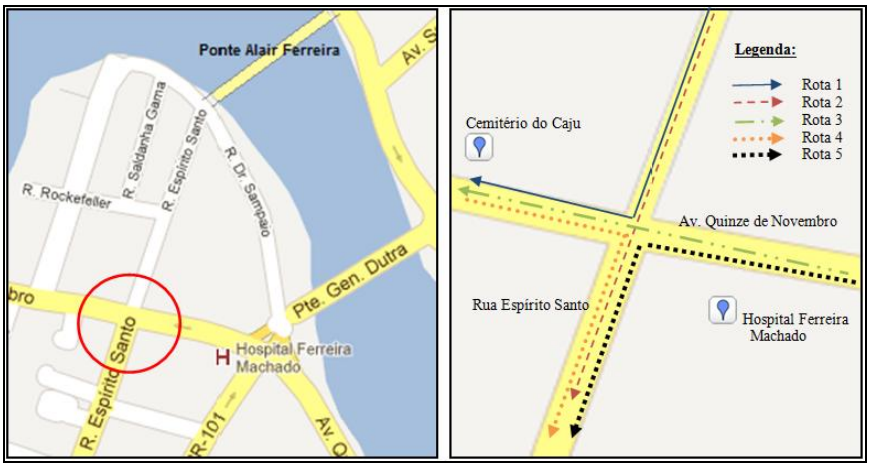

FIGURA 1 – Cruzamento modelado. Fonte: Adaptado do Google Maps.

A intercessão em destaque na Figura 1a foi escolhida para estudo e simulação, por apresentar congestionamentos com frequência. Uma ampliação do

cruzamento em destaque pode ser observada pela Figura 1b, em que são apresentadas cada uma das rotas que compõem o sistema em estudo.

A intercessão é semaforizada e apresenta as seguintes rotas possíveis:

- ROTA 1: Veículos que trafegam pela Av. Quinze de Novembro (Hospital Ferreira Machado sentido Cemitério do Caju) podem dobrar à esquerda na Rua Espírito Santo;
- ROTA 2: Veículos que trafegam pela Av. Quinze de Novembro (Hospital Ferreira Machado sentido Cemitério do Caju) podem seguir em frente na mesma rua;
- ROTA 3: Veículos que trafegam na rua Espírito Santo (Ponte Alair Ferreira sentido Rio de Janeiro) podem seguir à frente;
- ROTA 4: Veículos que trafegam na rua Espírito Santo (Ponte Alair Ferreira sentido Rio de Janeiro) podem dobrar à direita na Av. Quinze de Novembro.
- ROTA 5: Veículos que trafegam pela Av. Quinze de Novembro (Cemitério do Caju sentido Rio de Janeiro) podem dobrar à direita na Rua Espírito Santo.

Ciclo de Temporização dos Semáforos

O tempo do ciclo dos semáforos que compõem o sistema é de 86 segundos.

A Figura 2 apresenta a atual configuração do ciclo semafórico do cruzamento em estudo. Nesta Figura é possível visualizar o sincronismo dos semáforos de forma que as áreas em cores diferentes representam os sinais luminosos emitidos por cada um deles nas cores verde, vermelho e amarelo.

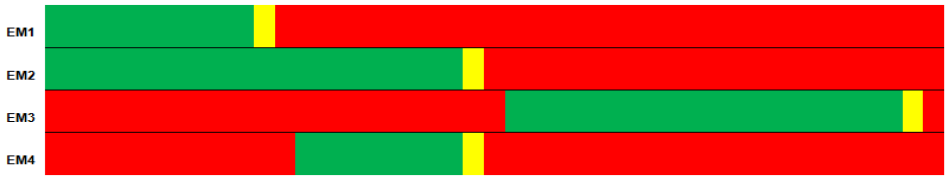

FIGURA 2 – Ciclo semafórico do cruzamento

O SEM1 realiza o controle do tráfego de veículos relacionados à ROTA 1, sendo possível constatar os seguintes tempos: 20 segundos de verde, 2 segundos de amarelo e 64 segundos de vermelho.

O SEM2 controla os veículos da ROTA 2, tendo tempo de verde igual a 40 segundos, tempo de amarelo igual a 2 segundos e tempo de vermelho igual a 44 segundos. Já o SEM3 que realiza o controle das ROTAS 3 e 4, possui tempos de verde, amarelo e vermelho iguais a 38 segundos, 2 segundos e 46 segundos respectivamente.

A ROTA 5 é controlada pelo SEM4 que possui tempo de verde igual a 16 segundos, tempo de amarelo igual a 2 segundos e tempo de vermelho igual a 68 segundos.

Tempo de Chegada de Veículos em cada VIA

As funções que representam os Tempos entre Chegadas (TEC) dos veículos que acessam o sistema composto pelas Vias 1, 2, 3 e 4 podem ser descritas como apresentadas na Tabela 1, tendo sido obtidas por meio de medições realizadas no cruzamento em estudo.

As funções das distribuições de probabilidade encontradas para o modelo apresentaram o *p-value* maior que 0,05, mostrando que são aderentes ao conjunto de valores da amostra.

As medições ocorreram no dia 10 de Janeiro de 2013, no horário compreendido entre às 9h45min e 11h30min, utilizando-se o aplicativo Cronômetro v1.28 instalado em *smartphone* Android. O aplicativo utilizado para coleta dos dados garante precisão na ordem de milésimos de segundos. O tamanho da amostra

coletada em todos os semáforos que compõem o sistema foi de 341 registros. Após a conversão dos registros em arquivos com extensão .CSV (*Comma Separated Values*), os dados foram tratados com o *software Input Analyzer* , onde foi possível verificar as funções de tempo entre chegadas mais aderentes em cada via.

Tabela 1 – Funções de TEC das Vias 1 e 2 .

Via	Rotas	Função de TEC (segundos)
V1	ROTA 1	LOGN(10.1, 14.2) segundos
V2	ROTA 2	1 + WEIB(6.2, 0.831) segundos
V3	ROTA 3 e 4	23 * BETA (0.436, 3.15) segundos
V4	ROTA5	41 * BETA (0.525, 1.47) segundos

Fonte: Elaboração própria com base em dados coletados e tratados.

É importante ressaltar que uma das vias (Via 3) que compõe a intercessão, recebe maior parte do fluxo de veículos oriundos da BR-101 com sentido ao Rio de Janeiro. Esse percurso é o único possível para os veículos que atravessam o Rio Paraíba do Sul utilizando a Ponte Alair Ferreira (principal ponte de acesso à cidade de Campos dos Goytacazes para os veículos que trafegam pela BR-101).

3. Modelo de Simulação

Para construção do modelo foi seguida a metodologia apresentada por Chwif e Medina (2010), composta por três grandes etapas: concepção ou formulação do modelo, implementação, e análise dos resultados. Para tal, observaram-se as advertências descritas por Banks e Chwif (2010).

Além disso, para a verificação e validação, foi seguida adicionalmente a metodologia proposta por Sargent (2010). O modelo foi testado apenas após a checagem completa da validade do mesmo.

O modelo conceitual foi elaborado utilizando a linguagem IDEF-SIM (MONTEVECHI, *et al.* 2010). A Figura 3 apresenta cada uma das vias que compõem o cruzamento bem como os semáforos relacionados às mesmas, e as rotas de saída

para cada uma delas. A Tabela com os códigos, significados e parâmetros correspondentes a Figura 3 encontra-se no Apêndice A.

A Via 1 recebe o fluxo de veículos que trafegam pela Av. Quinze de Novembro (Hospital Ferreira Machado sentido Cemitério do Caju) e podem dobrar à esquerda na Rua Espírito Santo (Rota 1).

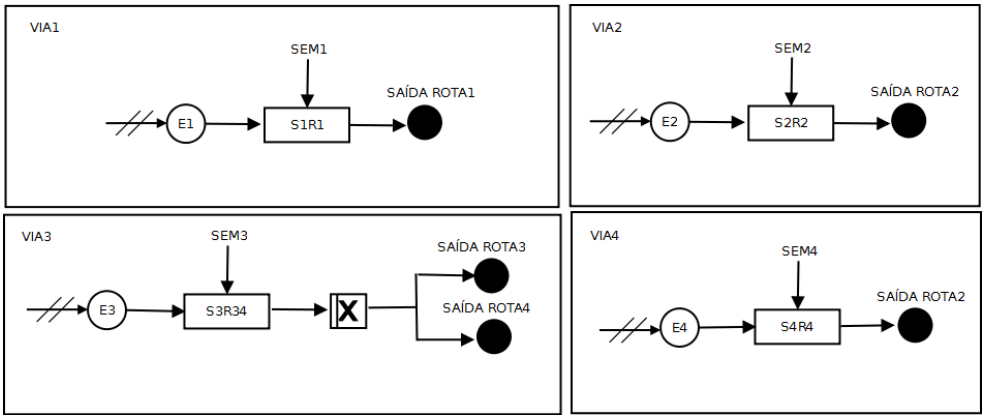

Figura 3 – Modelo conceitual das Vias que compõem o cruzamento em estudo.

