RICARDO LUIZ BARROS LEITE CAMPOS

REGULAGEM EM TEMPO-REAL DE UM SISTEMA INTEGRADO DE SEMÁFOROS: UMA APLICAÇÃO DE REDES NEURAIS RECORRENTES E CONTROLE ADAPTATIVO

Dissertação apresentada ao Programa de Pós-Graduação "*Stricto-Sensu*" em Gestão do Conhecimento e Tecnologia da Informação da Universidade Católica de Brasília, como requisito para obtenção do Titulo de Mestre em Gestão do Conhecimento e Tecnologia da Informação.

Orientadora: Prof[a]. Dra. Lourdes Mattos Brasil

Co-Orientador: Prof. Dr. Roberto Célio Limão de Oliveira

BRASÍLIA-DF

2006

C198r Campos, Ricardo Luiz Barros Leite.
Regulagem em tempo-real de um sistema integrado de semáforos: uma aplicação de redes neurais recorrentes e controle adaptativo / Ricardo Luiz Barros Leite Campos - 2006.
187 f. ; 30 cm

Dissertação (mestrado) – Universidade Católica de Brasília, 2006.
Orientação: Lourdes Mattos Brasil
Co-orientação: Roberto Célio Limão de Oliveira

1. Sistema de controle ajustável. 2. Redes neurais (computação). Trânsito – controle eletrônico. I. Campos, Ricardo Luiz Barros Leite. II. Título

CDU 004.7

Ficha elaborada pela Coordenação de Processamento do Acervo do SIBI – UCB.

SUMÁRIO

LISTAS DE FIGURAS 4
LISTAS DE TABELAS 6
RESUMO 7
ABSTRACT 8
1 Introdução 9
1.1 Justificativa 9
1.2 Objetivo 13
1.2.1 Objetivo Geral 13
1.2.2 Objetivos Específicos 13
1.3 Revisão da Literatura 13
1.4 Estrutura da Dissertação 18
2 Referencial Teórico 20
2.1 Regulagem de Semáforos 20
2.1.1 Trânsito Terrestre 20
2.1.2 Sinalização Semafórica de Regulamentação 25
2.2 Rede Neural Artificial 31
2.2.1 *Perceptron* 31
2.2.2 Número de Neurônios 33
2.2.3 Modo de Conexão 34
2.2.4 Neurodinâmica 36
2.3 Controle 43
2.3.1 Controle Adaptativo 49
2.3.2 Controlador *Fuzzy* 50
2.4 Gestão do Conhecimento 55
2.4.1 Gestão do Conhecimento de Um Sistema Integrado de Controle de Semáforos em Tempo-Real 59
3 Metodologia 64
3.1 Formulação do Problema 64
3.2 Alternativa para a Resolução do Problema 66
3.2.1 Controlador de Fluxo de Veículos 68
3.2.2 Controlador Semafórico da Intersecção 75

3.2.3 Rede Neural Local 78
4 Resultados e Discussões 80
4.1 A Arquitetura para a Solução do Problema 80
4.2 Controlador de Fluxo de Veículos 86
4.3 Controlador Semafórico da Intersecção 98
4.4 Rede Neural Local 104
5 Conclusão 130
Referências Bibliográficas 133
Anexos 143
Anexo 01 – 1º Simulador da Quantidade de Veículos da Via 143
Anexo 02 – Tabela com o Resultado da Execução do 1º Simulador da Quantidade de Veículos da Via 144
Anexo 03 – 2º Simulador da Quantidade de Veículos da Via 146
Anexo 04 – Tabela com o Resultado da Execução do 2º Simulador da Quantidade de Veículos da Via 148
Anexo 05 – Tabela com o Resultado da Execução do 2º Simulador da Quantidade de Veículos da Via 151
Anexo 06 – 3º Simulador da Quantidade de Veículos da Via e Fluxo da Via 154
Anexo 07 – Tabela com o Resultado da Execução do 3º Simulador da Quantidade de Veículos da Via e Fluxo da Via 156
Anexo 08 – Controlador de Fluxo de Veículos 162
Anexo 09 – Controlador de Semáforos da Intersecção 165
Anexo 10 – Parâmetros da RNL 171
Anexo 11 – Rede Neural em C 173

LISTAS DE FIGURAS

Figura 1-1 – Cruzamento entre Vias 11
Figura 2-1- Intersecção entre duas (2) vias..... 22
Figura 2-2 – Intersecção de três (3) ou mais vias 23
Figura 2-3 – Tipos de Conflitos 24
Figura 2-4 – Sinalizações Semafóricas de Regulamentação com Três (3) Cores (BRASIL, 1997, p.163) 27
Figura 2-5 – Cores Utilizadas nas Sinalizações Semafóricas de Regulamentação (BRASIL, 1997, p.164) 27
Figura 2-6 – Semáforo com Direção Controlada (BRASIL, 1997, p.164) 28
Figura 2-7 – Estratégias de Controle de Semáforos (LO, CHAN e CHOW, 2001) 30
Figura 2-8 – Neurônio Artificial (LUGER, 2002, p.419) 32
Figura 2-9 – Rede Neural Recorrente *Backpropagation Through Time* – BPTT (HAYKIN, 2001, p.49)..... 36
Figura 2-10 – Aprendizado Supervisionado (BRAGA, CARVALHO e LUDEMIR, 2000, p. 17) 40
Figura 2-11 – Aprendizado Não Supervisionado (BRAGA, CARVALHO e LUDEMIR, 2000, p. 19) 42
Figura 2-12 – Controle de Malha Aberta (Baseado em: DORF e BISHOP, 2001, p. 3; HAYKIN, 2001, p.551; HAYKIN e VAN VEEN, 2001, p. 26)..... 45
Figura 2-13 – Controle de Malha Fechada (Baseado em: DORF e BISHOP, 2001, p. 3; HAYKIN, 2001, p.551; HAYKIN e VAN VEEN, 2001, p. 26).... 46
Figura 2-14 – Controlador *Fuzzy* (DIAS e BARROS, 2005, p. 148; HAYKIN, 2001, p.551) 51
Figura 2-15 – Conjuntos *Fuzzy* 52
Figura 2-16 – Lógica *Fuzzy* (Baseado em DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.66)..... 53
Figura 2-17 – Quatro Modos de Conversão do Conhecimento (NONAKA e TAKEUCHI, 1997, p.69) 56

Figura 2-18 – Conteúdo do Conhecimento Criado (NONAKA e TAKEUCHI, 1997, p.81) 57
Figura 2-19 Gestão do Conhecimento x Controle de Semáforos (MULLER, MISKA e VAN ZUYLEN, 2005, p. 11) 62
Figura 3-1 – Cruzamento entre Três (3) Vias 65
Figura 3-2 – Sistema de Regulagem de Semáforos Distribuído 66
Figura 3-3 – Modelo Geral para Resolução do Problema 68
Figura 3-4 – Detectores Veiculares 69
Figura 3-5 – Rede Neural Local 78
Figura 4-1 – Cruzamento entre Três (3) Vias 80
Figura 4-2 – Arquitetura do Sistema Distribuído 81
Figura 4-3 – Nova Arquitetura do Sistema Distribuído 82
Figura 4-4 – Modelo Geral 84
Figura 4-5 – Intersecção entre duas (2) vias 85
Figura 4-6 – Detectores Veiculares 87
Figura 4-7 – Situação I 89
Figura 4-8 – Situação IV 90
Figura 4-9 – Situação II 90
Figura 4-10 – Situação III 91
Figura 4-11 – Conjuntos Difusos de Veículos 95
Figura 4-12 – Conjuntos Difusos de Velocidade 96
Figura 4-13 – Estratégias de Controle de Semáforos 99
Figura 4-14 – Rede Neural Local Inicial 110
Figura 4-15 – RNA Criada pelo EasyNN 112
Figura 4-16 – Rede Neural Local Inicial sem Recorrência 115
Figura 4-17 – Rede Neural Local – RNL 129

LISTAS DE TABELAS

Tabela 1-1 – Evolução da frota de veículos por Região em 1990 e de 2001 a 2003 9
Tabela 1-2 – Frota de Veículos do Distrito Federal de 2000 a 2005 10
Tabela 3-1 – Volume do Tráfego na Pista 70
Tabela 3-2 – Volume do Tráfego da Via 71
Tabela 3-3 – Velocidade Média 73
Tabela 3-4 – Velocidade do Tráfego da Via 73
Tabela 3-5 – Indicações Luminosas dos Semáforos 76
Tabela 3-6 – Fases dos Semáforos da Intersecção 76
Tabela 4-1 – Situações Possíveis de Acordo com os Detectores Veiculares 88
Tabela 4-2 – Comparação entre q_v e q_a na Simulação 92
Tabela 4-3 – Quantidade de Veículos 95
Tabela 4-4 – Velocidade Média 96
Tabela 4-5 – Indicações para a RNL 97
Tabela 4-6 – Composição das Indicações Luminosas 100
Tabela 4-7 – Ordem versus Composições das Indicações Luminosas 100
Tabela 4-8 – Fases dos Semáforos da Intersecção 102
Tabela 4-9 – Fluxos das Vias versus Saída da RNL 106
Tabela 4-10 – Fluxos das Vias versus Saída da RNL 108
Tabela 4-11 – Fluxos da Via, Estado Atual e Saída da RNL 109
Tabela 4-12 – Dados para Teste da Rede 123
Tabela 4-13 – Erro no Teste da Rede 123
Tabela 4-14 – Neurônios da Camada Intermediária 124
Tabela 4-15 – Taxa de Erro 125
Tabela 4-16 – Número de Épocas 125
Tabela 4-17 – Taxa de Aprendizagem 126
Tabela 4-18 – Momento 127
Tabela 4-19 – Simulações na RNA 128

RESUMO

O semáforo é um dispositivo de controle de tráfego que, através de indicações luminosas, alterna o direito de passagem de veículos em intersecções de duas ou mais vias, com o objetivo de melhorar o fluxo de veículos, em termos de fluidez e segurança. Para tanto é necessário que esteja bem regulado. A regulagem de semáforos tem o objetivo de estabelecer o melhor tempo de ciclo, que é o tempo necessário para a seqüência completa de indicações de verde, amarelo e vermelho de um semáforo, para tentar otimizar uma via, através da maximização da capacidade do fluxo da rede, da minimização do impacto negativo do tráfego no meio ambiente, do aumento da segurança no tráfego e diminuição do tempo de viagem. Uma alternativa viável é que o controlador de trânsito, que é o equipamento que comanda o semáforo, pudesse ter o tempo de ciclo estabelecido conforme as condições do fluxo de veículos de forma automática, através de uma Rede Neural Artificial Recorrente e Controle Adaptativo, que é o tema proposto neste trabalho.

Palavras-Chave: Controle Adaptativo; Redes Neurais Recorrentes; Semáforos.

ABSTRACT

Traffic light is a device for controlling traffic which, by means of colorful lights, alternates the right of vehicles to go across junctions of two or more roads, with the objective of improving the flow of vehicles, in terms of fluidity and safety. To do that, it is necessary for it to be well regulated. The regulation of traffic lights aims at establishing the best cycle time, which is the necessary time for the complete sequence of the green, yellow and red lights of a traffic light to try to enhance the road, through the maximization of the network flux capacity, the minimizing of the traffic negative impact on the environment and the decrease of the journey. A viable alternative is that the traffic controller, which is the equipment that commands the traffic light, could have the cycle time established according to the vehicle flux conditions in an automatic way, by means of a Recurrent Artificial Neural Network and Adaptable Controller, which is the theme proposed in this work.

Key-words: Adaptable Controller; Recurrent Neural Networks; Traffic Lights.

1 INTRODUÇÃO

Esta dissertação pretende estudar a aplicação de Redes Neurais Artificiais Recorrentes – RNR em conjunto com Controle Adaptativo. O resultado deste estudo será aplicado na regulagem em tempo-real de um sistema integrado de semáforos, que controle duas (2) intersecções entre três (3) vias.

1.1 Justificativa

Nas últimas décadas, o aumento da frota de veículos no Brasil (Tabela 1-1), em treze (13) anos, de 1990 a 2003, teve um crescimento de mais de 100%. Já no Distrito Federal, o acréscimo ocorrido nos últimos cinco (5) anos, de 2000 e 2005, foi de pouco mais de 40% (Tabela 1-2). Uma das conseqüências deste aumento, em especial nas principais capitais e em algumas cidades, é o "aumento do volume do tráfego e junto com ele, os níveis de utilização da rede viária" (DA SILVA e LINDAU, 1997).

Tabela 1-1 – Evolução da frota de veículos por Região em 1990 e de 2001 a 2003

Regiões	Frota de Veículos			
	1990	2001	2002	2003
Brasil	18.357.245	31.913.003	34.284.967	36.658.501
Norte	362.499	934.461	1.054.358	1.184.259
Nordeste	1.872.923	3.701.422	4.079.993	4.448.287
Sudeste	10.979.245	17.890.927	19.013.742	20.083.423
Sul	3.913.051	6.852.260	7.366.353	7.928.580
Centro-Oeste	1.229.527	2.533.933	2.770.521	3.013.952

Fonte: DENATRAN (2006).

O aumento dos níveis de utilização da rede viária faz com que o trânsito, em algumas vias e em determinados horários, torne-se lento e demorado, ou

seja, fique congestionado. A conseqüência imediata deste problema é o aumento do consumo de combustíveis, que é considerado um problema de ordem econômica, mas, também, um fator de degradação ambiental (DA SILVA e LINDAU, 1997). Outro ponto a considerar é que um fluxo de veículos lento e demorado, também é considerado um causador de acidentes de trânsito (MEIRELLES, 1998).

Tabela 1-2 – Frota de Veículos do Distrito Federal de 2000 a 2005

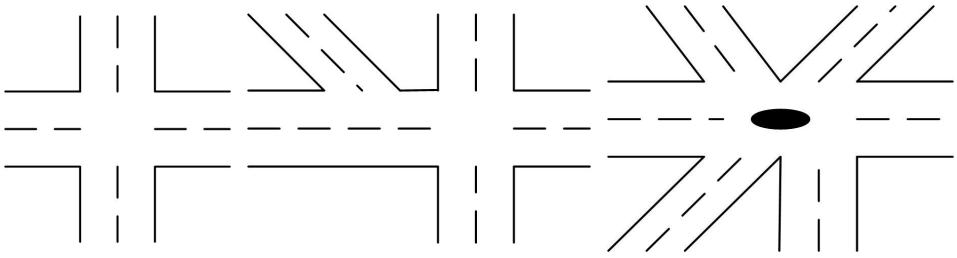
Ano	Frota	Variação Anual	
		Número	%
2000	585.424	--	--
2001	651.342	65.918	11
2002	688.746	37.404	06
2003	732.138	43.392	06
2004	775.112	42.974	06
2005	821.352	46.240	06

Fonte: DETRAN/DF (2006).

Uma das formas de evitar os problemas já citados é melhorando o fluxo de veículos nas vias, e isto deve ser uma das principais preocupações dos departamentos de trânsito locais. A melhoria no tráfego, além de reduzir o consumo de combustíveis, o que gera economia de recursos, e a emissão de poluentes no ar, também minimiza o estresse causado pelo trânsito lento e demorado (OSSOWSKI, CUENA e GARCIA_SERRANO, 1998, p.4).

Na literatura sobre Engenharia de Trânsito há várias alternativas para melhorar o fluxo dos veículos em uma cidade. No entanto, a maioria delas, como a de construção de vias alternativas, além de muito onerosa, não pode ser implementada imediatamente. No entanto, mesmo que a construção de vias alternativas possa ser feita imediatamente, se no seu resultado final houver cruzamento entre duas (Figura 1-1.a) ou mais vias (Figura 1-1.b e Figura 1-1.c), os movimentos dos veículos não poderão ser realizados

simultaneamente, e será necessário estabelecer regras de controle de direito de passagem (DENATRAN, 1984, p.13).



a) Duas (2) vias b) Três (3) vias c) Quatro (4) vias

Figura 1-1 – Cruzamento entre Vias

A melhor regra, quando o volume do tráfego é muito intenso, é a "definição de uma ordenação seqüencial e cíclica de passagem no cruzamento, através da determinação de um tempo para realização da travessia de uma determinada via enquanto a outra via permanece sem movimentar" (DENATRAN, 1984, p.13-14), utilizando um semáforo[1]. No entanto, a instalação de um semáforo em uma intersecção só melhorará o desempenho do tráfego, em termos de fluidez e segurança, se o mesmo estiver sempre bem regulado.