Já na Via 2 trafegam os veículos que podem seguir a frente na pela Av. Quinze de Novembro sentido Cemitério do Caju (Rota 2).

A Via 3 apresenta uma especificidade para o sistema, uma vez que os veículos que trafegam pela rua Espírito Santo descendo da Ponte Alair Ferreira podem seguir à frente sentido Rio de Janeiro (Rota 3), ou dobrar à direita na Av. Quinze de Novembro (Rota 4). A Via 4 é por onde trafegam os veículos fazem o trajeto da Av. Quinze de Novembro dobrando à direita na Rua Espírito Santo, sentido Rio de Janeiro (Rota 5).

A Figura 4 representa o modelo conceitual da transição de estados dos quatro semáforos do cruzamento modelado dentro do ciclo semafórico. A Tabela com os códigos, significados e parâmetros correspondentes a Figura 4 encontra-se no Apêndice B.

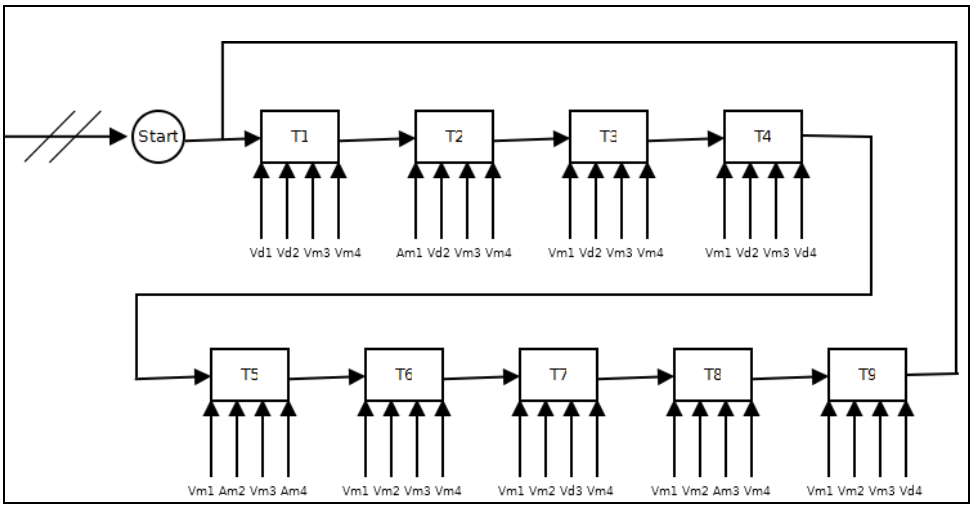

Figura 4 – Modelo conceitual das transições de estados dos semáforos.

A Figura 5 ilustra as 9 transições semafóricas apresentadas na Figura 2 considerando o esquema de cores dos sinais de cada semáforo.

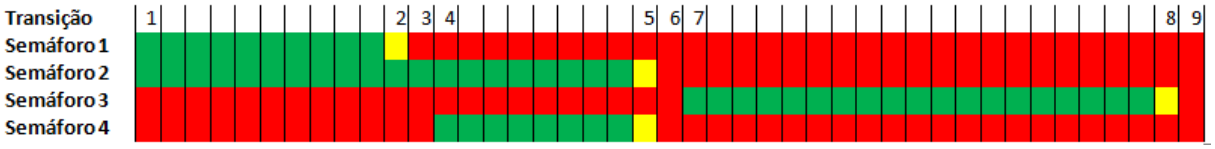

Figura 5 – Representação das transições dos estados dos semáforos.

4. Experimentos

Esta etapa foi realizada utilizando um computador com processador Intel® core i3 2.53 GHz, com 4 GB de memória RAM e sistema operacional Windows 7 Home Basic de 64 Bits. O software de simulação utilizado foi o ARENA® *Student* versão 14.

As configurações utilizadas para simulação foram: 8 replicações com 600 segundos de tamanho de replicação.

A atual configuração semafórica do cruzamento estudado resulta numa situação de retenção da Via 3 em horários de maior fluxo de veículos. Esta retenção

pode ser percebida na Tabela 2 que apresenta as quantidades médias de veículos em cada via.

A retenção da Via 3 fica mais evidente quando são observados os valores de tempo médio de espera nas filas das Vias 1, 2, 3 e 4, respectivamente, 15,05s, 3,14s, 110,16s e 11,86s. Como pode ser percebido, os veículos levam de 7 a 35 vezes mais tempo parados na Via 3 em relação as demais vias. Esta situação de retenção desproporcional causa congestionamentos e trânsito lento na rodovia BR 101.

Tabela 2: Quantidade e tempo médios de veículos em cada via.

	Via 1	**Via 2**	**Via 3**	**Via 4**
Quantidade média de veículos na fila	1,96	0,39	34,94	1,18
Tempo médio na fila	15,05 s	3,14 s	110,16 s	11,86 s

Fonte: Elaboração própria com base na simulação do modelo.

Com o objetivo de buscar soluções para o problema de formação de filas com grande taxa de retenção dos veículos na Via 3, foram realizados dois experimentos de simulação. Para tal, a configuração semafórica passou por modificações possíveis e que poderiam resolver ou amenizar os problemas.

4.1. Experimento 1

No experimento 1 o tempo do ciclo semafórico se manteve o mesmo (86 segundos), porém, os tempos de verde dos semáforos foram modificados. O tempo de verde do semáforo da Via 3 foi aumentado, conforme dados da Tabela 3.

Tabela 3: Ciclo dos semáforos 1, 2, 3 e 4 no experimento 1.

	Tempo de Verde	Tempo de Amarelo	Tempo de Vermelho
Semáforo 1	18 s	2 s	66 s
Semáforo 2	38 s	2 s	42 s
Semáforo 3	40 s	2 s	44 s
Semáforo 4	16 s	2 s	68 s

Fonte: Elaboração própria.

4.2 - Experimento 2

No experimento 2 optou-se por aumentar o tempo de ciclo de 86 segundos para 120 segundos e além disso, aumentar o tempo de verde da Via 3 de 38 segundos para 72 segundos, conforme a Tabela 4.

Tabela 4: Ciclo dos semáforos 1, 2, 3 e 4 no experimento 2.

	Tempo de Verde	Tempo de Amarelo	Tempo de Vermelho
Semáforo 1	20 s	2 s	98 s
Semáforo 2	40 s	2 s	78 s
Semáforo 3	72 s	2 s	46 s
Semáforo 4	16 s	2 s	102 s

Fonte: Elaboração própria.

5. Discussão dos Resultados

É possível observar que o tempo médio de veículos na fila da Via 3 atualmente é de 110,16 segundos, porém este tempo foi reduzido nos dois experimentos, passando para 102,66 segundos no experimento 1 em que o ciclo semafórico foi mantido. Já com experimento 2, foi possível observar uma redução de quase 50% do tempo de espera na fila da Via 3 (52,32 segundos), embora haja um aumento significativo desse tempo nas demais vias.

Tabela 5: Tempo médio na fila: situação atual e experimentos 1 e 2.

	Tempo médio na fila			
	Via 1	**Via 2**	**Via 3**	**Via 4**
Situação Atual	15,05 s	3,14 s	110,16 s	11,86 s
Experimento 1	36,67 s	3,71 s	102,66 s	5,26 s
Experimento 2	48,10 s	14,07 s	52,32 s	39, 40 s

Fonte: Elaboração própria.

Por outro lado, sabe-se que a Via 3 é a que apresenta maior fluxo de veículos pois recebe a carga de tráfego da BR 101, e uma redução no tempo de espera na fila nesta via, torna o sistema mais dinâmico e diminui a formação de grandes congestionamento (filas), inclusive sobre a ponte Alair Ferreira. Esta afirmação pode ser percebida através dos dados contidos na Tabela 6, em que é possível observar que com o aumento do tempo de ciclo dos semáforos do sistema, a quantidade média de veículos na Via 3 é reduzida em mais de 40%, embora verifique-se um aumento médio desse número nas demais vias.

Tabela 6: Quantidade média de veículos na fila: situação atual, experimentos 1 e 2.

	Quantidade média de veículos na fila			
	Via 1	**Via 2**	**Via 3**	**Via 4**
Situação Atual	1,96	0,39	34,94	1,18
Experimento 1	4,22	0,49	35,06	0,47
Experimento 2	6,27	1,90	18,40	4,57

Fonte: Elaboração própria.

O aumento médio de veículos na fila das Vias 1, 2 e 4 (experimento 2), não se demonstra como um problema, já que é possível observar que essa quantidade está entre 1 e 7 veículos, o que na prática não remete a uma situação de congestionamento, diferentemente da situação da Via 3 que atualmente apresenta quantidade média de 35 veículos na fila (neste caso a fila de veículos pode chegar a se concentrar sobre a ponte que dá acesso a via). Diminuir a quantidade média de

veículos na fila da Via 3 de 35 para aproximadamente 19 veículos, possui grande utilidade para o sistema como um todo.

Analogamente a situação verificada pelos experimentos e discutida aqui, na indústria pode-se notar comportamentos semelhantes. RINGHOFER (2010) fala da importância da definição de políticas de escalonamento para as cadeias de suprimento, uma vez que nem todas as partes da cadeia possuem a mesma prioridade (devido questões como data de entrega, partes perecíveis, entre outras), necessitando-se priorizar determinados fluxos da cadeia, em detrimento a outros não tão importantes em um determinado momento.

Priorizando-se a Via 3 por considerar que é a via que apresenta problemas de congestionamento por receber a maior concentração de veículos entre as vias do sistema modelado, foi possível observar que com aumento do ciclo semafórico no experimento 2 para 120 segundos, e tempo de verde do semáforo 3 para 72 segundos houve uma diminuição considerável na fila média.

6. Conclusões

A analogia entre os sistemas de tráfego urbano e as linhas de produção tem norteado diversos estudos atualmente. Por ser um ambiente aberto e público, o trânsito apresenta características favoráveis à coleta de dados para serem utilizados em simulação.

Os experimentos realizados com base nos dados coletados em um cruzamento urbano da cidade de Campos dos Goytacazes, permitiram verificar que é possível utilizar esses dados para simular possíveis melhorias e resolver problemas como a formação de filas, gargalos e também para priorizar fluxos.

Problemas como os descritos são análogos aos sistemas de produção das mais diversas áreas. O trânsito sob essa ótica apresenta uma vasta oportunidade de variáveis que podem resultar em estudos mais aprofundados nesse assunto.

Agradecimentos

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq e à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ pelo suporte financeiro para esta pesquisa.

Referências

BAPTISTA, R. C. T. ; RANGEL, J. J. A. . Simulação a Eventos Discretos de uma Via Semaforizada com Controle Automatizado em Tempo Real. Revista Produção Online, v. 13, p. 290-317, 2013.

BANKS, J.; CHWIF, L. Warnings about simulation. *Journal of Simulation* .1-13.2010.

CHWIF, L.; MEDINA, A. C.; Modelagem e Simulação de Eventos Discretos: Teorias & Aplicações. 3ª edição. São Paulo. Edição do Autor. 2010.

GUIDORIZZI, M. DOS S.; SANTOS, A.P. B; OLIVEIRA, A.B.; LEONARDI, F. *Simulação como apoio a tomada de decisão para a solução de problemas causados pelos gargalos formados no trânsito.* A Engenharia de Produção e o Desenvolvimento Sustentável: Integrando Tecnologia e Gestão. XXIX ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO. Salvador, BA, Brasil: 09 de outubro de. 2009.

HELBING, D. *Modelling and Optimization of production processes:* lessons fron traffic dynamics. Working Paper. Out.2003. Retirado de: http://www.santafe.edu/research/working-papers/abstract/dacaed19ae05872fce780032a1f12456/ em janeiro 2012.

NAGATANI, T., HELBING, D.. Stability Analyis and stabilization strategies por linear supply chains. *Elsivier Science*. 2nd February 2008.

MONTEVECHI, J.A.B.; LEAL, F.; DE PINHO, A.F.; DA SILVA, R.F.C.; DE OLIVEIRA, M.L.M.; DA SILVA, A.L.F. *Conceptual modeling in simulation projects by mean*

*adapted IDEF: An application in a Brazilian tech company***.** In: Winter Simulation Conference, 2010, Baltimore. Proceedings of the 2010 Winter Simulation Conference, 2010. p. 1624-1635, 2010

MUGNELA, B. S.; NETTO, M. L. *GenPolis - Prototipagem e aplicação de uma ferramenta especializada para otimização via algoritmos genéticos de planos fixos de sinalização semafórica em sub-redes urbanas***.** Cidades Inteligentes. VIII SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO. São Paulo - SP - Brasil: 18 de maio de 2012.

RINGHOFER, C. A level set approach to modeling general service rules in supply chains *Communications in Mathematical Sciences*. 909-930. 2010.

SARGENT, R.G.; *Verification and Validation of Simulation Models*. In: Winter Simulation Conference, B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds. Baltimore – EUA. 166-183. 2010.

APÊNDICE A - Dados dos Modelos referentes a Figura 3.

Código	Significado	Parâmetro
E1	Veículos oriundos da VIA 1.	LOGN (10.1, 14.2) s; Início: 0; 1 por vez; infinito.
E2	Veículos oriundos da VIA 2.	1 + WEIB (6.2, 0.831) s; Início: 0; 1 por vez; infinito.
E3	Veículos oriundos da VIA 3.	23 * BETA (0.436, 3.15) s; Início: 0; 1 por vez; infinito.
E4	Veículos oriundos da VIA 4.	41 * BETA (0.525, 1.47) s; Início: 0; 1 por vez; infinito.
S1R1	Bloquear veículos que trafegam pela VIA1, quando o sinal vermelho de SEM1 se encontra ativo para veículos que viram à esquerda.	Variável de acordo com o Semáforo1.
S2R2	Bloquear veículos que trafegam pela VIA2, quando o sinal vermelho do SEM2 se encontra ativo para os veículos que seguem em frente na mesma rua.	Variável de acordo com o Semáforo2.
S3R34	Bloquear veículos que trafegam pela VIA3, quando o sinal vermelho do SEM3 se encontra ativo para os veículos que seguem em frente na mesma rua ou viram à direita na Av. Quinze de Novembro.	Variável de acordo com o Semáforo3.
S4R5	Bloquear veículos que trafegam pela VIA4, quando o sinal vermelho de SEM4 se encontra ativo.	Variável de acordo com o Semáforo4.
SEM1	Semáforo da VIA1	Verm.: 64 seg.; Verde: 20 seg.; Amar.: 2 seg.
SEM2	Semáforo da VIA2	Verde: 40 seg.; Amar.: 2 seg.; Verm.: 44 seg.
SEM3	Semáforo da VIA3	Verde:38 seg.; Amar.: 2 seg.; Verm.: 46 seg.
SEM4	Semáforo da VIA4	Verm.: 68 seg.; Verde: 46 seg.; Amar.: 2 seg.

APÊNDICE B - Dados do Modelo referentes a Figura 4.

Código	Significado	Parâmetro
Start	Entidade auxiliar que indica o início do ciclo semafórico	Início: 1; Máximo: 1
T1	Primeira transição do ciclo. Abertura (verde) dos semáforos 1 e 2.	Duração: 20 segundos.
T2	Segunda transição do ciclo. Indicação de atenção (amarelo) no semáforo 1.	Duração: 2 segundos.
T3	Terceira transição do ciclo. Fechamento (vermelho) do semáforo 1.	Duração: 2 segundos.
T4	Quarta transição do ciclo. Abertura (verde) do semáforo 4.	Duração: 16 segundos.
T5	Quinta transição do ciclo. Indicação de atenção (amarelo) nos semáforos 2 e 4.	Duração: 2 segundos.
T6	Sexta transição do ciclo. Vermelho de segurança.	Duração: 2 segundos.
T7	Sétima transição do ciclo. Abertura (verde) do semáforo 3.	Duração: 38 segundos.
T8	Oitava transição do ciclo. Indicação de atenção (amarelo) no semáforo 3.	Duração: 2 segundos.
T9	Nona transição do ciclo. Vermelho de segurança.	Duração: 2 segundos.
Vd1	Sinal verde do semáforo 1.	Capacidade:1.
Vm1	Sinal vermelho do semáforo 1.	Capacidade:1.
Am1	Sinal amarelo do semáforo 1.	Capacidade:1.
Vd2	Sinal verde do semáforo 2.	Capacidade:1.
Vm2	Sinal vermelho do semáforo 2.	Capacidade:1.
Am2	Sinal amarelo do semáforo 2.	Capacidade:1.
Vd3	Sinal verde do semáforo 3.	Capacidade:1.
Vm3	Sinal vermelho do semáforo 3	Capacidade:1.
Am3	Sinal amarelo do semáforo 3.	Capacidade:1.
Vd4	Sinal verde do semáforo 4.	Capacidade:1.
Vm4	Sinal vermelho do semáforo 4.	Capacidade:1.
Am4	Sinal amarelo do semáforo 4.	Capacidade:1.