O Denatran (1984) estabelece uma série de critérios para instalação de semáforos, não apenas o volume de tráfego muito intenso. Por outro lado, não há semáforo que resolva quando a intersecção atinge a saturação.

O objetivo da regulagem de um conjunto de semáforos é determinar qual o melhor tempo de ciclo[2], para tentar, otimizar uma via utilizando os seguintes critérios:

---

[1] É um dispositivo de controle de tráfego que, através de indicações luminosas, alterna o direito de passagem de veículos em intersecções de duas ou mais vias (DENATRAN, 1984, p. 14).

[2] O tempo necessário para a seqüência completa de indicações de verde, amarelo e vermelho de um semáforo.

- Maximizar a capacidade de fluxo dos veículos;
- Minimizar o impacto negativo do tráfego no meio ambiente;
- Incrementar a segurança do tráfego; e
- Reduzir o tempo de viagem.

A principal forma de regulagem de semáforos, adotada no Brasil, é a de tempo fixo, onde os planos são calculados baseados em dados históricos do fluxo do tráfego e os semáforos são programados para entrarem em operação numa determinada hora do dia (DENATRAN, 1984, p.49; SERRA, 2004, p.20). O tempo fixo como regulagem de semáforos é considerado eficiente, mas exige que se faça uma previsão de médio prazo do comportamento do tráfego, aproximando os tempos próximos das piores condições encontradas dentro da operação do plano (SERRA, 2004, p.20). No entanto, se o fluxo de veículos assumir características diferentes da pré-estabelecida, para menos ou para mais, a regulagem não será eficiente e o trânsito poderá ficar lento para, pelo menos, uma das vias.

Para resolver o problema da regulagem de tempo fixo, o ideal seria se o controlador de trânsito, que “é o equipamento que comanda o semáforo” (SERRA, 2004, p.26), pudesse ser operado manualmente, ou seja, que o tempo das indicações luminosas não fosse constante ao longo do tempo e fosse dependente da situação do tráfego verificada por um operador. Mas, os critérios pessoais do técnico responsável pelo controlador podem determinar ciclos ruins e, conseqüentemente, prejudicar o fluxo dos veículos.

Como uma Rede Neural Artificial – RNA pode simular o processamento humano de informações, desconsiderando critérios pessoais, a sua utilização na regulagem em tempo-real de um conjunto de semáforos, de acordo com o fluxo de veículos num determinado momento, pode ser uma excelente alternativa. É esta a proposta desta dissertação: a utilização de Rede Neural Recorrente e de Controle Adaptativo para a regulagem de um sistema integrado de semáforos em tempo-real que controle duas (2) intersecções.

1.2 Objetivo

1.2.1 Objetivo Geral

Definir e desenvolver uma RNA recorrente e um controle adaptativo para a regulagem de um sistema integrado de semáforos em tempo-real para melhoria do tempo de cor verde.

1.2.2 Objetivos Específicos

- Identificar a melhor arquitetura de uma RNA recorrente para a resolução do problema.
- Identificar o melhor controle dinâmico para, em conjunto com a RNA, ajudar na resolução do problema.
- Demonstrar que a arquitetura de RNA recorrente proposta em conjunto com o controle dinâmico pode ser utilizada para a resolução do problema.
- Desenvolver um mecanismo para armazenamento e recuperação de todas as regulagens aplicadas ao sistema integrado de semáforos.

1.3 Revisão da Literatura

Na revisão de literatura sobre o controle de tráfego urbano em tempo-real foram identificados vários trabalhos[3], dos quais alguns serão comentados a seguir pela sua relevância ao tema abordado neste trabalho:

---

[3] As pesquisas foram feitas na ACM (www.acm.org), no IEEE (www.ieee.org), na Universidade Federal de Santa Catarina (www.ufsc.br), na Universidade Estadual de Campinas (www.unicamp.br), na Universias (www.universias.org.br); no *google* (www.scholar.google.com); no Transportation Research Board (www.trb.org); e na Associação Nacional de Pesquisa e Ensino de Transportes (www.anpet.org.br).

- O trabalho de Ossowski, Cuena e de García-Serrano (1998) refere-se ao sistema TRYSA2 (TRYS Agente Autônomo) que utiliza agentes inteligentes para controlar semáforos de Barcelona, com o objetivo de resolver o problema de congestionamento do trânsito. Os autores descrevem a infra-estrutura para o gerenciamento do trânsito, o funcionamento dos agentes inteligentes e como deve ser a comunicação entre o computador central e os agentes. No entanto, o sistema estava em fase experimental quando o trabalho foi escrito e não foi possível validar se o mesmo está sendo utilizado na Espanha e/ou no mundo, pois após a publicação desse artigo não houveram outros trabalhos escritos pelos mesmos autores dando continuidade ao mesmo.

- No trabalho desenvolvido por Park, Messer e Urbanik II (1998) é descrita a necessidade de controle otimizado de sinais de trânsito (semáforos) através de barreiras, identificação do tempo de fase mínimo e um tamanho de ciclo ideal. A proposta dos autores é a utilização de um programa utilizando algoritmo genético como otimizador do controle de semáforos em conjunto com um simulador para um conjunto de vários semáforos. A utilização de algoritmo genético num simulador se mostrou mais eficiente do que os simuladores existentes no mercado. No entanto, de acordo com a conclusão do artigo, os autores sugerem a utilização do otimizador em um ambiente de simulação reduzido (ou microscópico).

- A construção de um sistema de controle de tráfego, segundo o trabalho de Nakamiti e Freitas (2002), que seja flexível, autônomo, com inteligência para interagir com o ambiente é uma necessidade. O sistema em questão, além dos conceitos convencionais de sistemas, contém os conceitos e as técnicas de inteligência artificial, e computação em tempo-real, tornando-se um gerenciador de tráfego inteligente. Além disto, a utilização de agentes autônomos que trabalham cooperativamente, permitiu alcançar o objetivo da proposta

dos autores, através de uma simulação, baseada em informações reais da cidade de Campinas/SP.

- O trabalho de Roozemond e Rogier (2000) contém a proposta de construção de uma unidade de controle de trânsito adaptativo baseado na tecnologia de agentes inteligentes para alterar um semáforo em tempo-real. A aplicação de agentes inteligentes em um sistema de controle de semáforos, segundo os autores, poderá garantir uma operação ótima e balanceada de acordo com o fluxo de veículos. Como conclusão, é destacado que o sistema de controle adaptativo pode ajustar de forma otimizada semáforos para controlar o trânsito, fato comprovado através de simulação.

- Oliveira, Fargesy, Moreira e Kraus Jr (2002) propõem, no seu trabalho, um novo conceito para cruzamentos viários com uso intensivo de componentes de Sistemas Inteligentes de Transportes, através da substituição dos semáforos atualmente existentes por um Controlador Veicular Centralizado - CVC, que será responsável pela aceleração e rotas dos veículos nas proximidades de um cruzamento automatizado. Segundo os autores, o CVC permitirá que os veículos circulem com folgas menores entre si do que quando guiados por motoristas, reduzindo o congestionamento que ocorre devido as restrições de capacidade impostas pelos conflitos transversais. O problema apresentado no artigo é de uma intersecção de duas vias, onde o CVC, utilizando um algoritmo inteligente, minimizará os atrasos veiculares em tempo real, garantindo o tempo mínimo de percurso, e que segundo a conclusão escrita, permitiu o aumento de capacidade obtido pelo CVC em relação ao cruzamento semaforizado.

- Uma arquitetura de agentes múltiplos para controle de semáforos em tempo-real em trânsito urbano foi a proposta do trabalho desenvolvido por Choy, Cheu, Srinivasan e Logi (2002). A arquitetura

tem três (3) camadas de agentes controladores: o de intersecção; o de zonas; e o regional. Cada um dos agentes controladores foi implementado utilizando os conceitos de inteligência artificial com lógica *fuzzy*, RNA e algoritmos evolucionários, que permitiram alterar dinamicamente um conjunto de semáforos em um trânsito complexo, atualizando uma base de conhecimento para ser tratada posteriormente. Após simulação baseada em um ambiente real, o resultado, comparado ao controle de trânsito em tempo fixo, foi melhor em todas as condições tratadas.

- Um *software* denominado Sistema de Controle de Semáforos Ótimo que foi desenvolvido para controlar os semáforos dinamicamente é apresentado no trabalho de Saito e Fan (2000). Este sistema, baseado nos conceitos de inteligência artificial, foi desenvolvido em *Visual Basic* e utiliza uma RNA, a LOSANN (*Level of Service Analysis Neural Network*), com algoritmo de aprendizado *backpropagation*. A utilização da RNA permitiu estabelecer um ótimo tempo do sinal de trânsito para o controle de intersecções sinalizadas com duas (2) fases, em um ambiente de simulação para computador pessoal.

- O trabalho de De Ré (1995) apresenta a abordagem de utilização de RNA para ampliar o escopo de controle, possibilitando a atualização de regras da máquina de inferência do sistema de controle. Através desta abordagem, as melhorias são efetivas nas máquinas de inferência de sistemas inteligentes. Como exemplo para a aplicação desta proposta, foi estudado um controlador de tráfego urbano para uma única intersecção isolada, o qual apresenta uma lógica difusa, através de esquema adaptativo, com ajuste de funções de pertinências de acordo com determinadas condições de tráfego. A proposta foi comprovada no trabalho e com possibilidades de trabalhos futuros. A autora recomenda a ampliação do universo a ser analisado, estendendo o controle a um número maior de intersecções que possam influenciar sobre a determinação do fluxo de veículos.

- Chin e Smith (1996) propõem, em seu trabalho, a utilização de controlador neural para identificar as condições do trânsito e alterar os parâmetros de tempo de fases de um semáforo em uma intersecção. Através da utilização do controlador foi possível otimizar, em tempo-real, o tempo do ciclo baseado nas condições do tráfego.

- Um sistema distribuído com controladores *fuzzy* cooperativos para o gerenciamento de grupos de intersecções semaforizadas é a proposta do trabalho de Lee e Lee-Kwang (1999). Os autores relatam no trabalho, a necessidade de existir um mecanismo de cooperação entre os controladores de sinais para garantir a sincronização dos semáforos com o objetivo de melhorar o fluxo de veículos e reduzir o congestionamento do trânsito. As informações necessárias, que são utilizadas como parâmetros nos controladores *fuzzy*, são capturas em tempo-real através de sensores e através delas, o melhor tempo de ciclo é determinado dinamicamente. Como é um trabalho experimental, no laboratório a proposta mostrou-se satisfatória.

- A utilização de multi-agentes descentralizados para controlar uma rede de tráfego urbano é apresentado por Ferreira, Subrachmanian e Manstetten (2001). Cada um dos agentes é responsável em controlar os semáforos de uma única intersecção. Este controle, feito de forma dinâmica, está baseado nas informações coletados sobre o fluxo de veículos pelos sensores existentes nas faixas da via.

- Manikonda *at al* (2001) descrevem a infraestrutura de um sistema composta por agentes autônomos para controlar semáforos em vias urbanas. Este sistema utilizando processamento distribuído seria capaz de resolver todos os problemas de controle de intersecções semaforizadas. Cada agente será responsável em controlar, em tempo-real, uma intersecção.

Os trabalhos anteriormente relacionados referem-se a ajustes de controladores de tráfego urbano em tempo-real de acordo com o fluxo de veículos em duas (2) ou mais vias concorrentes.

Na literatura, há outros trabalhos sobre controle de tráfego urbano que relatam a necessidade da existência de sistemas que ajustem um conjunto de semáforos em tempo-real, mas apresentam somente o fundamento teórico sem implementação e nem validação da teoria proposta. Portanto, não foram relevantes para o trabalho em estudo, como são os casos de:

- Biora at al (1995);
- Camponogara, Souza e Kraus Junior (1995);
- Molina, Robledo e Fernández (2000);
- Huang (1996);
- Sadek, Smith e Demetsky (1998);
- Davis (2004);
- Kirshfink, Hernández e Boero (2000);
- Nauman, Rasche e Tacken (1998);
- Bellemans, De Schutter e De Moor (2002);
- Ho e Ioannou (1996);
- Stewart, Lepik e Van Aerde (1998);
- Mirchandani, Nobe e Wu (2001);
- Wei, Wang e Du (2005);
- Saito (2000); e
- Schönhof e Helbing (2004).

1.4 Estrutura da Dissertação

Esta dissertação está dividida em cinco (5) capítulos:

- Este capítulo, a introdução, é o primeiro;

- O Referencial Teórico que “consiste no detalhamento dos conceitos extraídos da bibliografia e os estudos já realizados por outros autores sobre o tema ou especificamente sobre o problema e que serão utilizados” neste trabalho (MORESI, 2004, p.106), será a segunda parte apresentada. Nele, os assuntos como Regulagem de Semáforos, Redes Neurais Artificiais, Lógica *Fuzzy*, Controles e Gestão do Conhecimento serão tratados.
- Na terceira parte, denominada de Metodologia, será descrita com detalhes “a metodologia utilizada” (MORESI, 2004, p.107), ou seja, da estrutura e funcionamento da RNA e dos Controles utilizados para a resolução do problema.
- O processo de desenvolvimento, a simulação e os resultados da aplicação dos Controles e da RNA na regulagem do sistema de semáforos serão apresentados na quarta parte em conjunto com avaliação dos resultados alcançados na regulagem do sistema de controle de semáforos e a comparação dos mesmos com outros estudos anteriormente realizados denominada de Resultados e Discussão.
- Na última e quinta parte, a conclusão sobre o trabalho desenvolvimento, bem como a proposta de estudos futuros sobre assuntos relacionados direta ou indiretamente com este trabalho, serão comentados.

2 REFERENCIAL TEÓRICO

O referencial teórico, segundo Moresi (2004), “consiste no detalhamento dos conceitos extraídos da bibliografia e os estudos já realizados por outros autores sobre o tema ou especificamente sobre o problema e que serão utilizados no trabalho de dissertação”.

Neste trabalho, o referencial teórico estará dividido em quatro (4) partes:

a) Na primeira parte, a regulagem de semáforos será apresentada, bem como alguns conceitos relacionados direta ou indiretamente com ela;
b) O estudo sobre as RNA será apresentado na segunda parte;
c) No terceiro item, será detalhada a parte referente aos controles;
d) A parte referente à gestão do conhecimento estará detalhada no quarto item.

2.1 Regulagem de Semáforos

2.1.1 Trânsito Terrestre

O trânsito terrestre, ou simplesmente, trânsito, segundo o artigo 1º do parágrafo 1° do Código de Trânsito Brasileiro (BRASIL, 1997), é a “utilização das vias por pessoas, veículos e animais, isolados ou em grupos, conduzidos ou não, para fins de circulação, parada, estacionamento e operação de carga e descarga”. As vias urbanas ou rurais, onde o trânsito ocorre (BRASIL, 1997, artigo 2º) são as ruas, as avenidas, os logradouros, os caminhos, as passagens, as estradas, as rodovias, as praias abertas à circulação pública e as vias internas pertencentes aos condomínios constituídos por unidades autônomas.

No trânsito são identificados três (3) componentes principais (GARBER e HOEL, 1999, p.41; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p.

3-4, KENNEDY, KELL e HOMBURGER, 1966): o ser humano; o veículo; e a via urbana ou rural. As características e as limitações de cada um destes componentes associados com os relacionamentos entre si, e, também, a influência que cada um pode exercer nos demais, interferem na eficiência do fluxo de veículos de uma via (GARBER e HOEL, 1999, p.41).

As características, as limitações e os relacionamentos entre os componentes do trânsito podem interferir em cada um dos três (3) elementos primários do tráfego da via (HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p.177):

- O fluxo, que é equivalente a razão entre os veículos que passam em um ponto da via durante um período de tempo inferior a uma hora, que pode ser determinada por:

$$q = \frac{3600.n}{t} \tag{2.1}$$

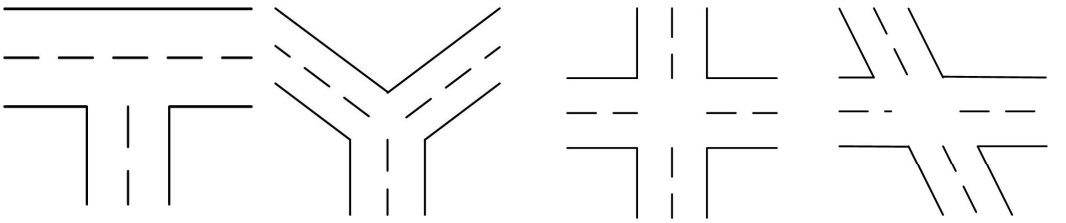
  onde:
  - *n* é o número de veículos passados no ponto da via em *t* segundos.
  - *t* é o tempo em segundos.
  - *q* é o equivalente ao fluxo em uma hora.
- A densidade ou concentração, que é a quantidade de veículos trafegando em um intervalo da via em um determinado tempo *t* (normalmente, um minuto). A unidade de densidade é veículos por metros (*vpm*).
- A velocidade, que é a distância percorrida por um veículo durante uma unidade de tempo *t*. Ela pode ser expressa em quilômetros por hora (*km/h*), por exemplo.