APÊNDICE G – ARTIGO PUBLICADO NA REVISTA PESQUISA OPERACIONAL PARA O DESENVOLVIMENTO EM MAIO DE 2014 – PODES V. 6, N. 2 (2014)

G.1.DECISÃO COM REDES NEURAIS ARTIFICIAIS EM MODELOS DE SIMULAÇÃO A EVENTOS DISCRETOS

Resumo

O objetivo deste trabalho é avaliar *frameworks* para aplicação de redes neurais artificiais acoplados a modelos de simulação a eventos discretos com o software Ururau. A análise está centrada em encontrar mecanismos que possibilitem a realização de decisões através de outros algoritmos integrados ao código do modelo de simulação. Na literatura atual, há pouca citação de como aplicar algum tipo de inteligência computacional a partes do código de um modelo, que necessite refletir comportamentos de ações com aspectos de maior complexidade, como ações de pessoas em uma parte do processo. Por outro lado, muitos software*s* de simulação permitem a customização dos modelos através de algoritmos que podem ser integrados aos modelos. Os resultados encontrados demonstraram que a utilização do *framework* Encog se mostrou adequado para a criação das redes neurais artificiais, sem apresentar erros de funcionamento e incompatibilidades com o código do software Ururau. Além disso, o trabalho permite que as pessoas interessadas em compreender esta estrutura possam visualizá-la diretamente no código do modelo proposto.

Palavras-chave: Ururau, Redes Neurais Artificiais, Encog, Decisão.

Abstract

The objective of this study is to evaluate frameworks for the application of artificial neural networks linked to models of discrete event simulation with the Ururau software. The research is focused on finding mechanisms to allow the implementation of decisions by other algorithms integrated into the code of the

simulation model. In the current literature, there is little mention of how to apply some kind of computational intelligence to parts of the code of a model that needs to reflect behaviors of actions with aspects of higher complexity, as actions of people in a part of the process. On the other hand, many simulation softwares allow customization of the models through algorithms that can be integrated into the models. The results showed that the use of the Encog framework is adequate for the creation of artificial neural networks, without presenting operating errors and incompatibilities with the software code Ururau. Moreover, the study allows people interested in understanding this structure visualizing it directly in the code of the proposed model.

Key words: Ururau, Artificial Neural Networks, Encog, Decision.

1. Introdução

Os softwares utilizados para construção de modelos de simulação a eventos discretos (SED), de uma forma geral, representam as decisões realizadas nos sistemas sob análise com regras básicas de operadores lógicos. Estes operadores realizam funções, como: “<” (menor que), “>” (maior que), “==” (igual a), “E”, “OU” e suas combinações. Ou ainda, em outros casos mais simples, utilizam apenas porcentagens obtidas em dados históricos para representar as tendências realizadas pelo conjunto das decisões tomadas. De certa forma, alguns aspectos dos sistemas que configuram algumas decisões mais sofisticadas, tomadas por pessoas, por exemplo, são representadas de forma simplificada nestes modelos. Assim, a possibilidade de representar tais ações de maneira mais realística e "inteligente" é um desafio que se apresenta e tem motivado pesquisadores como Robinson *et. al* (2001) e Bergmann *et al.* (2014).

Segundo Swain (2007), quase 71% dos softwares utilizados para a construção de modelos de SED permitem a customização dos modelos através de algoritmos provenientes de outras linguagens ou módulos externos. No trabalho de Bergmann *et al.* (2014), foram utilizadas redes neurais artificiais (RNA) em conjunto com modelos de simulação. Neste caso, a geração dos modelos pode ocorrer de forma automática, e se baseia na aquisição de dados armazenados em sistemas de ERP (*Enterprise Resource Planning*). Assim, uma dificuldade encontrada nesta tarefa é a representação do comportamento dinâmico nos sistemas de manufatura, por exemplo. Diversas informações sobre comportamentos, como estratégias de

controle, não são comumente armazenadas pelos sistemas de TI. Neste caso, é preciso a utilização de algoritmos específicos para extrair comportamentos complexos, e, muitas vezes, é necessário um especialista no sistema para adicionar o comportamento detalhado ao modelo que foi gerado automaticamente.

Segundo Zuben (2003), o comportamento humano pode ser emulado por uma RNA a partir de amostras de entradas e saídas de um sistema. Assim, o vetor de entrada contém o conjunto de informações recebidas por uma pessoa, por exemplo, e o vetor de saída, as respectivas ações tomadas por ele com base nas informações recebidas. Em outro trabalho sobre o assunto, Silva *et al.* (2012) utilizaram RNA para representar o comportamento de decisões mais sofisticadas, ou seja, com maior nível de detalhes, normalmente tomadas por pessoas em modelos de SED. Os autores realizaram os testes utilizando o software Ururau (Peixoto *et al.,* 2013). Neste caso, foi criado um módulo com um algoritmo de uma RNA que se comunicava através de soquetes TCP (*Transmission Control Protocol),* com o modelo de simulação, capaz de representar de forma mais realística as decisões realizadas por pessoas nos sistemas modelados.

Diante do que foi apresentado, o objetivo deste trabalho foi buscar um mecanismo para incorporar ao software Ururau um módulo capaz de executar uma RNA diretamente no código de um modelo de simulação. A utilização deste módulo pode facilitar a elaboração de modelos de simulação e evitar a comunicação dos mesmos com algoritmos executados externamente. Desta forma, podem-se representar partes de processos modelados, onde a decisão é tomada com base em fatores diferentes dos operadores lógicos ou porcentagens, que tenham a ver com conhecimentos e experiências prévias do decisor.

Para isso, foram pesquisados e avaliados *frameworks* para aplicação de RNA em modelos de SED. Isso permitiu a realização de um estudo mais aprofundado sobre recursos e tecnologias já consolidadas. A princípio, o que se buscou, ao realizar a pesquisa, foi encontrar algum *framework* em Java puro que permitisse a criação, treinamento e execução de uma RNA. A necessidade de ser um *framework* Java se deu pelo fato do software Ururau ter sido desenvolvido originalmente nesta linguagem.

2. Referencial Teórico

2.1 O Ururau

O Ururau é um software de simulação de código aberto e livre de custos, que utiliza como base a biblioteca de simulação JSL (*Java Simulation Library)* (Rosseti, 2008). O software permite a construção de modelos de simulação tanto em interface gráfica (GUI - *Graphic User Interface*) quanto em uma API - *Application Programming Interface* (Peixoto *et al.,* 2013).

A Figura 1 apresenta a arquitetura do Ururau. Observe que a linguagem Java permeia todas as camadas que o compõe. Já, a biblioteca JSL, que está na camada mais inferior, converte o modelo para uma sequência de eventos discretos.

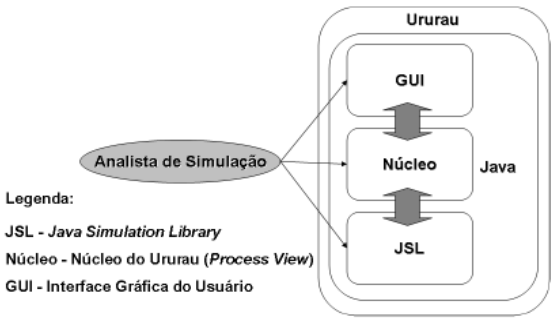

Figura 1: Arquitetura do Ururau.
Fonte: Peixoto, *et al*.(2013).

O núcleo do Ururau está na camada intermediária, sendo composto por comandos de processos específicos do JSL. A camada mais superior trata da conversão do modelo gráfico, que é composto por um grafo dirigido para uma sequência de comandos do núcleo do software. Observe que o analista de simulação pode atuar em qualquer camada do software de acordo com a necessidade observada no momento de elaboração do modelo.

Silva *et al.* (2012) realizaram uma extensão ao Ururau partindo de suas camadas mais baixas (JSL e Núcleo) de forma a possibilitar a comunicação deste com o módulo inteligente. Para testar este mecanismo, os autores desenvolveram

uma comunicação entre o Ururau e o módulo inteligente baseada em soquetes TCP, conforme ilustra a Figura 2.