O relacionamento matemático entre estes elementos primários auxilia o engenheiro de trânsito no planejamento, no projeto e na avaliação do trafego, e, conseqüentemente, na melhoria do transito através da redução do tempo de retenção do fluxo de veículos numa via, seja urbana ou rural (HOMBURGER,

HALL, LOUTZNHEISER e REILLY, 1989, p.177). Esta redução implica na determinação da extensão ideal da pista que garanta um trânsito livre, bem como das possíveis intersecções da mesma com outras vias. A eficiência operacional da rede viária depende muito do projeto das intersecções (ANTP, 2000, p.102).

A intersecção ou cruzamento entre vias é a área compartilhada por duas ou mais vias, que tem a função principal de permitir a alteração da direção da rota dos veículos(GARBER e HOEL, 1999, p.219) e depende do volume e da composição do tráfego, do volume de pedestres e suas condições de travessia, da topologia do local e das condições de segurança (ANTP, 2000, p.102). O cruzamento entre vias pode ser simples ou complexa (GARBER e HOEL, 1999, p.219):

- Uma intersecção é considerada simples quando é compartilhada por apenas duas (2) vias e pode ter três (3) pernas formando um T (Figura 2-1.a) ou formando um Y (Figura 2-1.b), ou quatro (4) pernas formando um ângulo reto entre as vias (Figura 2-1.c) ou formando um ângulo obtuso (Figura 2-1.c) (GARBER e HOEL, 1999, p.219; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 20-1).

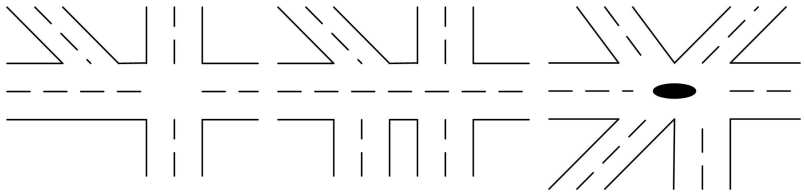


a) Com três (3) pernas em T. b) Com três (3) pernas em Y. c) Quatro (4) pernas em ângulo reto d) Quatro (4) pernas em ângulo obtuso

Figura 2-1- Intersecção entre duas (2) vias

- A intersecção complexa é aquela compartilhada por três (3) ou mais vias. Neste caso, pode ter cinco (5) (Figura 2-2.a) ou mais pernas

(Figura 2-2.b), ou ser uma rotatória[4] (Figura 2-2.c) (GARBER e HOEL, 1999, p.219; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 20-1).



a) Com cinco (5) pernas. b) Com seis (6) pernas. c) Rotatória

Figura 2-2 – Intersecção de três (3) ou mais vias

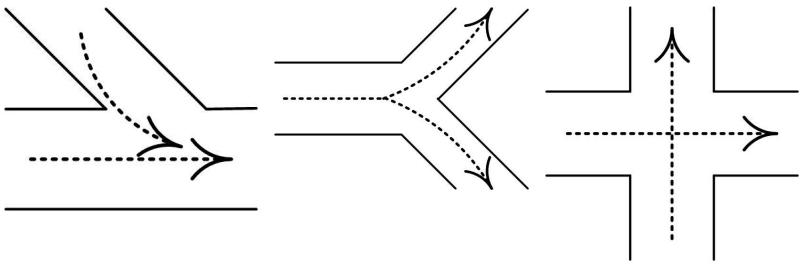
Em uma intersecção, os movimentos dos veículos podem ser conflitantes, ou seja, não podem ser realizados simultaneamente, (DENATRAN, 1984, p.13). Para determinar a existência ou não de movimentos conflitantes entre as vias do cruzamento deve-se estabelecer o número de possíveis pontos de conflito na via, sendo necessário, para tal, conhecer (GARBER e HOEL, 199, p.285):

- A quantidade de vias que a compõe;
- Os possíveis sentidos do fluxo de veículos de cada via; e
- O tipo de controle de trafego existente.

Após a identificação do número de possíveis pontos de conflito, é necessário determinar qual o tipo de conflito que ocorre na intersecção analisada. Segundo Garber e Hoel (1999, p.285) pode haver três (3) tipos de conflitos em um cruzamento:

[4] Rotatória é um obstáculo, geralmente no formato circular ou de uma elipse, que os veículos de uma via devem contornar para continuar ou modificar a sua trajetória (GARBER e HOEL, 1999, p.219; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p.20-1).

- A fusão ou convergência dos fluxos de veículos das vias ocorre quando o fluxo de uma via, a principal, recebe o fluxo das outras vias, as secundárias (Figura 2-3.a);
- A divergência entre os fluxos de veículos de uma via ocorre quando o mesmo é dividido em para outras vias, onde cada uma delas tem um sentido próprio e contrário as demais (Figura 2-3.b); e
- O cruzamento entre os fluxos das vias, quando o fluxo de uma via necessita transpor uma outra via para continuar o movimento (Figura 2-3.c).



a) Fusão ou convergência. b) Divergência. c) Cruzamento

Figura 2-3 – Tipos de Conflitos

Existindo conflito, será necessário estabelecer regras de controle de direito de passagem (DENATRAN, 1984, p.13). Há vários métodos ou regras de controle dos conflitos de filas nas intersecções. Cada um dos métodos ou regras depende do tipo de intersecção e do volume de tráfego (DENATRAN, 1984, p. 13; GARBER e HOEL, 1999, p.287; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 20-1).

Quando o volume de tráfego é considerado baixo, independente do tipo de intersecção, os conflitos entre os veículos são na maioria dos casos resolvidos pelos próprios condutores, considerando uma regra simples: “o primeiro veículo a chegar no cruzamento é o primeiro a atravessar” (DENATRAN, 1984, p.13 GARBER e HOEL, 1999, p.287; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 21-1).

À medida que o fluxo de veículos aumenta, a regra simples, apresentada anteriormente, não é mais respeitada, havendo disputa pelo direito de passagem, o que pode ser motivo de discórdia e discussão. Este problema é solucionado por outra regra simples: “escolher entre as duas vias a que será a transversal (a de menor prioridade), que deve parar e ceder a sua vez de passagem para a principal. Este esquema pode ser obtido através de colocação de placas e pintura no solo com a palavra ‘PARE’” (DENATRAN, 1984, p.13 GARBER e HOEL, 1999, p.287; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 21-1).

Quando o volume do tráfego da via principal é muito intenso, os veículos que circulam pela via transversal sofrem atraso, pois são obrigados a esperar um tempo maior para poder atravessar o cruzamento. Este problema será resolvido por uma outra regra: definição de uma ordenação seqüencial e cíclica de passagem no cruzamento, através da determinação de um tempo para realização da travessia de uma determinada via enquanto que a outra via permanece sem movimentar (DENATRAN, 1984, p.13-14; GARBER e HOEL, 1999, p.287; HOMBURGER, HALL, LOUTZNHEISER e REILLY, 1989, p. 21-1).

Uma das formas de autorizar ou não o movimento dos veículos em uma via é através do uso de sinalizações semafóricas de regulamentação, que “é um dispositivo de controle de tráfego que, através de indicações de luminosas, alterna o direito de passagem de veículos em intersecções de duas ou mais vias” (DENATRAN, 1984, p.14; GARBER e HOEL, 1999, p.295).

2.1.2 Sinalização Semafórica de Regulamentação

Um subsistema de sinalização viária, que é composto de luzes acionadas de forma alternada ou intermitentemente através de sistema elétrico/eletrônico, é denominado de sinalização semafórica e tem a função de controlar o fluxo de veículos (BRASIL, 1997, p.162) e pode ser de dois (2) tipos:

- A de advertência, que tem a função de indicar a existência de obstáculos ou situação perigosa em uma via, devendo o condutor do veículo reduzir a velocidade do mesmo e adotar as medidas de precaução compatíveis com a segurança para seguir adiante (BRASIL, 1997, p.164; ANTP, 2000, p.218).
- A de regulamentação, que tem a função de controle de trafego de veículos num cruzamento ou seção de uma via, através das indicações luminosas, alternando o direito de passagem dos vários fluxos de veículos e/ou pedestres (BRASIL, 1997, p.162; ANTP, 2000, p.218).

A sinalização semafórica de regulamentação é composta de luzes de cores dispostas em seqüência pré-estabelecidas, agrupadas em um único conjunto, o semáforo[5]. O mesmo pode ser disposto verticalmente ao lado da via ou suspenso sobre ela, podendo, neste caso, serem fixadas horizontalmente (BRASIL, 1997, p.162).

Para o controle de fluxo de veículos[6], as cores, normalmente, utilizadas nos semáforos são (Figura 2-4) (BRASIL, 1997, p.162-163; DENATRAN, 1984, p.15):

- Vermelha, que indica a obrigatoriedade de parar o veículo.
- Amarela, que indica atenção, devendo o condutor parar o veículo, a não ser que a sua parada resulte em situação de perigo para os veículos que vêm atrás;

[5] É um dispositivo de controle de tráfego que, através de indicações luminosas, alterna o direito de passagem de veículos em intersecções de duas ou mais vias (DENATRAN, 1984, p. 14).

[6] Para o controle de fluxo de pedestres (BRASIL 1997, p.162-163) utilizam-se as cores: vermelha (indicando que o pedestre não pode atravessar a via); vermelha intermitente (indicando que a fase durante a qual podem passar os pedestres está a ponto de terminar); e verde (indicando que os pedestres podem passar). Além das cores, utilizam-se, também, legendas ou figuras (DENATRAN, 1984, p.15).

- Verde, que indica a permissão do motorista prosseguir na marcha efetuando a operação indicada pelo sinal luminoso.

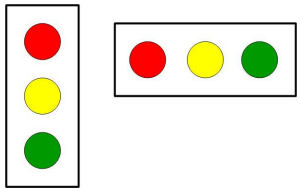


Figura 2-4 – Sinalizações Semafóricas de Regulamentação com Três (3) Cores (BRASIL, 1997, p.163)

Podem existir semáforos veiculares que utilizem apenas duas (2) cores (Figura 2-5): a verde e a vermelha. Neste caso, o comando ”amarelo” é substituído pelas duas luzes, vermelha e verde, acessas ao mesmo tempo (BRASIL, 1997, p.164).

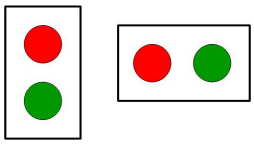


Figura 2-5 – Cores Utilizadas nas Sinalizações Semafóricas de Regulamentação (BRASIL, 1997, p.164)

Há, ainda, a possibilidade de um semáforo veicular possuir, também, a indicação da direção controlada. Neste caso, somente as luzes vermelha e verde apresentam a direção associada com a ordem (Figura 2-6).

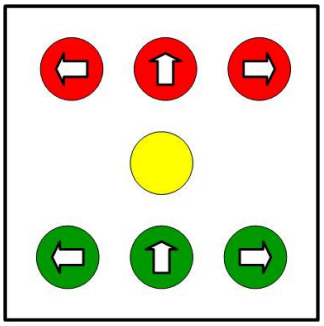


Figura 2-6 – Semáforo com Direção Controlada (BRASIL, 1997, p.164)

Independente do tipo de semáforo veicular, na sua regulagem, a seqüência de indicação do verde, amarelo e vermelho e, novamente, o verde, quando aplicada a uma via do trânsito é denominada de **fase**, e o tempo total, em segundos, para a completa seqüência de sinalização numa intersecção (todas as fases) é denominado de **ciclo**. Um dos vários períodos de um ciclo é denominado de **estágio** ou **intervalo**, e corresponde ao tempo em que uma determinada indicação luminosa dá o direito de passagem a uma corrente de tráfego e/ou pedestre, enquanto as demais ficam paradas (DENATRAN, 1984, p.16).

A determinação do número de fases de uma intersecção, que na maioria dos casos é dois (2), depende do número de aproximações ou vias, do volume de conversão e dos conflitos entre os movimentos (DENATRAN, 1984, p.16).

O controlador de tráfego é o equipamento que comanda o semáforo através do envio de pulsos elétricos para a mudança das luzes do foco, que pode trabalhar de duas formas (DENATRAN, 1984, p.26; GARBER e HOEL, 1999, p. 295):

1. Manual, quando os comandos são acionados de forma manual. A duração dos estágios, neste tipo de operação, é determinada por critérios pessoais do operador, de acordo com a situação do tráfego; e

2. Automática, quando o tempo de ciclo, a duração e instantes de mudança dos estágios são definidos pelo controlador, através de uma programação interna.

Um controlador de tráfego pode ser de três (3) categorias (DENATRAN, 1984, p.26-27):

1. Controle Isolado de Cruzamento, quando o controle dos movimentos de tráfego baseia-se no fluxo de veículos de uma única intersecção de vias. Neste caso, não há preocupação com as eventuais influências das intersecções adjacentes;
2. Controle Arterial de Cruzamento (Rede Aberta), quando o controle dos movimentos de uma via principal (corredor) visa garantir a continuidade do seu fluxo de veículos sobre as intersecções adjacentes (sistema progressivo ou onda verde); e
3. Controle de Cruzamentos em Área (Rede Fechada), quando no controle são considerados todas as intersecções sinalizadas em uma determinada área.

Há dois (2) tipos básicos de controladores (DENATRAN, 1984, p.27):

- Nos controladores de tempo fixo, o tempo de ciclo é constante, enquanto que a duração e os instantes de mudança dos estágios são fixos em relação ao ciclo. Todos os valores são parâmetros do plano de tráfego ou programação semafórica;
- Os controladores por demanda de tráfego são mais complexos que os de tempo fixo, por requisitarem detectores de veículos e lógica de decisão. O objetivo básico é dar o tempo de verde de cada corrente de tráfego de acordo com a sua necessidade, ajustando-se dinamicamente ao fluxo dos veículos que ocorrem em uma intersecção.

O bom desempenho do tráfego, em termos de fluidez e segurança, está relacionado diretamente com a regulagem de semáforos de um sistema viário, que tem o objetivo de (DENATRAN, 1984, p.61):

1. Determinar o tempo de ciclo ótimo da intersecção;
2. Calcular os tempos de verde necessários para cada fase, em função do ciclo ótimo adotado; e
3. Calcular a defasagem entre os semáforos adjacentes, se necessário.

Lo, Chan e Chow (2001) identificam três (3) tipos de estratégias de regulagem de semáforos:

1. Tempo Fixo, quando o tempo de cada ciclo e o tempo de cada uma das fases é sempre o mesmo (Figura 2-7.a);
2. Tempo de Verde Variável em Ciclo Fixo, quando o tempo de cada ciclo é fixo, mas o tempo de verde é variável de acordo com o fluxo do trânsito (Figura 2-7.b); e
3. Tempo de Verde Variável em Ciclo Não-Fixo, quando o tempo de cada ciclo e o tempo de verde é variável de acordo com as condições do tráfego (Figura 2-7.c).

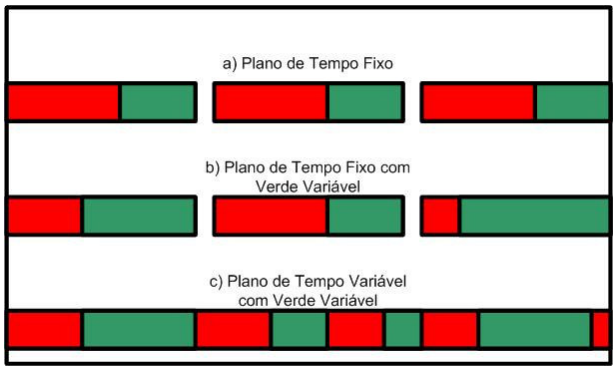


Figura 2-7 – Estratégias de Controle de Semáforos (LO, CHAN e CHOW, 2001)

Independente do tipo de controlador utilizado, a regulagem deve desenvolver planos de tráfego que efetuem da melhor maneira o controle de

veículos em uma intersecção. Para o Vilanova (2004), “o semáforo é particularmente sensível às flutuações do trânsito. Um pequeno erro de alguns segundos num tempo de verde, dependendo da situação de saturação do local, pode acarretar a formação de enormes filas de veículos”.