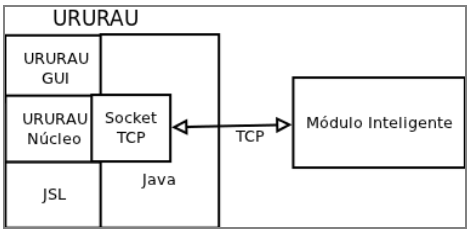

Figura 2: Forma de comunicação do módulo inteligente com o modelo.
Fonte: Silva *et al.* (2012).

Na abordagem proposta por Silva *et al.* (2012), o modelo de simulação foi criado no Ururau a partir de linhas de código, e este, quando executado, enviava os dados relacionados ao processo decisório para o módulo inteligente. Ao receber os dados, o módulo inteligente executava uma RNA, previamente treinada, para retornar a decisão a ser executada pelo modelo de simulação.

2.2 Frameworks de Redes Neurais Artificiais em Java

De acordo com Baptista & Dias (2012), a escolha da solução mais conveniente para a aplicação envolvendo RNA pode ser uma tarefa complexa. A avaliação deve considerar a arquitetura da RNA, o algoritmo de treinamento, o sistema operacional, e o tipo de licença do software. Desta forma, três dos principais *frameworks* não comerciais foram pesquisados. São eles: JOONE, NEUROPH, e Encog.

2.2.1 JOONE

O JOONE (Java Object Oriented Neural Engine) é um framework escrito em linguagem Java para a construção e execução de aplicações de Inteligência Artificial baseadas em RNA. Pode ser executado em qualquer plataforma e é compatível com vários sistemas operacionais como: Linux, Mac OSX, Windows 2000, Windows XP e SUN Solaris (MARRONE, 2007).

O software é composto por um motor central, um editor de GUI e um ambiente de treinamento distribuído (JOONE, 2013). De acordo com Marrone (2007), esta ferramenta está licenciada sob a LGPL (GNU *Lesser General Public License*), desenvolvida pela comunidade JOONE, e pode ser utilizada tanto por entusiastas quanto por usuários profissionais.

Marrone (2007) afirma ainda que o JOONE possibilita que as RNAs sejam criadas em uma máquina local, treinadas em ambiente distribuído e executadas em qualquer dispositivo. O mesmo é desenvolvido em componentes que possuem algumas características básicas, tais como, persistência, *multithreading*, serialização e parametrização, o que garante a escalabilidade, confiabilidade e expansibilidade.

2.2.2 NEUROPH

O NEUROPH é um *framework* para desenvolvimento de RNA que consiste em uma biblioteca Java e um editor de RNA denominado NEUROPH *Studio* (Sevarac & Koprivica, 2013). É um projeto *open source* hospedado no repositório *SourceForge* e, desde a versão 2.4, é licenciado sob Apache 2.0. Versões anteriores estão sob a licença LGPL3 (SEVARAC & KOPRIVICA, 2013).

2.2.3 ENCOG

O Encog é um *framework* de Inteligência Artificial Java, .Net e C/C++. Inicialmente, o Encog foi criado para suportar somente RNA, contudo, com o tempo, foi se expandindo para trabalhar com aprendizagem de máquina em geral (HEATON, 2011).

De acordo com Heaton (2011), o Encog faz uso de diversas bibliotecas de terceiros para adicionar funcionalidades necessárias, sem “reinventar a roda”.

O ENCOG possui uma aplicação gráfica conhecida como Encog *Workbench*, que permite realizar diversas tarefas de aprendizagem de máquina, sem que seja necessário escrever código Java ou C#. Ele é escrito em Java, mas gera arquivos que podem ser usados com qualquer *framework* Encog.

2.2.4 Comparação dos Frameworks de RNA Java

Baptista & Dias (2012) afirmam que as informações contidas em artigos e sites que tratam das ferramentas de RNA são suficientes para que se possa tomar uma decisão a respeito daquela que melhor se encaixa ao problema a ser resolvido.

Matviykiv & Faitas (2013) realizaram uma análise com o objetivo de escolher a melhor ferramenta para o desenvolvimento de uma RNA para classificação espectral. Os autores testaram o ENCOG 3.1, o JOONE 2 RC1, o NEUROPH 2.6, e também a ferramenta FANN 2.2, que não é objeto deste estudo. Assim, com base na análise apresentada, os autores puderam concluir que o ENCOG apresentou melhores resultados e maior facilidade de utilização.

Baptista & Dias (2013) apresentaram dados sobre um grande número de ferramentas que podem ser utilizadas para criação de RNA. O trabalho desenvolvido pelos autores levou em consideração os seguintes aspectos: sistema operacional, requisitos mínimos de hardware e software, licença, arquitetura da rede e algoritmo de treinamento.

O Quadro 1 apresenta a relação dos sistemas operacionais suportados por cada *framework* em estudo.

Nota-se que todas as ferramentas são compatíveis mais de um sistema operacional, o que, na prática, garante uma maior interoperabilidade da ferramenta. Em relação aos requisitos mínimos de software e *hardware* necessários para o funcionamento das ferramentas, Baptista & Dias (2013) apresentam poucas informações a respeito dos *frameworks* em estudo.

Quadro 1: Sistemas Operacionais

Frameworks	**Sistemas Operacionais**					
	Windows	**Mac OSX**	**Unix**	**Linux**	**Sun**	**Outros**
ENCOG	sim	sim	sim	sim	não	não
JOONE	sim	sim	não	sim	sim	não
NEUROPH	sim	não	não	sim	não	não

Fonte: Adaptado de Baptista & Dias (2013).

No que diz respeito ao JRE (*Java Runtime Environment)*, o ENCOG precisa ter, no mínimo, versão 1.5 instalada. O NEUROPH funciona somente com versão a partir de 1.6. Quanto ao JOONE, não é apresentada nenhuma informação sobre o JRE, porém, é mencionada a necessidade de, no mínimo 256 MB de memória RAM (*Random Access Memory).* Este fato deve estar associado à escassez de informações sobre o assunto nas diversas fontes consultadas pelos autores.

O Quadro 2 traz a relação das arquiteturas de redes que podem ser desenvolvidas por cada *framework*.

Quadro 2: Arquitetura de Rede

Frameworks	**Arquitetura de Rede**
ENCOG	*Adaline Linear Neuron; Adaptive Resonance Theory; Bidirectional Associative Memory; Boltzmann Machine; Counter-Propagation Neural Network; Elman Network; Hopfield Network; Jordan Recurrent; Kohonen Networks; Multilayer Perceptron; Neuro evolution of Augmenting Topologies; Radial Basis Function; Self-Organizing Map.*
JOONE	*Feed-Forward Neural Network; Kohonen Networks; Modular Neural Network; Principal Component Analysis; Time-Delay Neural Network*
NEUROPH	*Adaline Linear Neuron; Bidirectional Associative Memory; Competitive Neural Network; Hebbian Network; Hopfield Network; Kohonen Networks; Maxnet; Multilayer Perceptron; Radial Basis Function.*

Fonte: Adaptado de Baptista & Dias (2013).

Baptista & Dias (2013) afirmam que a arquitetura de uma RNA está relacionada com a estrutura e o tipo da rede, e que é uma das decisões mais importantes no momento da escolha da ferramenta. Algumas redes são mais apropriadas para alguns tipos de tarefas do que outros, e, desta forma, cada ferramenta abrange apenas um subconjunto de todas as arquiteturas disponíveis.

Outro aspecto importante e destacado pelos autores são os algoritmos de treinamento. O Quadro 3 apresenta os resultados encontrados pelos autores.

Quadro 3: Algoritmo de Treinamento

Frameworks	**Algoritmos de Treinamento**
ENCOG	*Backpropagation; Conjugate Gradient; Competitive Learning; Genetic Algorithm; Levenberg–Marquardt.*
JOONE	*Backpropagation; Batch Training; Conjugate Gradient Descent; Delta-bar-Delta; Resilient Propagation.*
NEUROPH	*Backpropagation; Competitive Learning.*

Fonte: Adaptado de Baptista & Dias (2013).

O ENCOG e o JOONE apresentaram a mesma quantidade de algoritmos de treinamento, embora com diferenças entre ambos. Somente o algoritmo *backpropagation* pode ser utilizado por todos os *frameworks*. Embora não esteja listado no trabalho de Baptista & Dias (2013), o algoritmo de treinamento *Resilient Propagation* também está disponível no ENCOG 3, de acordo com informações apresentadas por Heaton (2010).

O Codeproject (2010a) realizou uma comparação entre o NEUROPH, o ENCOG e o JOONE. Para tal, foi criada uma RNA *feedforward* para reconhecer uma operação XOR.