A impossibilidade de trabalhar com padrões pré-estabelecidos e a exigência de um ajuste fino na programação semafórica levam à conclusão de que a única forma de lidar eficientemente com o problema é a de se valer de técnicas de controle que procurem, continuamente, se adaptar às variações da realidade (VILANOVA, 2004). Para conseguir isso, é necessário, em primeiro lugar, “ler” a situação vigente do trânsito de rua, para em seguida calcular as mudanças que vão adequar as temporizações dos semáforos às variações percebidas no perfil do trânsito e, finalmente, implementar tais mudanças nos semáforos de uma forma organizada (VILANOVA, 2004), ou seja, uma regulagem em tempo-real.

2.2 Rede Neural Artificial

A arquitetura de uma RNA (BRASIL, 2002, p.6), a que será adotada neste trabalho, terá três (3) componentes:

1. O número de elementos de processamento ou neurônios;
2. O modo como as conexões entre os neurônios são feitas; e
3. A neurodinâmica dos neurônios, que corresponde à função somatório, a função de transferência e as regras de aprendizado.

2.2.1 *Perceptron*

Um dos componentes da arquitetura de uma RNA é o número de nós em cada uma das camadas. O nó é também denominado de neurônio e “é a base de uma RNA” (LUGER, 2002, p.419).

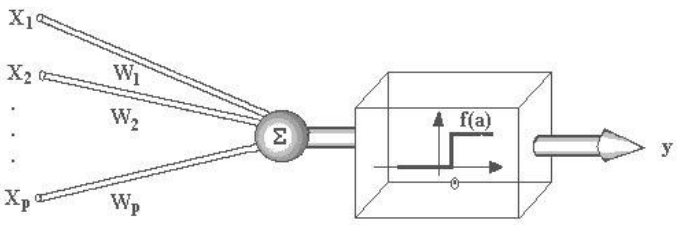


Figura 2-8 – Neurônio Artificial (LUGER, 2002, p.419)

O *percepton* (Figura 2-8), segundo Rich e Knight (1993, p.566), foi um dos primeiros modelos de RNA, e modela um único neurônio que contém os seguintes elementos (LUGER, 2002, p.419-420; HAYKIN, 2001,p.36-37; BRAGA, CARVALHO e LUDEMIR, 2000, p.14; RICH e KNIGHT, 1993, p.566):

1. Um conjunto de sinapses ou elos de conexão, cada uma delas caracterizada por um peso ou força própria. Especificamente, um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj}. É importante notar que a maneira como são escritos os índices do peso sináptico $w_{kj,}$, o primeiro índice se refere ao neurônio em questão e o segundo se refere ao terminal de entrada da sinapse à qual o peso se refere. Ao contrário de uma sinapse do cérebro, o peso sináptico de um neurônio artificial pode estar em um intervalo que inclui valores negativos bem como positivos;
2. Um somador para somar os sinais de entrada, ponderados pelas respectivas sinapses do neurônio, cujas as operações descritas constituem um combinador linear; e
3. Uma função de ativação para restringir a amplitude da saída de um neurônio. A função de ativação é também referida como função restritiva já que limita o intervalo permissível de amplitude do sinal de saída a um valor finito. Tipicamente, o intervalo normalizado da amplitude da saída de um neurônio é escrito como o intervalo unitário fechado [0,1] ou alternativamente [-1,1].

O modelo neural inclui também um *bias* aplicado externamente, representado por b_k. O *bias* b_k tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente (HAYKIN, 2001, p. 37).

2.2.2 Número de Neurônios

Uma RNA é composta por neurônios artificiais e cada um dos neurônios possui um conjunto de elos de conexões, um somador e uma função de ativação. Além disto, um *bias* que pode aumentar ou reduzir a ativação do neurônio.

A conexão é uma linha única de comunicação que vai de um neurônio que recebe uma informação a outro que envia esta mesma informação, podendo ser de excitação ou inibição. Nas conexões de inibição há a tendência de inibir a ativação do neurônio, enquanto que nas conexões de excitação há a tendência de ativação do neurônio (BRASIL, 2002, p. 8).

Além das conexões, os neurônios da rede são agrupados em camadas e de acordo com o número de camadas, uma RNA pode ser (HAYKIN, 2001, p.47; BRAGA, CARVALHO e LUDEMIR, 2000, p.11; BRASIL, 2002, p. 7):

- De camada única, quando só existe um nó entre qualquer entrada e qualquer saída da rede; e
- De múltiplas camadas (*Multilayer Percepton* - MLP), quando existe mais de um neurônio entre alguma entrada e alguma saída da rede. Esta camada intermediária entre a de entrada e a de saída, é denominada de camada oculta e pode existir uma ou mais dela (HAYKIN, 2001, p.47).

Numa MLP, o número de camadas intermediárias depende do número de exemplos de treinamento, da quantidade de ruídos presente no exemplo, da complexidade da função a ser aprendida e da distribuição estatística de

treinamento. Em outras palavras, o tamanho de uma RNA é definido pelo número de camadas escondidas e pelo número de processadores (nós ou neurônios) de cada uma das camadas (BRAGA, CARVALHO e LUDEMIR, 2000, p.55).

Braga, Carvalho e Ludemir (2000, p.55) relatam que o número de nós nas camadas intermediárias, em geral, é definido empiricamente. Para Kovács (2002, p.107), de acordo com o Teorema de Kolmogorov-Nielsen, em uma rede neural de três (3) camadas, com dimensão *n* na camada de entrada, a camada oculta pode ser composta de até *2n + 1* neurônios, e a camada de saída com *m* neurônios.

2.2.3 Modo de Conexão

A conectividade de uma RNA pode ser (BRAGA, CARVALHO e LUDEMIR, 2000, p.11):

- Fracamente ou parcialmente conectada, quando cada um dos nós de uma camada (exceto, aos da camada de saída) não está ligado a todos os nós da camada posterior a sua; e
- Fortemente ou completamente conectada, quando cada um dos nós de uma camada (exceto, aos da camada de saída) está ligado a todos os nós da camada posterior a sua.

Além da definição da conexão existente entre cada um dos neurônios de uma camada em relação a sua camada posterior, é necessário, para uma RNA, o estabelecimento da sua topologia que pode ser (BRAGA, CARVALHO e LUDEMIR, 2000, p. 11-12):

- *Feedforward* ou acíclica (a saída do neurônio na *i-ésima* camada da rede não pode ser usada como entrada de nós em camadas de índice menor ou igual a *i*) são redes cuja saída final (única) não possui qualquer ligação com as entradas.

- *Feedback* ou cíclica (a saída de algum neurônio na *i-ésima* camada da rede é usada como entrada de nós em camadas de índice menor ou igual a *i*), são redes cuja saída final (única) é ligada às entradas que se comportam como autômatos reconhecedores de cadeias, onde a saída, que é realimentada, fornece o estado de autômato. Se todas as ligações são cíclicas, a rede é denominada de auto-associativa. Estas redes associam a um padrão de entrada com ele mesmo e são particularmente úteis para recuperação ou regeneração de um padrão de entrada.

Um dos exemplos de uma RNA *feedback* ou cíclica, segundo Haykin (2001, p. 65), e Braga, Carvalho e Ludemir (2000, p.75), é a Rede Neural Artificial Recorrente – RNR. Este tipo de rede é considerado como um sistema não linear e dinâmico (contém retardos). É dinâmico porque para o cálculo de um elemento em um determinado instante é necessário conhecer o mesmo elemento em um instante anterior. Os principais modelos de RNR são:

- *Backpropagation Through Time* – BPTT.
- *Real-Time Recurrent Learning* – RTRL.
- *Real-Time Recurrent Networks* – RTRN.
- *Cascate Correlation Recurrent* – CCR.

O modelo de RNR *Backpropagation Through Time* – BPTT (Figura 2-9) pode ser implementado a partir de uma RNA *Backpropagation*, que é estática, com a utilização de janelas de tempo, em que a entrada da rede utiliza trechos dos dados temporais como se eles formassem um padrão estático (BRAGA, CARVALHO e LUDEMIR, 2000, p.209; HAYKIN, 2001, p. 49).

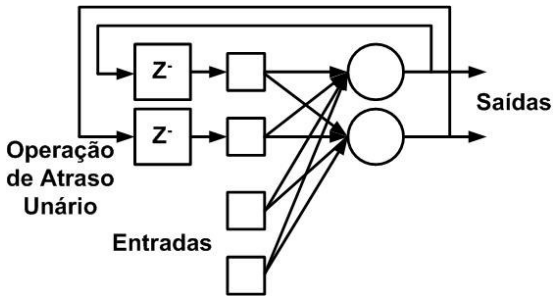


Figura 2-9 – Rede Neural Recorrente *Backpropagation Through Time* – BPTT (HAYKIN, 2001, p.49)

Dentro do escopo deste trabalho, o modelo adotado de RNR será o BPTT, que melhor será explicado nos capítulos posteriores. Quanto aos demais modelos fogem do escopo dessa pesquisa.

2.2.4 Neurodinâmica

Dos componentes da RNA, o número de elementos e o modo de conexão entre os neurônios têm o objetivo de determinar quanto de detalhes a rede pode manipular, mas a neurodinâmica, o último componente, afeta os modos nos quais a rede processará os dados e como está relacionada, com freqüência, com a estratégia de conexão utilizada. A neurodinâmica é composta pela (BRASIL, 2002, p. 7):

- Função somatório;
- Função de transferência; e
- Regras de aprendizado

A função somatório (BRASIL, 2002, p.7) tem o objetivo de calcular a estimulação interna ou nível de ativação (função de ativação) do neurônio através do somatório do produto de cada uma das entradas multiplicada pelo seu respectivo padrão de conexão (peso).

De acordo com o nível de ativação, o neurônio exibe ou não uma saída. O relacionamento entre o nível de ativação interna e a saída pode ser linear ou não. Essas relações são expressas por uma função de transferência, que pode ser dos mais variados tipos diferentes. Além disso, a seleção de uma função específica determina qual a operação da rede, juntamente com o objetivo de modificar os níveis de saída para um valor razoável (BRASIL, 2002, p. 8).

2.2.4.1 Regra de Aprendizagem das RNA

Além da função somatório e da função de ativação, o último componente da neurodinâmica é a regra de aprendizado.

A regra de aprendizagem de uma RNA é o conjunto preestabelecido de procedimentos ou regras bem-definidas para adaptar os parâmetros de uma RNA de modo que ela possa aprender uma determinada função e solucionar um problema de aprendizagem. No entanto, não há um único algoritmo, mas sim um conjunto de ferramentas representadas por diversos algoritmos, cada qual com suas vantagens e desvantagens. Os algoritmos basicamente diferem pela maneira pela qual o ajuste dos pesos é feito (BRAGA, CARVALHO e LUDEMIR, 2000, p.15).

APRENDIZADO

A principal propriedade de uma RNA é a sua capacidade de aprender a partir do seu ambiente e de aprimorar o seu desempenho através da aprendizagem. O desempenho é melhorado com o tempo de acordo com alguma medida preestabelecida. A RNA aprende através de um processo interativo no qual ocorre ajuste nos pesos sinápticos e no nível do *bias*, tornando-se mais instruída no seu ambiente após cada iteração do processo de aprendizagem (BRAGA, CARVALHO e LUDEMIR, 2000, p.15-16).

Antes de apresentar o conceito de aprendizagem em uma RNA é importante entender que no aprendizado conexionista, não se procura obter

regras como na abordagem simbólica da Inteligência (IA), mas sim determinar a intensidade de conexões entre neurônios.

A aprendizagem é:

- O processo pelo qual os parâmetros livres de uma RNA são adaptados através do processo de estimulação pelo ambiente no qual a rede está inserida. Este tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre (HAYKIN, 2001, p.75).
- "O processo pelo qual os parâmetros de uma RNA são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizado definido pela maneira particular, como ocorrem os ajustes realizados nos parâmetros" (BRAGA, CARVALHO e LUDEMIR, 2000, p.15).

Através das duas definições do processo de aprendizagem, descritas anteriormente, é possível inferir que:

1. O ambiente estimula a RNA;
2. Os parâmetros na RNA são modificados a partir das estimulações do ambiente; e
3. A RNA responde de uma nova maneira ao ambiente, de acordo com as alterações ocorridas na sua estrutura interna.

O conjunto preestabelecido de regras bem-definidas para a solução de um problema de aprendizagem é denominado de algoritmo de aprendizagem. Segundo Haykin, não há um único algoritmo de aprendizado para o projeto de RNA, pois a diferença entre um algoritmo e outro está na maneira como é formulado o ajuste de um peso sináptico de um neurônio e pela maneira pela qual a rede neural (máquina de aprendizagem), constituída de um conjunto de

neurônios interligados, se relaciona como seu ambiente - o paradigma de aprendizagem (HAYKIN, 2001, p.76).

Para Braga, Carvalho e Ludemir (2000, p.15) e Haykin (2001, p.76), há diversos métodos ou regras para treinamento de rede que foram desenvolvidos: aprendizagem por correção; aprendizagem baseada em memória; aprendizagem hebbiana; aprendizagem competitiva; aprendizagem de Boltzmann.

Esses métodos ou regras listadas anteriormente podem ser agrupados em dois paradigmas principais de aprendizagem: o aprendizado supervisionado ou aprendizagem com um professor; e aprendizado não-supervisionado ou aprendizagem sem um professor (BRAGA, CARVALHO e LUDEMIR, 2000, p. 15-16; HAYKIN, 2001, p.76-77).

Aprendizado Supervisionado

O aprendizado supervisionado ou aprendizado com um professor (Figura 2-10) é o método mais comum de treinamento de RNA com neurônios com pesos ou com neurônios sem peso. A denominação de aprendizado supervisionado é porque existe um supervisor (professor) externo que fornece a entrada e a saída desejadas para a rede com o objetivo de ajustar os parâmetros da rede, de forma a encontrar uma ligação entre os pares de entrada e saída fornecidos (BRAGA, CARVALHO e LUDEMIR, 2000, p. 17).

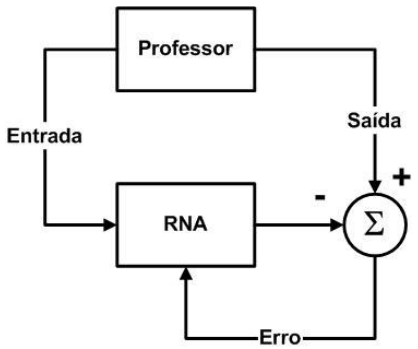


Figura 2-10 – Aprendizado Supervisionado (BRAGA, CARVALHO e LUDEMIR, 2000, p. 17)

Na aprendizagem com supervisor, o professor indica explicitamente um comportamento bom ou ruim para a rede, com o objetivo de direcionar o processo de treinamento. A rede tem sua saída corrente (calculada) comparada com a saída desejada (que representa uma ação ótima para ser realizada pela rede) com resposta calculada, ajustando-se os pesos das conexões para minimizar o erro. A minimização da diferença é incremental, já que pequenos ajustes são feitos nos pesos a cada etapa de treinamento, de tal forma que caminhem - se houver solução possível - para uma solução. A soma dos erros quadráticos de todas as saídas é normalmente utilizada como medida de desempenho da rede e também como função de custo a ser minimizada pelo algoritmo de treinamento (HAYKIN, 2001, p.16-17).

A desvantagem do aprendizado supervisionado é que, na ausência do professor, a rede não conseguirá aprender novas estratégias para situações não cobertas pelos exemplos do treinamento da rede. Os exemplos mais conhecidos de algoritmos para aprendizado supervisionado são a regra delta e sua generalização para redes de múltiplas camadas, o algoritmo *backpropagation* (HAYKIN, 2001, p. 16-17; BRAGA, CARVALHO e LUDEMIR, 2000, p.17-18).

O algoritmo *backpropagation* tem duas fases (BRAGA, CARVALHO e LUDEMIR, 2000, p.59-61): a *Forward* e a *Backward*.

- Na *Forward*, o algoritmo executa os seguintes procedimentos:

    1. A entrada é apresentada à primeira camada da rede, a camada inicial C_0.
    2. Para cada camada C_i a partir da camada de entrada:

- Após os nodos da camada C_i ($i > 0$) calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C_{i+1}.
    3. As saídas produzidas pelos nós da última camada são comparados às saídas desejadas.

- Na *Backward*, o algoritmo executa os seguintes procedimentos:

    1. A partir da última camada, até chegar a camada de entrada:

- Os nós da camada atual ajustam seus pesos de forma a reduzir os erros.
- O erro de um nó das camadas intermediárias é calculado utilizando os erros dos nós da camada seguinte conectados a eles, ponderada pelos pesos das conexões entre elas.