As versões testadas e comparadas de cada f*ramework* são as seguintes: ENCOG v2.4, NEUROPH v2.4 e JOONE v2 RC1. O computador utilizado foi um *Dell Studio* XPS 8000, com processador *Intel Core* i7 860@ 2.8ghtz. O processador é *quadcore* com *hyperthreading*.

Segundo o autor, a ideia da realização dos testes se baseou na observação de que o *framework* JOONE*,* com o qual o mesmo trabalhava, apresenta uma alta complexidade no código fonte. O objetivo foi verificar se outro *framework* seria capaz de realizar ciclos de treinamento de forma mais rápida fazendo uso do recurso *multicore* das máquinas modernas. O Quadro 4 apresenta algumas informações importantes sobre a RNA criada para os testes:

Quadro 4: Dados de criação da RNA.

Dados da RNA	
Neurônios de Entrada	10
Neurônios de Saída	10
Neurônios da Camada Oculta	20
Função	TANH
Conjunto de Treinamento	100 elementos
Iteração	50
Método de Treinamento	*backpropagation* com momento

Fonte: Codeproject (2010 a)

A Figura 3 apresenta a comparação do tempo total de processamento das RNA criadas utilizando cada um dos *frameworks*. O ENCOG e o JOONE oferecem recursos *multithread*, que possibilitam o processamento paralelo beneficiado, por exemplo, pela tecnologia *multicore* presente nos computadores atuais. O NEUROPH

não apresenta essa funcionalidade. O recurso *multithread* está representado pelas letras MT (*multithread*) e ST (*monothread)* quando não tiver o recurso.

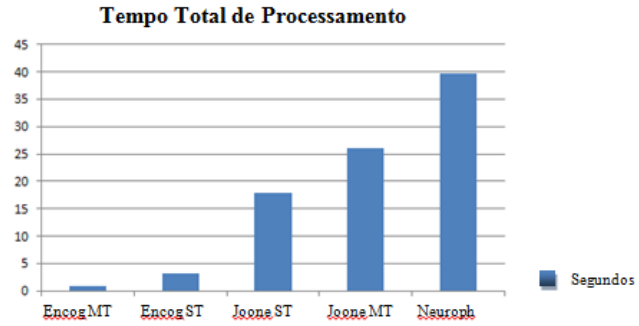

Figura 3: Tempo total de Processamento da RNA.
Fonte: Adaptado de Codeproject (2010 a)

De acordo com o resultado dos testes realizados por Codeproject (2010 a) e apresentados na Figura 3, é possível notar que o ENCOG teve os menores tempos de processamento, tanto em modo MT quanto em modo ST.

O JOONE mostrou um desempenho inferior trabalhando em modo MT, o que significa que, com uma arquitetura desenvolvida utilizando múltiplas *threads,* o processamento da RNA pode levar mais tempo, mesmo sendo utilizada uma máquina *multicore.*

A RNA criada utilizando o NEUROPH foi a que gastou maior tempo para ser processada.

A partir dos testes realizados, o Codeproject (2010a-b) apresentou algumas conclusões, cujas mais relevantes estão relatadas aqui. No geral, o ENCOG e o NEUROPH apresentaram melhores resultados que o JOONE, porém, o *framework* que apresentou superioridade na maioria dos testes foi o ENCOG.

Foram necessárias apenas 18 iterações para o treinamento da rede utilizando o ENCOG, enquanto o NEUROPH realizou 613 iterações e o JOONE mais de 5000 iterações. Talvez essa diferença na quantidade de iterações do ENCOG, em relação aos outros dois *frameworks*, esteja relacionada com o fato do ENCOG utilizar GPU (*Graphics Processing Unit*, ou Unidade de Processamento Gráfico) para aumentar sua velocidade de treinamento.

2.3 Conclusões do Referencial Teórico

As conclusões obtidas com os trabalhos de Baptista & Dias (2013), Matviykiv & Faita (2013) e Codeproject (2010 a-b) possibilitaram observar que o ENCOG apresentou melhores resultados em relação à execução da RNA e, também, na maioria dos quesitos da comparação.

Durante as buscas e pesquisas realizadas, foi possível observar que o ENCOG possui uma grande gama de informações disponibilizadas sob forma de tutoriais, fóruns, listas de discussões e livros publicados, o que, em termos software livre, garante uma maior perspectiva de continuidade, crescimento e desenvolvimento de novos recursos e melhorias da ferramenta.

Por meio deste estudo de pesquisa envolvendo os três *frameworks,* foi possível observar que o ENCOG se apresentou como o *framework* mais completo em termos de funcionalidades, e também o que apresentou melhor desempenho.

3. Metodologia

Ao buscar o *framework* mais adequado ao projeto, os seguintes fatores foram considerados como tendo grande importância:

f) Interoperabilidade – capacidade de funcionar em vários sistemas operacionais;
g) *Framework* Java – compatibilidade com o código do Ururau existente;
h) Possuir código fonte aberto – possibilidade de expansão e melhorias; uso acadêmico;
i) Ser gratuito – uso acadêmico;
j) Possuir facilidade de acesso a informações, atualizações, comunidade engajada – indicativo de melhorias, atualizações, expansão; crescimento; uso acadêmico.

A proposta deste trabalho está centrada em avaliar a possibilidade de acoplamento do módulo contendo um algoritmo de uma RNA a toda estrutura interna do código de um modelo de simulação em Ururau. A Figura 4 ilustra esta proposta.

Note que a ideia é que o próprio modelador possa manipular o módulo inteligente através da camada mais superior do software, ou seja, através de módulos da GUI.

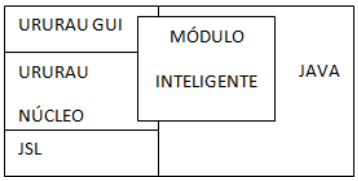

Figura 4: Módulo Inteligente acoplado ao Ururau.

A incorporação do módulo inteligente ao Ururau possibilita, entre outras coisas, que a RNA tenha seus parâmetros configurados no momento da criação do modelo, e pela GUI.

Observe, ainda, que o usuário também tem a opção de poder manipular a RNA nas camadas do núcleo, por meio de linhas de código em Java. Com base nos critérios preestabelecidos, e nas pesquisas realizadas, o ENCOG foi o *framework* escolhido para o desenvolvimento do módulo inteligente no Ururau.

A etapa seguinte à escolha do *framework* foi a realização de testes para saber se a ferramenta seria realmente adequada. Para tal, foram realizados testes na ferramenta, e, inicialmente, duas principais perguntas foram feitas:

- O ENCOG funciona sem falhas ou inconsistências junto ao Ururau?
- O módulo implementado com o ENCOG está gerando os resultados corretos?

Para responder a primeira pergunta, foi realizado um teste, chamado teste de acoplamento. O objetivo deste teste foi identificar possíveis falhas e conflitos entre o código existente do Ururau e o código do *framework* ENCOG. Nesta etapa, também foi possível fazer pequenos ajustes nos códigos, e verificar se o Ururau continuaria "rodando" sem a ocorrência de problemas de incompatibilidades. A princípio, o objetivo foi que os resultados obtidos no teste fossem analisados a partir de um modelo bem simples.

A etapa seguinte foi chamada de teste de funcionamento, que tem relação com a segunda pergunta. Neste caso, não bastaria que os códigos do Ururau e do ENCOG, já unificados, não gerassem conflitos, mas que os resultados obtidos durante a execução de modelos de simulação utilizando RNA gerassem os

resultados corretos. Para isso, foi importante partir de algum modelo já validado e realizar comparações em relação aos resultados obtidos.

Os próximos subitens detalham os passos dos testes descritos.

3.1. Teste de Acoplamento

O primeiro teste realizado com o ENCOG teve como objetivo verificar se o mesmo possui um bom acoplamento com o Ururau em termos de compatibilidade entre os códigos.

Para tal, o ENCOG foi incorporado ao núcleo do Ururau. Pequenas alterações e ajustes foram realizados de forma a interligar o núcleo existente do Ururau e o código do ENCOG. Passada esta primeira etapa de ajuste dos códigos, foi realizado um teste a partir de um modelo hipotético que representa uma decisão “ou exclusivo”, (XOR). Neste primeiro teste, foi criada uma RNA XOR conforme modelo conceitual apresentado na Figura 5.

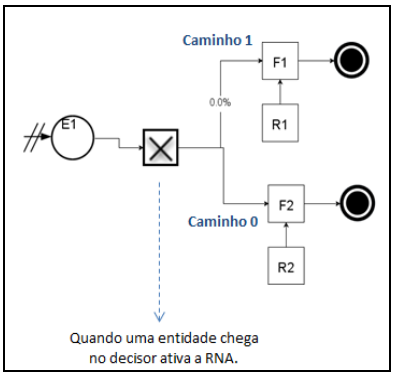

Figura 5: Modelo Conceitual de uma rede XOR no Ururau.