Na implementação do algoritmo *backpropagation* é necessário executar os seguintes passos (BRAGA, CARVALHO e LUDEMIR, 2000, p. 61):

1. Iniciar os pesos e os parâmetros.
2. Definir as saídas da rede através da fase *forward*.
3. Comparar as saídas produzidas com as saídas desejadas.
4. Atualizar os pesos dos nós através da fase *backward*.

Há duas formas básicas de implementação do algoritmo com professor: *off-line* e *on-line*. No treinamento *off-line*, os dados do conjunto de treinamento não mudam e uma vez obtida uma solução para a rede, esta deve permanecer

fixa. Caso novos dados sejam adicionados ao conjunto de treinamento, um novo treinamento, envolvendo também os dados anteriores, deve ser realizado para evitar interferência no treinamento anterior. Por sua vez, no aprendizado *on-line*, o conjunto de dados muda continuamente, e a rede deve estar em contínuo processo de adaptação.

Aprendizado Não-Supervisionado

O aprendizado supervisionado, descrito no item anterior, necessita de um professor ou instrutor que fornece a entrada e saída desejada para garantir a aprendizagem da RNA. A ausência do supervisor é considerada uma desvantagem para este paradigma, pois a rede não conseguirá aprender novas estratégias para situações não identificadas nos exemplos do treinamento da rede.

Uma alternativa para este problema é o aprendizado sem supervisor ou aprendizado não supervisionado (Figura 2-11). Neste tipo de aprendizado não existe a necessidade do professor ou do supervisor para acompanhar o processo de aprendizado. Apesar da semelhança entre o aprendizado supervisionado e o aprendizado dos seres humanos, é importante entender que muitos dos sistemas biológicos aprendem sem supervisor, dentre os quais podemos destacar os estágios iniciais dos sistemas de visão e o de audição. (BRAGA, CARVALHO e LUDEMIR, 2000, p. 19). Nesses algoritmos, somente os padrões de entrada estão disponíveis para a rede, ao contrário do aprendizado supervisionado, cujo conjunto de treinamento possui pares de entrada e saída.

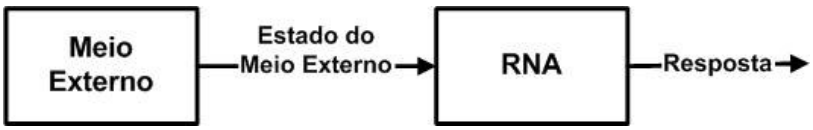


Figura 2-11 – Aprendizado Não Supervisionado (BRAGA, CARVALHO e LUDEMIR, 2000, p. 19)

Segundo Braga, Carvalho e Ludemir (2000, p.19), a partir do momento em que a rede estabelece uma harmonia com as regularidades estatísticas da entrada de dados, desenvolve-se nela uma habilidade de formar representações internas para codificar características de entrada e criar novas classes ou grupos automaticamente. Este tipo de aprendizado só se torna possível quando existe redundância nos dados de entrada. Sem redundância seria impossível encontrar quaisquer padrões ou características dos dados de entrada.

A estrutura do sistema de aprendizado não-supervisionado pode adquirir uma variedade de formas diferentes, tais como ter uma camada de entrada, uma camada de saída, conexões *feedforward* da entrada para a saída e conexões laterais entre os neurônios da camada de saída. Outro exemplo possível é uma rede *feedforward* com múltiplas camadas, em que a livre organização procede na base de camada por camada. Nestes dois exemplos, o processo de aprendizado consiste em modificar repetidamente o peso sináptico de todas as conexões do sistema em resposta às entradas (BRAGA, CARVALHO e LUDEMIR, 2000, p. 19).

2.3 Controle

Controlar é atuar sobre as grandezas[7] envolvidas nas interações do sistema[8] com o meio ambiente[9], de modo que o sistema possua um comportamento adequado, de acordo com as especificações fornecidas

---

[7] As grandezas podem ser classificadas em entradas e saídas. As entradas que podem ser ajustadas são denominadas de variáveis de controle ou variáveis manipuladas, e as que não podem ser, perturbações ou ruídos (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14).

[8] Um sistema é composto por diversos componentes que interagem entre si, em conformidade com as leis da natureza (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14).

[9] O meio ambiente é tudo aquilo que não integra o sistema, mas interage com ele, alterando as suas características. As interações podem ser através de transferência de massa, energia ou mesmo de informação (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14).

anteriormente (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14), ou seja, “é assegurar a estabilidade do sistema” (DORF e BISHOP, 2001, p.229).

As grandezas envolvidas nas interações do sistema com o meio ambiente, segundo Nascimento Júnior e Yoneyama (2000, p. 14), podem ou não ser ajustadas convenientemente de acordo com as especificações dos projetistas. Essas grandezas são classificadas em entradas e saídas.

Há dois tipos de entradas: as perturbações ou ruídos, que são as entradas que não podem ser ajustadas; e as variáveis de controle ou manipuladas, que são as entradas que podem ser ajustadas (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14).

As saídas, por sua vez, são as grandezas que foram selecionadas como de interesse e, às vezes, referem-se a leitura de instrumentos de medida. O controlar é justamente atuar sobre essas grandezas de modo que o sistema possua um comportamento adequado de acordo com as especificações fornecidas (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.14), ou seja, controlar a saída da planta para que a mesma permaneça próxima a uma resposta desejada (HAYKIN e VAN VEEN, 2001, p.551). Por isso, segundo Haykin e Van Veen (2001, p.26), os dois mais importantes motivos para o uso de controles para a engenharia são a obtenção de uma resposta satisfatória e o desempenho robusto:

- Resposta: diz-se que uma planta[10] produz uma resposta satisfatória se sua saída segue ou acompanha uma entrada de referência específica. O processo de manter a saída da planta próximo da entrada de referência é denominado de regulação.
- Robustez: Diz-se que um sistema de controle é robusto se ele exibe boa regulação, apesar da presença de perturbações externas e na

[10] A planta é o objeto a ser controlado (HAYKIN e SIMON, 2001, p.26).

presença de mudança nos parâmetros da planta devido a condições ambientais variáveis.

Segundo Haykin e Van Veen (2001, p.551), Dorf e Bishop (2001, p.2-3), Phillips e Harbor (1996, p.1-4), Ogata (2003, p.5), Da Silveira e Dos Santos (2004, p.23-24), há dois (2) tipos básicos de controle: o controle de malha aberta; e o controle de malha fechada.

- O primeiro, o controle de malha aberta (Figura 2-12), utiliza um dispositivo de atuação para controlar diretamente o processo sem usar a retroação (DORF e BISHOP, 2001, p.2), ou seja, a modificação da entrada da planta $c(t)$ é derivada diretamente da resposta alvo desejada $y_d(t)$ (HAYKIN e VAN VEEN, 2001, p.551). Este tipo de controle tem dois (2) componentes conectados entre si: o controlador; e a planta.

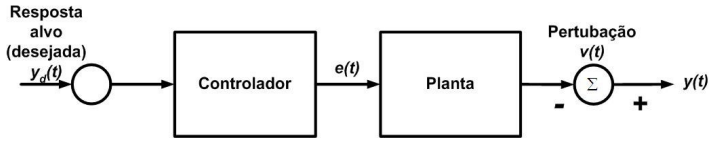


Figura 2-12 – Controle de Malha Aberta (Baseado em: DORF e BISHOP, 2001, p. 3; HAYKIN, 2001, p.551; HAYKIN e VAN VEEN, 2001, p. 26)

- Já o segundo, o controle de malha fechada ou realimentação (Figura 2-13), usa uma medida de saída e a retroação ou realimentação deste sinal para compará-lo com a saída desejada (referência ou comando) (DORF e BISHOP, 2001, p.3; HAYKIN e VAN VEEN, 2001, p.551).

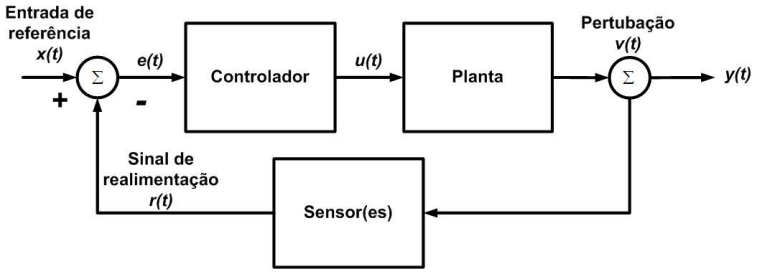


Figura 2-13 – Controle de Malha Fechada (Baseado em: DORF e BISHOP, 2001, p. 3; HAYKIN, 2001, p.551; HAYKIN e VAN VEEN, 2001, p. 26)

Segundo Haykin e Van Veen (2001, p.540), um sistema de realimentação na sua forma mais básica consiste de três (3) componentes conectados entre si para formar uma única malha de realimentação:

- Uma planta, a qual age a partir de um sinal de erro $e(t)$ para produzir o sinal de saída $y(t)$.
- Um sensor, o qual mede o sinal de saída $y(t)$ para produzir o sinal de realimentação $r(t)$.
- Um comparador, o qual calcula a diferença do sinal de entrada $e(t)$ aplicado externamente (referência) e o sinal de realimentação $r(t)$ para produzir o sinal de erro $e(t)$: $e(t) = x(t) - r(t)$.

Os componentes conectados do sistema de controle com realimentação estabelecem uma relação de comparação entre a saída $y(t)$ e a entrada de referência $x(t)$, utilizando a diferença $e(t)$ como meio de controle, tendo como conseqüência a redução do erro do sistema (OGATA, 2003, p.5), ou seja, tornar o sistema estável[11] (DORF e BISHOP, 2001, p.229).

Dependendo da quantidade de entradas e/ou saídas do sistema, os tipos básicos de controle, a malha aberta e a malha fechada, podem ser um sistema de entrada única/saída única (SISO – *Single-Input/Single-Output*), quando

[11] Um sistema estável deve apresentar uma saída limitada se a entrada correspondente for limitada (DORF e BISHOP, 2001, p.229).

possui uma única entrada de planta e uma única saída da planta, ou um sistema de múltiplas entradas/múltiplas saídas (MIMO – *Multiple-Input/Multiple-Output*), quando o número de entradas da planta e/ou o número de saídas da planta (HAYKIN, 2001, p.27).

Segundo De Azevedo, Brasil e De Oliveira (2000, p.304), e De Oliveira (1999, p.304), há duas abordagens de controle para tratar a questão de desempenho do controlador, quando a dinâmica do processo é desconhecida: o controle robusto; e o controle adaptativo (item 2.3.1).

Independente de ser robusto ou adaptativo, o controle pode ser: automático, "quando é realizado com pouca ou nenhuma intervenção humana" (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.15); ou manual, quando necessita que um operador humano assista constantemente a operação do sistema (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.15).

Um controle automático poderá permitir que as especificações de segurança, de produtividade, de qualidade, de conforto e outras, sejam alcançadas (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.15).

Para Nascimento Júnior e Yoneyama (2000, p.22), uma das vantagens do uso de controladores automáticos é "a melhoria da confiabilidade no controle de processos". Os dados são capturados pelos sensores[12], em um determinado tempo, e são processados para produzirem saídas aplicadas nos transdutores.

O problema de controle é "obter uma estratégia de atuação sobre o sistema, de modo que este se comporte de forma conveniente. A especificação

---

[12] Interface perceptiva entre o controlador e o ambiente. Os sensores podem ser: passivos (como câmeras, simples observadoras do ambiente: eles captam sinais gerados por outras fontes no ambiente); ou ativos, como o caso do sonar, que enviam energia ao ambiente e utilizam a reflexão da energia para observarem o ambiente (RUSSEL e NORVIG, 2004, p.872).

do comportamento desejado pode envolver conceitos de estabilidade, rejeição de distúrbios (ruídos), robustez a incertezas no modelo, forma de resposta do sistema a entrada padrão, simplicidade de implementação, custo operacional" (NASCIMENTO JÚNIOR e YONEYAMA, 2000, p.23). Por isso, para definir um problema de controle é necessário responder as seguintes perguntas (DE AZEVEDO, 1993, p.121):

1. O que o sistema deve controlar? (a planta –*plant*).
2. Qual a performance do sistema? (a meta – *goal*).
3. Quais as vias (*ways*) de controle do sistema? (o controlador – *controller*).
4. Como os controladores atuam no sistema? (*the control topology*).

No entanto, para modelar um sistema, ou seja, responder as perguntas acima, pode-se ter várias situações (OLIVEIRA JR, 1999, p. 11):

- Aplicando as mesmas entradas em momentos diferentes, é possível afirmar que as saídas serão idênticos em ambos os casos e que a relação entre as duas segue leis dinâmicas sem elementos probabilísticos. Esta situação é denominada de determinística.
- Aplicando as mesmas entradas em momentos diferentes, somente é possível determinar a probabilidade de que determinada variável assuma um ou outro valor, ou seja, é um caso de incerteza e para tratá-la utilizam-se hipóteses simplificadoras. Esta situação é denominada de estocástica.

Para ambas situações, a determinística e a estocástica, para os controladores convencionais, é necessário o desenvolvimento de equações diferenciais para descrever o sistema (FERNANDEZ, PEDROZA e DE REZENDE, 2003, p. 176).

No entanto, há casos onde não é possível enquadrar satisfatoriamente o objeto de estudo em modelos matemáticos, pois a imprecisão é semântica e/ou

estrutural (OLIVEIRA JR, 1999, p.12), ou seja, onde o conhecimento só pode ser expresso de forma intuitiva, usando as variáveis lingüísticas (FERNANDEZ, PEDROZA e DE REZENDE, 2003, p. 176). Para tanto, deve utilizar, ao invés de um controlador convencional, um controlador *fuzzy*.

2.3.1 Controle Adaptativo

Um controle adaptativo ou sistema dinâmico apresenta parâmetros do controlador variáveis que são ajustados, a cada iteração, no sentido de respeitar um dado critério de desempenho. Em geral, a abordagem adaptativa é aplicada a uma ampla faixa de imprecisão sendo mais difíceis de implementar do que os controladores robustos, por apresentarem ajustes temporais nos parâmetros do controlador quando ocorrem variações no processo (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.304).

Um sistema de controle dinâmico ou adaptativo é utilizado em várias áreas (SCHOPPEK, 2001, p. 2), como exemplo, o controle do vôo de aeronave; da temperatura de forno industrial; ou dos aparelhos de uma unidade de tratamento intensivo. Em todos os exemplos, o ambiente é dinâmico.

Em um ambiente dinâmico, a cada alteração da situação do ambiente é necessário que um operador humano execute, imediatamente, uma ação com o objetivo de manter a estabilidade do ambiente (SCHOPPEK, 2001, p.2).

As aplicações listadas onde um controle adaptativo pode ser utilizado são exemplos de aplicações que necessitam de estabilidade. E para garantir esta estabilidade do sistema de controle adaptativo, segundo De Oliveira (1999, p.135), é necessário algum conhecimento "*a priori*" sobre o processo a ser controlado. Este conhecimento "*a priori*" também é necessário para implementar os identificadores ou observadores que têm a mesma ordem do processo.

Para Borges e De Azevedo (1993, p.1), a aplicação de algoritmos de controle com base na teoria dos conjuntos *Fuzzy* pode ser considerado como um controle adaptativo, baseado em processos lingüísticos, obtidos pela experiência própria e regras heurísticas usadas por operadores humanos (BORGES e DE AZEVEDO, 1993, p.1). O detalhamento deste tipo de controlador, o Controle *Fuzzy*, será apresentado no próximo item.

2.3.2 Controlador *Fuzzy*

Um controlador *fuzzy* é um sistema baseado em regras *fuzzy*[13] e possui, basicamente, quatro componentes principais (Figura 2-14) (DIAS e BARROS, 2005, p. 148):

- Um processador de entrada (ou *fuzzificador*) ou a interface escalar-*fuzzy*;
- Um conjunto de regras lingüísticas ou sistema de tomada de decisões;
- Um método de inferência *fuzzy* ou base do conhecimento*;* e
- Um processador de saída (ou *defuzzificador*) ou a interface *fuzzy*-escalar.

[13] Para Oliveira Jr (1999, p.5), a regra *fuzzy* é o conjunto de métodos baseados no conceito de conjunto difuso (*fuzzy set*) e operações difusas, que possibilita uma modelagem realista e flexível de um sistema.