O referido modelo está representado em linguagem IDEF-SIM (Montevechi *et al*., 2010). É importante citar que o Ururau utiliza componentes gráficos semelhantes aos elementos desta linguagem. Essa característica torna a conversão do modelo conceitual em modelo de simulação de entendimento mais fácil e direto.

No problema modelado, quando uma entidade encontra o operador de decisão, a RNA é ativada e a mesma consulta os valores dos neurônios de entrada. A tabela verdade da função XOR funciona como demonstrado na tabela 1:

Tabela 1: Tabela Verdade da Função XOR.

Neurônios de Entrada		Neurônio de Saída
N1	**N2**	**S**
0	0	0
0	1	1
1	0	1
1	1	0

A arquitetura da RNA criada neste teste é apresentada na Figura 6, em que os neurônios de entrada obtêm dados resultantes de expressões processadas pelo Ururau, como por exemplo: tamanho da fila, tempo na fila, testes condicionais, entre outras.

No exemplo representado pela Figura 5, F1 representa a quantidade de entidades na fila do processo F1, e F2 representa a quantidade de entidades na fila do processo F2. O caminho a ser percorrido no modelo pela entidade é definido segundo a decisão da própria RNA e não mais por critérios pré-definidos. Após a execução da RNA, cada entidade segue seu percurso no modelo de acordo com a decisão tomada pela mesma.

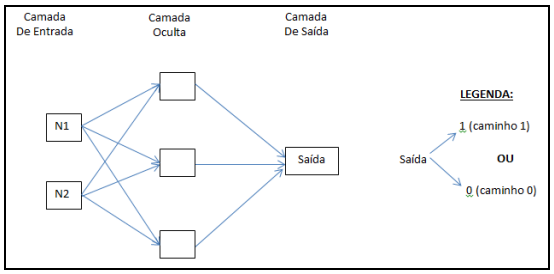

Figura 6: Exemplo da RNA XOR implementada no teste.

Os neurônios da camada de entrada foram configurados de forma que receberão o valor binário 1, no caso de existirem entidades aguardando na fila dos processos F1 e F2, e receberão valor binário 0, no caso de não haver entidades aguardando na fila.

Por se tratar de um problema binário, convencionou-se que, no caso do valor de saída da RNA ser 0, a entidade é encaminhada para o caminho 0 do modelo, e, no caso da saída da RNA ser o valor 1, a entidade será encaminhada para o caminho 1 do modelo (Figura 6).

Neste caso, supondo-se, por exemplo, que, em um dado momento da simulação, F1 seja igual a 1, ou seja, há uma entidade aguardando na fila F1, e F2 seja igual a 0, ou seja, não há entidades aguardando na fila F2, temos que:

N1= “F1>0”, resulta 1 na entrada do primeiro neurônio (N1);

N2= “F2>0”, resulta 0 na entrada do segundo neurônio (N2).

Desta forma, temos as seguintes entradas nos neurônios: N1 = 1 e N2 = 0. Como a RNA implementa um XOR, temos a SAÍDA = 1. Na simulação, a entidade deve seguir pelo CAMINHO 1.

3.2 Teste de Funcionamento

Objetivando verificar o funcionamento do novo módulo inteligente (acoplado ao Ururau), decidiu-se criar um modelo de simulação já validado por Silva *et al.* (2012), de forma que fosse possível afirmar se o módulo inteligente acoplado ao Ururau estaria gerando os resultados corretos.

Para isso, o mesmo modelo de simulação foi gerado tanto no módulo acoplado quanto no módulo inteligente desenvolvido por Silva *et al.* (2012), de modo a permitir a comparação de resultados. É importante destacar que se optou por utilizar uma massa de dados diferente da utilizada por Silva *et al.* (2012).

A Figura 7 apresenta as etapas da simulação como módulo inteligente se comunicando por meio de soquete TCP e as respectivas ferramentas necessárias para a realização das mesmas.

Conforme pode ser observado na Figura 7, a primeira etapa a ser realizada é a criação do modelo de simulação, utilizando-se o Ururau. O modelo de simulação que foi utilizado para simulação e comparação dos resultados obtidos em ambos os módulos inteligentes foi o modelo desenvolvido por Silva *et. al* (2012) em seu trabalho, conforme pode ser visualizado conceitualmente na figura 8.

Após a criação do modelo de simulação, foi necessária a criação da RNA, que, neste caso, foi realizada utilizando o JavaNNS (*Java Neural Network Simulator*). Para a construção desta RNA, Silva *et al.* (2012) utilizaram três neurônios na primeira camada, sendo um neurônio para cada parâmetro.

Definiu-se que o “operador” tomaria suas decisões com base nos seguintes parâmetros: tamanho da fila P2; tempo de vida da entidade E1; e turno (Figura 9).

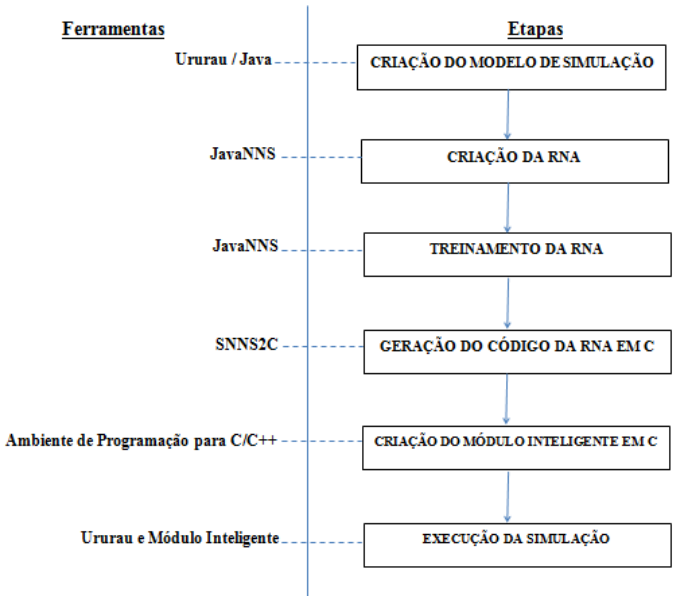

Figura 7: Ferramentas e Etapas da Simulação Inteligente.

Além disso, foram criados três neurônios na camada oculta e um neurônio na camada de saída. Após a RNA criada, foi necessário que a mesma fosse treinada. O treinamento também foi realizado utilizando-se a ferramenta JavaNNS.

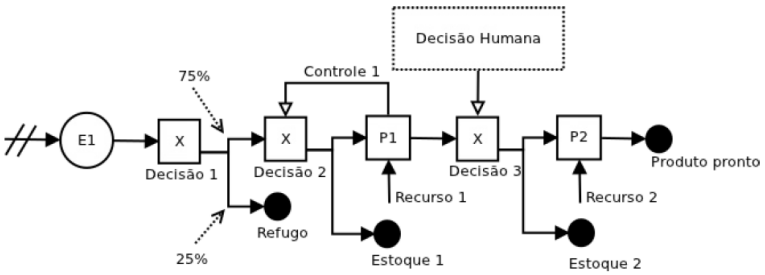

Figura 8: Modelo Conceitual em IDEF-SIM com “decisão 3” usando RNA.
Fonte: Silva, *et al.* (2012).

No trabalho proposto por Silva *et al.* (2012), foi necessário que, após o treinamento da RNA, fosse gerado um código da mesma em linguagem de

programação C. Depois da geração do código em C da RNA, este código foi incorporado ao módulo inteligente, que, por sua vez, pôde interagir com o código do modelo em Ururau, através de uma comunicação com soquetes TCP.

Quando o modelo foi executado, os dados utilizados para a tomada de decisão foram enviados para o módulo inteligente. A RNA foi responsável por definir a decisão, que, por sua vez, é enviada de volta para o modelo em Ururau, fechando, desta forma, o ciclo de operação.

Por outro lado, a partir da utilização do *framework* ENCOG e do acoplamento realizado, foi possível criar o código Java da decisão compatível com o modelo em Ururau. Esse código foi executado todo em memória interna do computador. Assim, as etapas da simulação puderam ser reduzidas em apenas duas. A Figura 9 apresenta a nova estrutura desenvolvida, em que todas as etapas ocorrem diretamente no software Ururau.