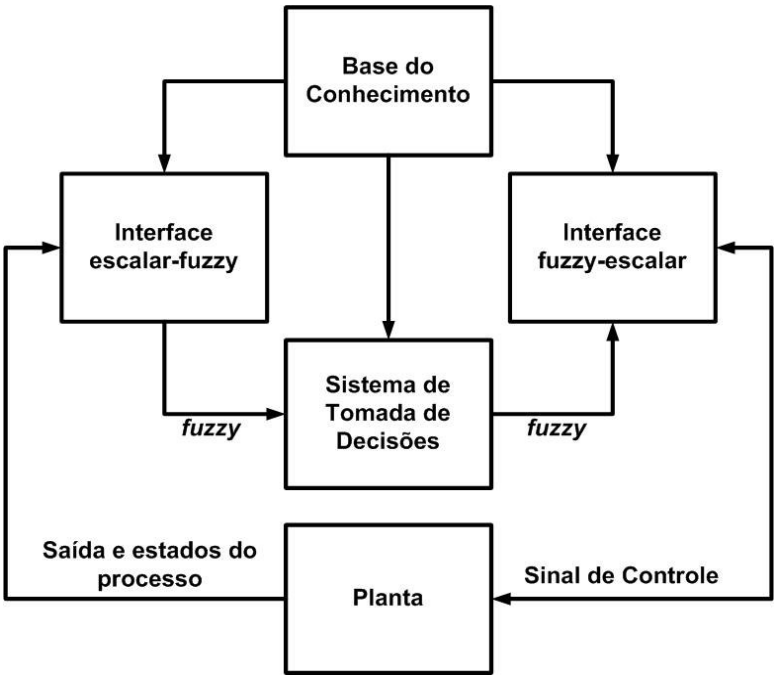


Figura 2-14 – Controlador *Fuzzy* (DIAS e BARROS, 2005, p. 148; HAYKIN, 2001, p.551)

Em um controlador *fuzzy*, o conhecimento pode ser expresso de forma intuitiva, usando as variáveis lingüísticas. As aplicações ideais para utilização de Controladores *Fuzzy* são as seguintes (FERNANDEZ, PEDROZA e DE REZENDE, 2003, p.176):

- Processos muito complexos, onde não há modelo matemático claro;
- Processos altamente não lineares ou com comportamento probabilístico;
- Aquelas em que o processamento de conhecimento especializado (formulados lingüisticamente) é o único possível.

A aplicação de algoritmos de controle com base na teoria dos conjuntos *Fuzzy* pode ser considerada como um controle adaptativo, baseado em processos lingüísticos, obtidos pela experiência própria e regras heurísticas usadas por operadores humanos (BORGES e DE AZEVEDO, 1993, p. 1).

A implementação de tal controle consiste na transformação das variáveis de entrada para uma linguagem (como negativo médio, zero, positivo grande) e o estabelecimento de uma regra de controle para que uma tomada de decisão possa produzir a saída desejada. Se necessário, essas saídas lingüísticas são transformadas em valores numéricos (BORGES e DE AZEVEDO, 1993, p.1).

A Lógica *Fuzzy* foi introduzida por Lofti Zedeh (*apud* FERNANDEZ, PEDROZA e DE REZENDE, 2003, p.175; *apud* FERNANDEZ, PEDROZA e DE REZENDE, p.4; *apud* OLIVEIRA, 1999, p.5), como uma generalização da teoria clássica. A extensão sugerida por Zadeh está na possibilidade de um determinado elemento poder pertencer a um conjunto com um valor chamado grau de pertinência. Assim, um elemento não simplesmente pertence ou não pertence a um conjunto, como na lógica clássica, mas poderá pertencer a um conjunto com grau de pertinência que varia no intervalo [0,1], onde o valor 0 indica uma completa exclusão, o valor 1 representa completa inclusão.

A lógica *fuzzy* utiliza as variáveis lingüísticas ao invés de variáveis numéricas (Figura 2-15). Variáveis lingüísticas admitem como valores apenas expressões lingüísticas, como "muito grande", "pouco frio", "mais ou menos jovem", que são representadas por conjuntos *fuzzy*. A construção de um sistema de controle *fuzzy* é baseada na idéia de se incorporar a experiência ou conhecimento especializado de um operador humano para se obter a melhor estratégia de controle. Desse modo, a forma das regras empregadas depende do processo a ser controlado (FERNANDEZ, PEDROZA e DE REZENDE, 2003, p.175; FERNANDEZ, PEDROZA e DE REZENDE, 2003, p.4).

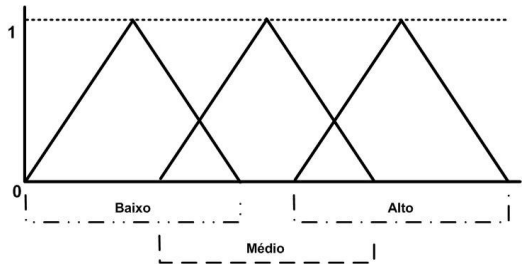


Figura 2-15 – Conjuntos *Fuzzy*

Para Fernandez, Pedroza e De Rezende (2003, p. 176), os controladores *fuzzy* são a mais importante aplicação da Teoria *Fuzzy*. Seu funcionamento é diferente dos controladores convencionais, pois estes exigem o desenvolvimento das equações diferenciais que descrevem o sistema.

A maioria dos sistemas baseados na Lógica *Fuzzy* compreende as seguintes etapas: *fuzzificação*, avaliação de regras e *defuzzificação*. A interação destas etapas é apresentada na Figura 2-16 (DE AZEVEDO, BRASIL e OLIVEIRA, 2000, p.66).

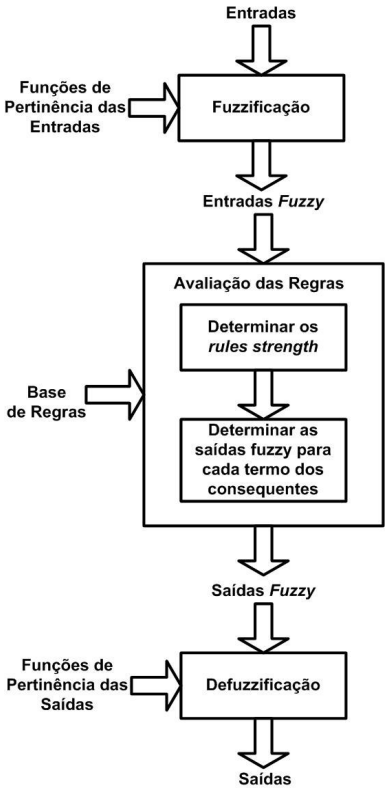


Figura 2-16 – Lógica *Fuzzy* (Baseado em DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.66)

A *fuzzificação* é o primeiro passo no processamento de um sistema *fuzzy* onde as entradas do sistema são modeladas por um conjunto *fuzzy*. Esta etapa recebe valores do meio externo e, através das funções de pertinência associadas à parte antecedente das regras, efetua o mapeamento destes

valores para graus de pertinência (BORGES e DE AZEVEDO, 1993, p.4; DE AMORIM, 2002, p.42).

A avaliação das regras, também conhecida como inferência *fuzzy*, é o segundo passo. O processo de avaliação pode ser descrito da seguinte forma (DE AMORIM, 2002, p.42):

- Encontrar o grau de pertinência (*rule strength* ou *firing strength*) para cada regra: O grau de verdade indica o grau para o qual a parte antecedente da regra é satisfeita. Seu valor é obtido através da aplicação do operador *fuzzy* presente na parte antecedente da regra;
- Determinar a saída *fuzzy*: Esta fase compreende a aplicação de um operador de agregação em todas as regras que envolvem o mesmo termo do conseqüente. O máximo e a soma disjunta são os operadores mais utilizados. Existe uma saída *fuzzy* para cada conjunto *fuzzy* da saída.

A base de regras pode ser considerada como o “núcleo" do controlador *fuzzy* e é nela que se encontram as proposições *fuzzy*, que são fornecidas de acordo com um especialista. É neste ponto que as variáveis e suas classificações lingüísticas são catalogadas e, em seguida, modeladas por conjuntos *fuzzy*, isto é, funções de pertinência. As proposições *fuzzy* aqui descritas são feitas na forma lingüística (BORGES e DE AZEVEDO, 1993, p.4; DIAS e BARROS, 2005, p. 148-149; DE AMORIM, 2002, p.43):

$$R_1 : SE\ x\ é\ A_1\ E\ y\ é\ B_1\ ENTÃO\ z\ é\ C_1$$

$$R_1 : SE\ x\ é\ A_2\ E\ y\ é\ B_2\ ENTÃO\ z\ é\ C_2$$

onde A_i, B_i e C_i são conjuntos *fuzzy.*

A *defuzzificação* é o último passo no processamento de um sistema *fuzzy*. Sua função é combinar todas as saídas *fuzzy* e gerar um resultado não *fuzzy*. Portanto, esta etapa é necessária apenas em problemas em que uma solução não *fuzzy* é requerida (DE AMORIM, 2004, p.43).

Os métodos de *defuzzificação* mais utilizados são (Cox *apud* DE AMORIM, 2004, p.43):

- Primeiro Máximo (*Smallest of Maximum* – SOM): Encontra o valor de saída através do ponto em que o grau de pertinência atinge o primeiro valor máximo;
- Média dos Máximos (*Mean of Maximum* – MOM): Encontra o ponto médio entre os valores que têm o maior grau de pertinência inferido pelas regras;
- Centro da Área (*Center of Area* – COA), Centróide ou Centro da Gravidade (*Center of Gravity* - COG): O valor de saída é o centro de gravidade da função de pertinência resultante.

Os métodos de *defuzzificação* apresentam diferenças em termos de velocidade e eficiência. Portanto, a seleção do método mais apropriado está diretamente relacionada com os requisitos da aplicação.

2.4 Gestão do Conhecimento

A Gestão do Conhecimento é um processo sistemático, articulado e intencional, apoiado na geração, codificação, disseminação, aprendizagem e aplicação de conhecimentos, com o propósito de atingir a excelência organizacional (FGVSP, 2006).

Para Davenport e Prusak (1998, p.18), o conhecimento é ação, pois o conhecimento é uma informação ajuizada, possuindo reflexão, síntese e contexto, útil para a satisfação de uma meta e reside na mente de uma pessoa: é uma interpretação, ajuizada como útil para ação. Conhecimento é uma ação.

O conhecimento pode ser classificado (NONAKA e TAKEUCHI, 1997, p. XIII):

- Explícito, que é o conhecimento que pode ser expresso em palavras e números, sendo fácil e prontamente comunicado e compartilhado entre pessoas sob a forma de dados brutos, fórmulas científicas, procedimentos codificados, banco de dados, patentes, relacionamento com clientes ou princípios universais. É o conhecimento existente nos manuais e normas, por exemplo, de uma corporação;
- Tácito, que é o conhecimento que não é facilmente visível e "é muito difícil de expressá-lo por meio de palavras" (SVEIBY, 1998, p.35-36), pois é altamente pessoal e difícil de formalizar. Por este motivo, há dificuldade na sua transmissão e no seu compartilhamento com outras pessoas, principalmente, considerando que "os indivíduos mudam ou adaptam os conceitos de acordo com as próprias experiências e reinterpretam a linguagem utilizada para expressá-los" (SVEIBY, 1998, p.36). Os exemplos de conhecimento tácito são as conclusões, os *insights* e os palpites subjetivos, ou seja, qualquer tipo de conhecimento resultado das ações e experiências individuais, contendo as emoções, os valores e os ideais de que o detém.

Figura 2-17 – Quatro Modos de Conversão do Conhecimento (NONAKA e TAKEUCHI, 1997, p.69)

Nonaka e Takeuchi (1997, p.68-79) descrevem que o conhecimento é criado por meio da interação entre o conhecimento tácito e o conhecimento explícito, e que há quatro modos diferentes de conversão do conhecimento (Figura 2-17):

1. A socialização, de conhecimento tácito em conhecimento tácito;
2. A externalização, de conhecimento tácito em conhecimento explícito;
3. A combinação, de conhecimento explícito em conhecimento explícito; e
4. A internalização, de conhecimento explícito para conhecimento tácito.

"A criação do conhecimento organizacional é uma interação contínua e dinâmica entre o conhecimento tácito e o conhecimento explícito. Essa interação é moldada pela mudança entre diferentes modos de conversão do conhecimento que, por sua vez, são induzidos por vários fatores" (NONAKA e TAKEUCHI, 1997, p.79-80). O conteúdo do conhecimento criado por cada modo de conversão do conhecimento é naturalmente diferente (Figura 2-18): o conhecimento compartilhado, através da socialização; o conhecimento conceitual, através da externalização; o conhecimento operacional, através da internalização; e o conhecimento sistêmico, através da combinação.

Figura 2-18 – Conteúdo do Conhecimento Criado (NONAKA e TAKEUCHI, 1997, p.81)

A Sistemática da Gestão do Conhecimento compreende quatro (4) áreas de ênfase (WIIG, 1998, p. 8):

1. Monitorização de alto a baixo e facilitação de atividades relacionadas ao conhecimento;
2. Criação e manutenção da infra-estrutura do conhecimento;
3. Renovar, organizar e transformar ativos de conhecimentos; e

4. Utilizar os ativos de conhecimento para compreender seu valor.

A monitorização e a facilitação de atividades relacionadas ao conhecimento são de responsabilidade das áreas cujas funções sejam gerenciais e possui as seguintes atividades: inventariar e mapear o panorama do conhecimento; supervisionar a gestão do conhecimento; administrar os ativos intelectuais; implementar incentivos para motivar a criação, o compartilhamento e o uso do conhecimento; procurar uma estratégia centrada no conhecimento; reestruturar as operações e a organização (MELO, 1998; WIIG, 1998, p. 7-8).

Com o objetivo de estabelecer e atualizar a infra-estrutura do conhecimento, as funções do pessoal devem desenvolver as seguintes atividades: programas com as lições aprendidas para toda a companhia; bancos de conhecimentos com ontologias organizadas; fundos de recursos de conhecimentos profissionais; inventários de conhecimentos; capacidade múltipla para desenvolver e transferir conhecimento; universidade corporativa (WIIG, 1998, p. 7-8).

A área de funções operacionais tem como objetivo criar, renovar, construir e organizar os ativos de conhecimento, e suas atividades são: descobrir e inovar constantemente; adquirir conhecimento; educar e treinar; manter bancos de conhecimento; automatizar as transferências de conhecimento; fazer pesquisa e desenvolvimento; transformar e embutir conhecimento (WIIG, 1998, p. 7-8).

A última área, a de compreensão do valor do conhecimento, tem como objetivo distribuir e aplicar eficientemente os ativos do conhecimento e possui as seguintes atividades: usar sempre o melhor conhecimento; fazer a empresa toda compartilhar o conhecimento; colaborar para o *pool* de conhecimento apropriado; adotar as melhores práticas; e vender produtos com alto conteúdo de conhecimento (WIIG, 1998, p. 7-8).

2.4.1 Gestão do Conhecimento de Um Sistema Integrado de Controle de Semáforos em Tempo-Real

O conhecimento, segundo Davenport e Prusak (1999, p.1), não é dado e nem informação, embora esteja relacionado com ambos. Enquanto os dados correspondem a fatos distintos e objetivos relativos a eventos, a informação são dados dotados de relevância e propósito. Para que isto ocorra, os dados são transformados em informação agregando-se valor de diversas formas (DAVENPORT e PRUSAK, 1999, p.5):

- Contextualização: qual a finalidade dos dados coletados;
- Categorização: quais as unidades de análise ou componentes essenciais aos dados;
- Cálculo: como os dados podem ser analisados matematica e estatisticamente;
- Correção: como os erros dos dados podem ser eliminados; e
- Conclusão: como os dados podem ser resumidos para uma forma mais concisa.

Da mesma forma que a a informação deriva de dados, o conhecimento deriva da informação, e para tanto requer a sua transformação através de (DAVENPORT e PRUSAK, 1999, p.7);

- Comparação: de que forma as informações relativas a esta situação se comparam a outras situações conhecidas;
- Conseqüências: que implicações estas informações trazem para as decisões e tomadas de ações;
- Conexões: quais relações deste novo conhecimento com o conhecimento acumulado; e
- Conversão: o que as outras pessoas pessam desta informação.

Baseado nos conceitos apresentados, torna-se necessário identificar quais são os dados, as informações e o conhecimento que podem existir em

uma regulagem em tempo-real de um sistema de integrado de controle de semáforos.

Este tipo de regulagem, segundo Muller, Miska e Van Zuylen (2005, p.3), tem o objetivo de reduzir o congestionamento e maximizar a utilização da rede viária. Para tanto, o sistema deve ser capaz de compreender “a situação vigente do trânsito de rua, para em seguida calcular as mudanças que vão adequar as temporizações dos semáforos às variações percebidas no perfil do trânsito e, finalmente, implementar tais mudanças nos semáforos de uma forma organizada” (VILANOVA, 2004).

A situação vigente do trânsito, que é a informação resultante da transformação de dados detectados sobre tráfego da via (a velocidade – km/h – ; o fluxo – veículos/hora –; e a ocupação), é utilizada para (*i*) diagnosticar e justificar o diagnóstico dos problemas existentes em uma determinada área controlada e (*ii*) para propor ações sobre os controladores dos semáforos (HERNÁNDEZ, CUENA e MOLINA, 1997, p.2, p.5). Segundo, Kirshfink, Hernandez e Boero (2000, p.36), a velocidade, o fluxo e a ocupação – os dados – em conjunto com os dados da área de localização do semáforo são a base para geração de informações dos sistemas de controle de trânsito urbano.

Segundo Hernández, Cuena e Molina (1997, p.5), é possível identificar três (3) grupos distintos de conhecimento em um sistema de controle de trânsito urbano: estrutura física; problema do tráfego; e controle do tráfego.

O conhecimento proveniente da estrutura física corresponde as informações estáticas da topologia da rede viária (seus componentes e os relacionamentos entre eles) e os aspectos dinâmicos desta rede. Este conhecimento é resultado de abstrações de dados quantitativos (velocidade, fluxo e ocupação) recuperados através de sensores (HERNANDEZ, CUENA e MOLINA, 1997, p.5-6).

No problema do tráfego, o segundo grupo de conhecimento, refere-se a detecção e apresentação dos problemas do trânsito, como congestionamento e incidentes. Este conhecimento, recuperado a partir das informações provenitentes dos sensores, permite entender a malha viária com relação a sua capacidade de fluxo e o aumento da demanda que pode ou poderá impedir a fluidez do tráfego (HERNANDEZ, CUENA, MOLINA, 1997, p.6).

O controle de tráfego tem por objetivo identificar a situação do trânsito em uma determinada, através das informações originadas dos processamento dos dados capturados pelos sensores, e, através de mensagens de alertas aos motoristas, apresentar caminhos alternativos fora da área onde exista um congestionamento (HERNANDEZ, CUENA, MOLINA, 1997, p. 7-8).

Para Gardner (2000, p.1), os atuais sistemas inteligentes de transporte (*Intelligent Transport Systems – ITS*), que controlam semáforos, necessitam ter a informação, em tempo-real, sobre o tráfego para serem efetivos. No entanto, como o monitoramento contínuo do trânsito em um determinado ponto de uma determinada via é praticamente impossível, os dados sobre as condições do fluxo de veículos devem ser armazenadas para análise posterior.

Esta coleção de dados armazenados representa os registros dos movimentos dos veículos (velocidade média, quantidade de veículos por unidade de tempo) (GARDNER, 2000, p.2), e é a base, após análise das mesmas, para entendimento das características do trânsito: a densidade de veículos; a velocidade média dos veículos da via em um determinado momento; as rotas seguidas pelos veículos (MULLER, MISKA e VAN ZUYLEN, 2005, p.1).

A informação resultante da análise dos dados é utilizada para o desenvolvimento/modificação da infraestrutura de trânsito cujo o objetivo é reduzir o congestionamento e maximizar a utilização das vias (MULLER, MISKA, e VAN ZUYLEN, 2005, p.3). O conhecimento resultante da análise das informações sobre o trânsito auxiliará no gerenciamento do trânsito.

A geração do conhecimento é representado na Figura 2-19 (MULLER, MISKA e VAN ZUYLEN, 2005, p.11). Este processo inicia-se com o armazenamento dos dados puros sobre a situação do tráfego fornecidos pela infraestrutura utilizada no controle de semáforos e finaliza a utilização dos modelos resultantes (estimativas e prognósticos) no próprio Controle do Tráfego.

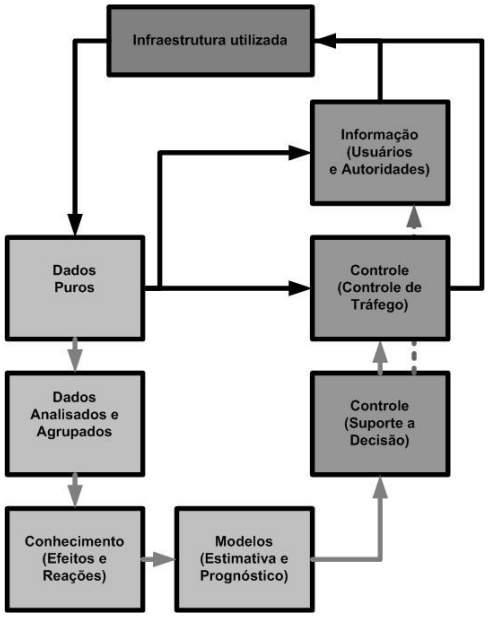


Figura 2-19 Gestão do Conhecimento x Controle de Semáforos (MULLER, MISKA e VAN ZUYLEN, 2005, p. 11)

Neste processo, os dados são transformados em informação, após agrupamentos e análises. A informação é, então, estudada no sentido de identificar quais os efeitos e as reações possíveis, ou seja, o conhecimento. A partir deste conhecimento é possível estabelecer modelos a serem aplicados no gerenciamento do trânsito (MULLER, MISKA e VAN ZUYLEN, 2005, p.3-4).

Os modelos gerados a partir do conhecimento são estimativas e prognósticos sobre o comportamento do tráfego, e serão utilizados nos sistemas de suporte a decisão e nos sistemas de controle de semáforos. O resultado da aplicação dos modelos são informações aos usuários do sistema

e do tráfego, e a própria infraestrutura do trânsito (MULLER, MISKA e VAN ZUYLEN, 2005, p.4).

3 METODOLOGIA

Segundo Moresi (2004, p.107-108), a metodologia é item de uma dissertação que contém a descrição completa e concisa da metodologia que foi empregada no desenvolvimento do trabalho.

O objetivo principal deste item é apresentar os fundamentos para a compreensão e interpretação dos resultados, bem como fornecer meios para a reprodução do presente estudo.

Para atingir o seu objetivo, este item foi dividido em duas (2) partes:

- Na primeira parte, será formulado o problema que a dissertação pretende resolver; e
- O detalhamento da alternativa escolhida será apresentado na segunda parte.

3.1 Formulação do Problema

O problema a ser solucionado pela dissertação é como deve ser a regulagem em tempo-real[14] de um sistema integrado de semáforos utilizando uma RNA recorrente com controle adaptativo.

Este sistema integrado de semáforos controlará apenas duas (2) intersecções (Figura 3-1) entre três (3) vias: uma (1) via principal (a horizontal); e duas (2) secundárias (as verticais).

[14] Sistemas em tempo-real são sistemas de computação que monitoram, respondem ou controlam um ambiente externo conectado através de sensores, atuadores e outras interfaces de entrada-saída. Este sistema deve satisfazer as várias restrições, temporais e outras, impostas a ele pelo comportamento de tempo-real do mundo externo com o qual faz interface (SHAW, 2003, p.15-17).

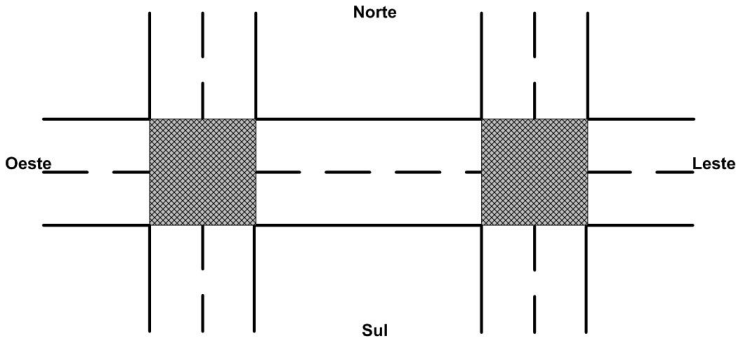


Figura 3-1 – Cruzamento entre Três (3) Vias

Cada uma das intersecções ou cruzamento (a parte cinza da Figura 3-1) será formado por duas (2) vias, uma horizontal e outra vertical, e cada uma das vias que a compõem conterá duas (2) faixas:

- Na horizontal, uma no sentido oeste-leste e outra no sentido, leste-oeste;
- Na vertical, uma no sentido norte-sul e a outra, sul-norte.

Um veículo, independente da faixa e da via, só poderá se movimentar em dois sentidos: em frente, em qualquer parte da pista; ou para a direita, quando estiver entrando no cruzamento.

Quatro (4) semáforos controlaram o cruzamento ou intersecção entre duas vias: o primeiro (S_1), controlará o fluxo da pista no sentido oeste-leste; o segundo (S_2), controlará o tráfego na pista do sentido norte-sul; o terceiro (S_3) controlará a movimentação dos veículos da pista no sentido leste-oeste; e o último (S_4), controlará o trânsito da pista no sentido sul-norte. Os semáforos estarão sincronizados dois a dois (S_1 e S_3; S_2 e S_4), ou seja, na mesma fase.

A regulagem dos quatros (4) semáforos da intersecção (S_1, S_2; S_3 e S_4.) deverá ser feita de acordo com a situação do fluxo de veículos em todas as faixas que compõem o cruzamento. E, também, deverá considerar, desde que exista tal informação, o estado atual do tráfego da via principal da outra intersecção.

Não sendo possível identificar o estado do tráfego de veículos das vias da intersecção, o sistema deverá ser capaz de garantir o funcionamento dos semáforos do cruzamento, utilizando outras estratégias de controle: plano de tempo fixo; plano de tempo fixo com verde variável.

Com o detalhamento do problema, é possível apresentar uma das alternativas, a escolhida para o desenvolvimento da dissertação, para a resolução do problema. O que será feito no próximo item.

3.2 Alternativa para a Resolução do Problema

Qualquer alternativa para resolver o problema proposto no item 3.1, deve garantir que o tempo das indicações luminosas não seja constante ao longo do tempo e que seja dependente da situação verificada do tráfego. No entanto, caso o fluxo de veículos não possa ser identificado, deve-se adotar uma estratégia que não venha prejudicar o trânsito nas vias.

Considerando a premissa apresentada no parágrafo anterior, a proposta para a resolução do problema será de um sistema com processamento distribuído[15] entre dois (2) computadores (Figura 3-2), cada um deles representando um Controlador de Intersecção.

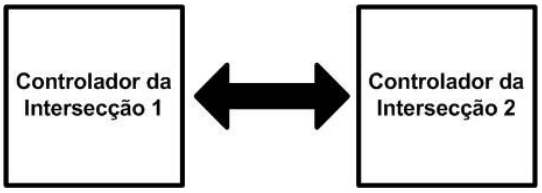


Figura 3-2 – Sistema de Regulagem de Semáforos Distribuído

[15] Sistema Distribuído é uma coleção de computadores independentes no qual o usuário vê como se fosse um simples sistema consistente (TANEMBAUM e VAN STEEN, 2001; COULOURIS, DOLLIMORE e KINDBERG, 2001).

Cada Controlador de Intersecção será responsável pela regulagem dos semáforos que controlam o seu cruzamento. Esta regulagem será baseada no fluxo de veículos das suas vias e, também, da fase atual do semáforo da via principal da outra intersecção. A fase atual do semáforo da via principal de uma intersecção será informada pelo outro controlador através da troca de mensagens (seta da Figura 3-2).

A troca de mensagens entre ambos os controladores tem o objetivo de permitir que um controlador conheça qual o estado do semáforo da via principal da outra intersecção, garantindo com isto, o sincronismo de bloqueio ou liberação do fluxo de veículos das vias.

No caso da comunicação entre os controladores ser interrompida ou perdida, cada um deles deve agir como uma unidade autônoma, ou seja, deve ter a capacidade de continuar o processo de regulagem dos semáforos sob sua responsabilidade com as informações do fluxo de veículos das vias sob seu controle, desconsiderando a fase do semáforo da outra intersecção.

Como proposta, foi desenvolvido o modelo geral da solução (Figura 3-3) composto de:

- Quatro (4) Controladores de Fluxos de Veículos – CFV11, CFV12, CFV21 e CFV22;
- Dois (2) Controladores de Semáforos da Intersecção – CSI1 e CSI2; e
- Duas (2) Redes Neurais Locais – RNL1 e RNL2.

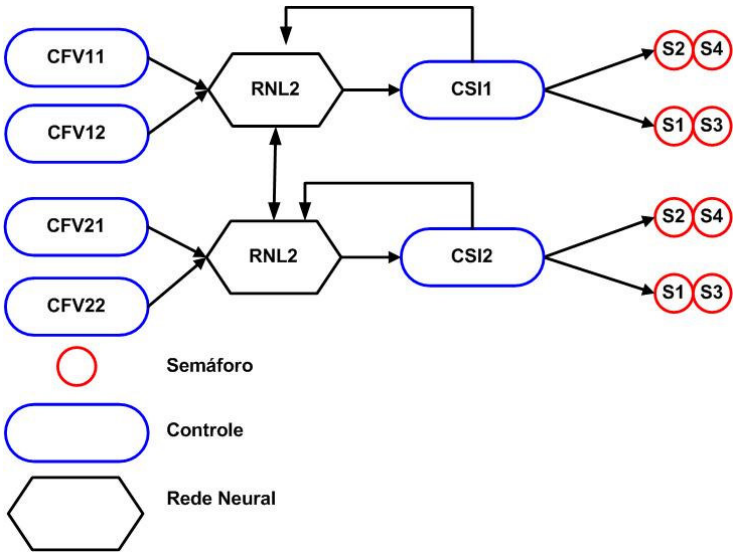


Figura 3-3 – Modelo Geral para Resolução do Problema

Cada um dos componentes do modelo geral terá o seu funcionamento explicitado nos próximos itens.

3.2.1 Controlador de Fluxo de Veículos

O Controlador de Fluxo de Veículos – CFV, que é um controlador dinâmico, tem o objetivo de determinar qual o fluxo de veículos de uma das faixas da via. Esta informação será determinada a partir dos detectores veiculares[16] .

Os detectores veiculares, em número de dois (2) por sentido da via (Figura 3-4), registrarão a presença e a velocidade dos veículos que entrarem na via no sentido da intersecção (o detector de entrada – d_e) e que saírem da via entrando no cruzamento (o detector de saída – d_s). Estas duas informações

[16] Detectores de veículos são equipamentos utilizados para registrar a presença ou passagem de veículos em um determinado ponto da via (HOMBURGER, WOLFGANG, LOUTZENHEISER e REILLY, 1989, p.5-2, p.17-9).

permitiram determinar o volume do tráfego e a velocidade média dos veículos na via.

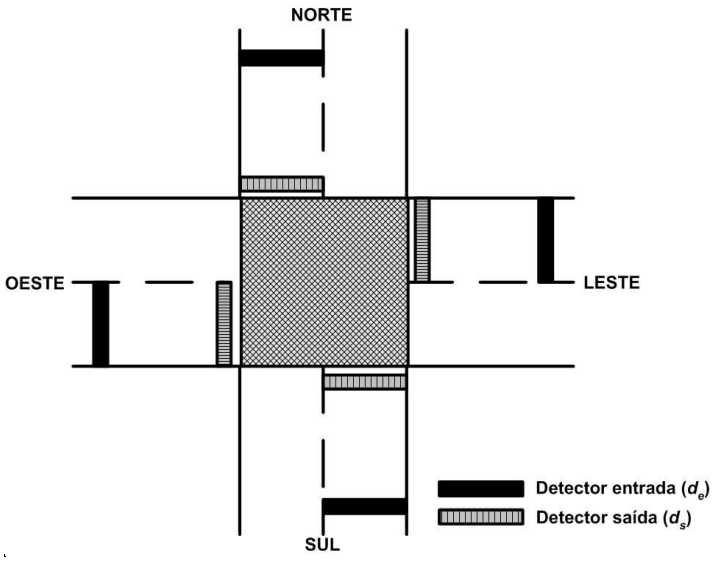


Figura 3-4 – Detectores Veiculares

A quantidade de veículos (*q*) ou o volume do tráfego na pista de uma via, utilizando o registro de cada um dos detectores veiculares, será determinado por:

$$q = q_{ant} + d_e - d_s \tag{3.1}$$

onde:

- q é a quantidade de veículos na pista da via no momento corrente (t).
- q_{ant} é a quantidade de veículos na pista da via no momento anterior ($t - 1$).
- d_e é o registro ou não de um (1) veículo entrando na pista da via no momento corrente (t).
- d_s é o registro ou não de um (1) veículo saindo da faixa da via no momento anterior ($t - 1$).