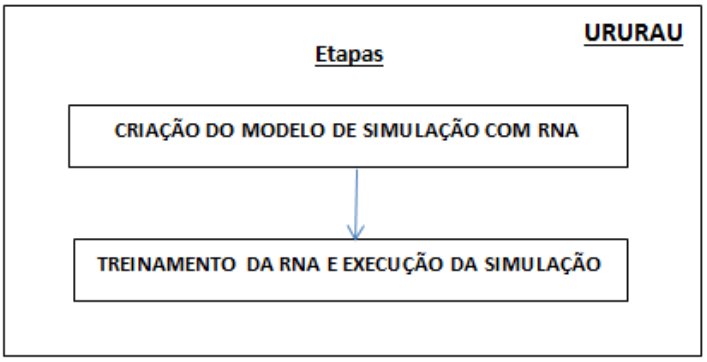

Figura 9: Etapas da Simulação Inteligente

A mesma massa de dados foi submetida a testes no novo módulo inteligente. O modelo utilizado também foi o mesmo de Silva, *et al.,* apresentado na Figura 8. Os resultados obtidos nos testes realizados são discutidos na seção seguinte.

4. Resultados e Discussão

Com a realização do primeiro teste, foi possível observar que o ENCOG funcionou sem problemas de incompatibilidades com o código do Ururau. A RNA criada também apresentou os resultados esperados para os cenários possíveis testados.

Partindo do princípio que os ajustes realizados nos códigos do Ururau e do ENCOG foram realizados com sucesso no processo de acoplamento, e que os resultados obtidos no primeiro teste estavam de acordo com o esperado, deu-se prosseguimento para a etapa seguinte. Desta forma, ao executar o modelo em ambos os módulos inteligentes, os resultados obtidos no teste podem ser visualizados conforme Quadro 5, que segue.

Comparando os resultados da simulação da rede neural JavaNNS (Silva, *et al.* 2012) com o módulo inteligente desenvolvido com o ENCOG, observa-se que, das 2116 decisões, ambas RNA tomaram a mesma decisão em 2106 das vezes.

Quadro 5: Resultados dos Experimentos 1 e 2.

Parâmetro	Experimento 1	Experimento 2
Ferramenta de RNA	**JavaNNS**	**ENCOG**
Tempo médio das entidades na fila P2	233,67 min	235,22 min
Quantidade média de entidades na fila P2	12,01 un	12,20 un
Recurso R2 ocupado	96,48%	96,86%
Decisões com RNA	2116 unidades	
Decisões divergentes entre o experimento 1 e 2	10 unidades	
Percentual de decisões divergentes	0,47%	

Em relação ao modelo de simulação, observou-se uma pequena variação entre os valores de “Tempo médio das entidades na fila P2”; “Quantidade média de entidades na fila P2”; e “Recurso R2 ocupado” obtidos nos dois experimentos. Contudo, é difícil afirmar se tais variações são consequência das decisões divergentes entre as RNA, que ocorrem em apenas 0,47% das vezes, ou da aleatoriedade dos sistemas estocásticos.

5. Conclusões

Os resultados deste trabalho mostraram que o *framework* ENCOG permitiu a realização do acoplamento de uma rede neural artificial com um modelo de

simulação a eventos discretos executado no software Ururau. Ou seja, o respectivo *framework* se mostrou adequado para a criação de um módulo inteligente, sem apresentar erros de funcionamento e incompatibilidades com o código do modelo de simulação. Em relação ao teste comparativo realizado com o módulo proposto e o apresentado pela literatura, foi possível observar que os valores dos resultados encontrados foram compatíveis, mostrando que o ENCOG foi capaz de executar a rede neural de maneira confiável acoplado ao software de simulação utilizado.

O principal benefício obtido com os resultados deste trabalho foi possibilitar que um desenvolvedor construa um modelo de simulação a eventos discretos que tenha algoritmos com decisões inteligentes inseridos no próprio modelo. Ou seja, permitir que um modelador construa um modelo de simulação com uma decisão inteligente baseada em rede neural utilizando somente o software Ururau. Como mostrado em trabalhos anteriores, para se alcançar resultado semelhante era necessário que o desenvolvedor utilizasse ferramentas como: Ururau, JavaNNS, SNNS2C, e ambiente de programação C/C++.

Em relação a trabalhos futuros, novos ajustes e adaptações estão sendo realizados no Ururau, de forma a permitir que modeladores menos familiarizados com a linguagem Java, possam elaborar modelos de simulação com decisões com redes neurais, apenas pela interface gráfica do software. Além disso, o ENCOG permite que outras funcionalidades baseadas em inteligência computacional sejam adicionadas. Assim, por se tratar de um projeto com software de código aberto e livre de custos, os autores convidam outros pesquisadores a cooperarem nesta questão.

Agradecimentos

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) pelo suporte financeiro para esta pesquisa.

Referências

Baptista, D. & Dias, F.M. (2013). A survey of artificial neural network training tools. Neural Comput & Appic. New Applications of Artificial Neural Network in Modeling & Control. Springer-Verlag, London.

Baptista, D. & Dias, F.M. (2012). Artificial Neural Networks: A Review of Training Tools . 10Th Portuguese Conference on Automatic Control, Portugal.

Bergmann, S. & Stelzer, S. & Strassburger, S. (2014). On the use of artificial neural networks in simulation-based manufacturing control. Journal of Simulation, 8, 76-90.

Codeproject. Benchmarking and Comparing ENCOG, NEUROPH and JOONE Neural Networks. (2010 - a). Disponível em: http://www.codeproject.com/Articles/85487/Benchmarking-and-Comparing-Encog-NEUROPH-and-JOONE. Acesso em: 28/06/2013.

Codeproject. Comparing Neural Networks in NEUROPH, ENCOG and JOONE. (2010 - b). Disponível em: http://www.codeproject.com/Articles/85385/Comparing-Neural-Networks-in-Neuroph-Encog-and-JOO. Acesso em: 28/06/2013.

Heaton, J. (2010). Introduction to ENCOG 2.5 . Heaton Research, Inc. 108p. Disponível em: Http://Www.Heatonresearch.Com/Encog. Acesso: 15 de Julho de 2013.

Heaton, J. (2011). Programming Neural Networks with Encog3 in Java. Heaton Research, Inc. St Louis, MO, USA. 242p.

JOONE. (2013). An Object Oriented Neural Engine. Disponível em: http://sourceforge.net/projects/joone/. Acesso em 03 de Junho de 2013.

Marrone, P. (2007). JOONE Java Object Neural Engine. The Complete Guide: All you need to know about JOONE. 142 p. Disponível em: http://www.joone.org.

Matviykiv,O.M & Faitas, O.I., (2013). Data Classification of Spectrum Analysis Using Neural Network. Lviv Polytechnic National University, Computer-Aided Design Department.

Montevechi, J.A.B. & Leal, F. & De Pinho, A.F. & Da Silva, R.F.C. & De Oliveira, M.L.M. & Da Silva, A.L.F. (2010). Conceptual modeling in simulation projects by mean adapted IDEF: An application in a Brazilian tech company. In: Winter Simulation Conference, p. 1624-1635.

Peixoto, T.A. & Rangel, J.J.A. & Matias, I. O. & Montevechi, J.A.B. & Miranda, R.C. (2013). Ururau – Um Ambiente para Desenvolvimento de Modelos de Simulação a Eventos Discretos. PODes - Revista Eletrônica Pesquisa Operacional para o Desenvolvimento. Vol.5, n.3, p.373-405.

Sevarac, Z & Koprivica, M. (2013). Getting Starded With NEUROPH. Retirado de: http://neuroph.sourceforge.net. Acesso em 13/07/2013.

Silva, D.V.C. & Rangel, J.J.A. & Matias, I.O. & Vianna, D.S. & Peixoto, T.A. (2012). Modelos de Simulação a Eventos Discretos com Aspectos de Decisão Humana: Uma Aplicação com o Ururau. PODes - Revista Eletrônica Pesquisa Operacional para o Desenvolvimento. Vol.4, n.3, p.339-355.

Robinson, S. & Alifantis, T. & Hurrion, R. & Edwards, J. S. & Ladbrook, J. & Waller, T. (2001). Modelling and Improving Human Decision Making With Simulation. In: Winter Simulation Conference, IEEE, Arlington, P. 913-920.

Rosseti, M. D. (2008). Java Simulation Library (JSL): An Open-Source Object-Oriented Library for Discrete-Event Simulation in Java. International Journal of Simulation and Process Modelling, Vol. 4, N. 1, p.69-87.

Swain, J. J. (2007). Discrete Event Simulation Software: New Frontiers in Simulation, OR/MS Today - INFORMS, Vol. 34, No. 5, pp.32-43.

Zuben, F. J. V. Uma caricatura funcional de redes neurais artificiais. Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais, Vol. 1, No. 2, p.- 66-76, 2003.