O volume do tráfego na pista da via será expresso qualitativamente (Tabela 3-1) através da comparação do resultado calculado (*q*) com a

quantidade máxima de veículos pré-estabelecida para a faixa da via (q_{max}), e poderá ser:

- Se a quantidade de veículos determinada (q) for igual a zero (não há veículos na pista), então o volume será NENHUM (Situação I);
- Se a quantidade de veículos (q) for maior que um (1) e menor ou igual à metade da quantidade máxima, então o volume será POUCO (Situação II);
- Se a quantidade de veículos (q) for maior que a metade da quantidade máxima e menor ou igual a dois terços $\left(\frac{2}{3}\right)$ da quantidade máxima, então o volume será NORMAL (Situação III); e
- Se a quantidade de veículos (q) for maior que dois terços $\left(\frac{2}{3}\right)$ da quantidade máxima, então o volume será MUITO (Situação IV).

Tabela 3-1 – Volume do Tráfego na Pista

Situação	Condição	Volume
I	$q = 0$	NENHUM
II	$0 \langle q \leq \frac{q_{\max}}{2}$	POUCO
III	$\frac{q_{\max}}{2} \langle q \leq \frac{2q_{\max}}{3}$	NORMAL
IV	$\frac{2q_{\max}}{3} \langle q$	MUITO

Como cada via contém duas (2) pistas e como o volume do tráfego de cada uma delas pode ser diferente em um determinado momento (t), será necessário considerar o maior volume entre as duas pistas como o volume da via (Tabela 3-2):

- Se ambos os volumes das pistas forem iguais, então o volume da via será o mesmo (Situação I, II, III e IV); e

- Se o volume de uma pista for diferente do da outra pista, então o volume da via será o maior volume entre os dois (Situação V, VI e VII).

Tabela 3-2 – Volume do Tráfego da Via

Situação	Volume		Volume da Via
	Uma Pista	Outra Pista	
I	NENHUM	NENHUM	NENHUM
II	POUCO	POUCO	POUCO
III	NORMAL	NORMAL	NORMAL
IV	MUITO	MUITO	MUITO
V	NENHUM	POUCO	POUCO
VI	NENHUM ou POUCO	NORMAL	NORMAL
VII	NENHUM, POUCO ou NORMAL	MUITO	MUITO

O volume do tráfego da via (Tabela 3-2) é a qualificação da quantidade de veículos determinada pela aplicação da equação (3.1) em um determinado momento (*t*), mas é insuficiente para, sozinho, determinar o fluxo de veículos da via de forma conclusiva, ou seja, não consegue responder as seguintes perguntas:

- O fluxo está parado?
- O fluxo está parando?
- O fluxo está começando a se movimentar?
- O fluxo está em movimento?

Para responder as perguntas acima é necessário, além de conhecer o volume do trânsito, ter conhecimento da velocidade média (v_m) da pista em um determinado momento (*t*). A velocidade média pode ser expressa por:

$$v_m = \frac{v_s + v_e}{2} \tag{3.2}$$

onde:

- v_m é a velocidade média da pista no momento *t*.
- v_s é a velocidade identificada pelo detector de entrada (d_e) da pista no momento *t*.
- v_e é a velocidade identificada pelo detector de saída (d_s) da pista no momento *t*.

OBSERVAÇÃO: Os detectores de entrada (d_e) e de saída (d_s) devem ser capazes de fornecer a velocidade do veículo que foi identificado.

A velocidade média (v_m) da pista será expressa qualitativamente (Tabela 3-3) através da comparação entre o resultado calculado e a velocidade máxima da via[17] (v_{max}) e poderá ser:

- Se a velocidade média calculada for igual a zero (não há movimento na pista), então será PARADO (Situação I).
- Se a velocidade média calculada for maior que zero (0) e menor e igual à velocidade mínima[18] estabelecida para a via, então será BAIXA (Situação II).
- Se a velocidade média calculada for maior que a velocidade mínima e menor ou igual à velocidade máxima, então será REGULAR (Situação III).
- Se a velocidade média calculada for maior que a velocidade máxima, então será ALTA (Situação IV).

[17] A velocidade máxima permitida para uma via será indicada por meio de sinalização obedecida as suas características técnicas e as condições de trânsito (BRASIL, 1997, Art. 61).

[18] A velocidade mínima permitida em uma via não poderá ser inferior a metade da velocidade máxima estabelecida, respeitadas as condições operacionais de trânsito e da via (BRASIL, 1997, Art. 62).

Tabela 3-3 – Velocidade Média

Situação	Condição	Velocidade
I	$V_m = 0$	Parada
II	$0 < V_m \leq V_{\min}$	Baixa
III	$V_{\min} < V_m \leq V_{\max}$	Regular
IV	$V_m > V_{\max}$	Alta

Como cada via contém duas (2) faixas e a velocidade média cada uma delas pode ser diferente em um determinado momento (*t*), será necessário considerar a pior velocidade entre as duas pistas como a velocidade da via (Tabela 3-4):

- Se ambas as velocidades forem iguais, então a velocidade será a mesma (Situação I, II, III e IV).
- Se as velocidades forem diferentes, então a velocidade será a mais baixa entre elas (Situação V, VI e VII).

Tabela 3-4 – Velocidade do Tráfego da Via

Situação	Velocidade		Velocidade da Via
	Uma Pista	Outra Pista	
I	PARADA	PARADA	PARADA
II	BAIXA	BAIXA	BAIXA
III	REGULAR	REGULAR	REGULAR
IV	ALTA	ALTA	ALTA
V	PARADA	BAIXA, REGULAR ou ALTA	PARADA
VI	BAIXA	REGULAR ou ALTA	BAIXA
VII	REGULAR	ALTA	REGULAR

As informações sobre o volume de tráfego (Tabela 3-2) e a velocidade do trânsito (Tabela 3-4) observadas separadamente não permitem determinar

qual a situação do trânsito em uma via. No entanto, quando analisadas em conjunto, permitem estabelecer um qualificador sobre o tráfego de veículos da via: nenhum; baixo; médio; e alto. Esta informação será uma das uma das entradas da RNL.

Para identificar o qualificador do fluxo de veículos será necessário estabelecer as seguintes regras para o funcionamento do controlador:

Regra	Descrição da Regra
R01	Se o Volume for NENHUM, então o Tráfego será NENHUM.
R02	Se a Volume for MUITO e a Velocidade for ALTA, então o Tráfego será BAIXO.
R03	Se a Volume for MUITO e a Velocidade for NORMAL, então o Tráfego será MÉDIO.
R04	Se a Volume for MUITO e a Velocidade for BAIXA, então o Tráfego será ALTO.
R05	Se o Volume for NORMAL e a Velocidade for ALTA, então o Tráfego será BAIXO.
R06	Se o Volume for NORMAL e a Velocidade for NORMAL, então o Tráfego será MÉDIO.
R07	Se o Volume for NORMAL e a Velocidade for BAIXA, então o Tráfego será ALTO.
R08	Se o Volume for POUCO e a Velocidade for ALTA, então o Tráfego será BAIXO.
R09	Se o Volume for POUCO e a Velocidade for NORMAL, então o Tráfego será MÉDIO.
R10	Se o Volume for POUCO e a Velocidade for BAIXA, então o Tráfego será ALTO.

Os dois parâmetros – a velocidade média calculada e a quantidade de veículos – devem ser armazenados para serem utilizadas futuramente.

3.2.2 Controlador Semafórico da Intersecção

O Controlador Semafórico da Intersecção - CSI, que é um controlador dinâmico, tem o objetivo de estabelecer a indicação luminosa dos semáforos do cruzamento sob a sua responsabilidade.

Este controlador trabalhará de duas formas, dependo da existência de comunicação entre a RNL e ele:

- Na inexistência de comunicação, o CSI irá alterar o intervalo do ciclo de acordo com o tempo pré-estabelecido.

OBSERVAÇÃO: O tempo pré-estabelecido deverá estar armazenado em memória e nela será armazenado após a sua recuperação em um dos dispositivos de armazenamento, quando o controlador iniciar a sua operação. Além deste tempo, também será recuperado o horário de inatividade do semáforo.

- Quando existir comunicação, o CSI irá acatar a ordem da RNL e reduzirá o intervalo do ciclo atual a fim de que a indicação luminosa que será apresentada espelhe a ordem recebida. Esta ordem poderá ser:

    - Liberar o fluxo da via principal, ou seja, a indicação luminosa no semáforo desta via principal será verde;
    - Bloquear o fluxo da via principal, ou seja, a indicação luminosa no semáforo desta via principal será vermelha; ou
    - Bloquear o fluxo de ambas as vias, ou seja, a indicação luminosa das duas vias será vermelha.

OBSERVAÇÃO: A aceleração do tempo de cada um dos ciclos ao invés da mudança imediata da indicação luminosa que represente a ordem recebida está na necessidade de não quebrar

a seqüência conhecida das alterações de cores do semáforo (Tabela 3-5).

Tabela 3-5 – Indicações Luminosas dos Semáforos

Situação	Indicação Atual	Indicação Futura
I	Verde	Amarelo
II	Amarelo	Vermelho
III	Vermelho	Verde

Para que o ciclo atual seja reduzido será necessário estabelecer um tempo mínimo para cada uma das indicações luminosas: o tempo mínimo de verde – vd_{min}; o tempo mínimo de amarelo – am_{min}; e o tempo mínimo de vermelho – vm_{min}. Estes tempos serão os utilizados como o tempo máximo de cada indicação luminosa em substituição aos tempos pré-estabelecidos.

Independente da forma de funcionamento, o CSI deverá ser capaz, a partir da sua situação atual, de estabelecer sua próxima indicação considerando, também, qual a última indicação luminosa antes da atual. Todas as situações possíveis estão expressa na Tabela 3-6.

Tabela 3-6 – Fases dos Semáforos da Intersecção

Situação	Par 1 de Sinais			Par 2 de Sinais		
	Anterior	Atual	Próxima	Anterior	Atual	Próxima
I	------	Intermitente	Vermelho	------	Intermitente	Vermelho
II	Intermitente	Vermelho	Verde	Intermitente	Vermelho	Vermelho
III	Vermelho	Verde	Amarelo	Vermelho	Vermelho	Vermelho
IV	Verde	Amarelo	Vermelho	Vermelho	Vermelho	Vermelho
V	Amarelo	Vermelho	Vermelho	Vermelho	Vermelho	Verde
VI	Vermelho	Vermelho	Vermelho	Vermelho	Verde	Amarelo
VII	Vermelho	Vermelho	Vermelho	Verde	Amarelo	Vermelho
VIII	Vermelho	Vermelho	Verde	Amarelo	Vermelho	Vermelho

OBSERVAÇÃO: INTERMITENTE corresponde à apresentação intermitentemente da indicação luminosa amarela, geralmente, no horário de inatividade do semáforo.

Baseado na Tabela 3-6 será necessário estabelecer as seguintes regras de funcionamento do controlador, considerando Sinal 1, como a indicação luminosa da via principal; e Sinal 2, a indicação luminosa da via secundária:

Regra	Descrição da Regra
R01	Se estiver no horário de inatividade do semáforo, os Sinal 1 e 2 serão INTERMITENTE.
R02	Se os Sinais 1 e 2 estiverem INTERMINTENTE e estiver fora do horário de inatividade do semáforo, então os Sinais 1 e 2 serão VERMELHO.
R03	Se o Sinais 1 e 2 estiverem VERMELHO e a indicação anterior do Sinal 1 foi INTERMITENTE, então o Sinal 1 será VERDE e o Sinal 2 será VERMELHO.
R04	Se o Sinal 1 estiver VERDE e o Sinal 2 estiver VERMELHO, então o Sinal 1 será AMARELO e o Sinal 2 será VERMELHO.
R05	Se o Sinal 1 estiver AMARELO e o Sinal 2 estiver VERMELHO, então o Sinal 1 será VERMELHO e o Sinal 2 será VERMELHO.
R06	Se o Sinal 1 estiver VERMELHO e o Sinal 2 estiver VERMELHO e a indicação anterior do Sinal 1 for AMARELO, então o Sinal 1 será VERMELHO e o Sinal 2 será VERDE.
R07	Se o Sinal 2 estiver VERDE e o Sinal 1 estiver VERMELHO, então o Sinal 2 será AMARELO e o Sinal 2 será VERMELHO.
R08	Se o Sinal 2 estiver AMARELO e o Sinal 1 estiver VERMELHO, então o Sinal 2 será VERMELHO e o Sinal 2 será VERMELHO.
R09	Se o Sinal 2 estiver VERMELHO e o Sinal 1 estiver VERMELHO e a indicação anterior do Sinal 2 for AMARELO, então o Sinal 2 será VERMELHO e o Sinal 1 será VERDE.

As regras estabelecidas serão aplicadas toda a vez em que o tempo para a indicação luminosa atual for atingido. Nos casos de ordem da RNL, os tempos aplicados serão os pré-estabelecidos (vd_{min}, am_{min},ou vm_{mi}).

O resultado deste controlador será uma das entradas da RNL e os tempos de ciclos utilizados devem ser armazenados para uso futuro.

3.2.3 Rede Neural Local

A Rede Neural Local – RNL tem o objetivo de estabelecer qual o estado que o semáforo da via principal deve ter após o processamento do CSI. Esta RNL é uma *Multilayer Percepton* – MLP, pois terá três (3) camadas: uma de entrada; uma intermediária; e uma de saída (Figura 3-5).

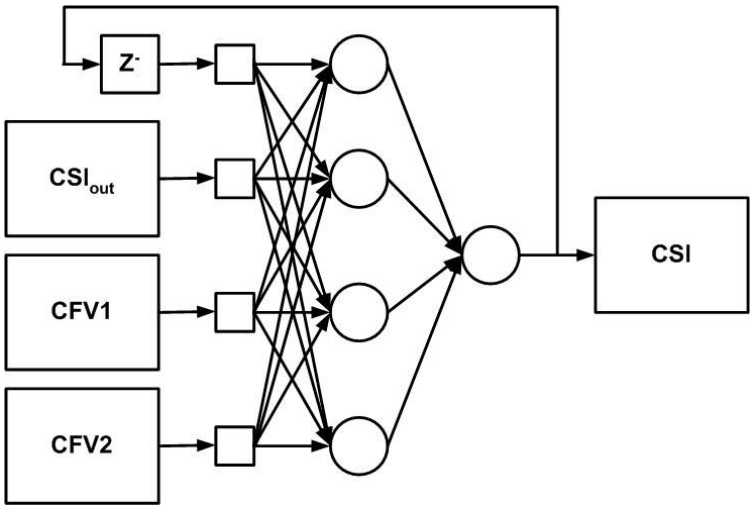


Figura 3-5 – Rede Neural Local

A camada de entrada terá quatro (4) nós:

- O primeiro, é a informação repassada pelo controlador de fluxo de veículos da via principal, o CFV1;
- O segundo, é a informação repassada pelo controlador de fluxo de veículos da via secundária, o CFV2;

- O terceiro, é a informação repassada pelo controlador semafórico da outra intersecção, o CSI_{out}; e
- O quarto e último, é a informação referente a ordem repassada para o controlador semafórico da sua intersecção, o CSI. Esta informação representa a recorrência da rede.

O algoritmo utilizado será o *Backpropagation Through Time* – BPTT, que é uma extensão do *Backpropagation* de uma RNA não recorrente com a utilização de janelas de tempo de $\frac{1}{100}$ segundos.

Haverá apenas uma camada intermediária, e a mesma terá somente quatro (4) neurônios. Esta quantidade de neurônios foi estabelecida após treino e teste da rede.

Correspondendo à ordem que o CSI da Intersecção receberá da rede, a camada de saída terá um (1) único neurônio. De acordo com as informações de entrada, a RNL deverá determinar as seguintes:

- Liberado – Neste caso, após o processamento pelo CSI, o semáforo da via principal deverá estar com a indicação luminosa VERDE, enquanto que o da via secundária deverá estar VERMELHO;
- Bloqueado – Neste caso, após o processamento pelo CSI, o semáforo da via principal deverá estar com a indicação luminosa VERMELHA e o da via secundária, VERDE.
- Ambos – Neste caso, após o processamento pelo CSI, ambos os semáforos, o da via principal e o da via secundária, devem estar com a indicação luminosa VERMELHA.

A RNL só será capaz de determinar qual o estado do semáforo da via principal se as entradas esperadas (do CSI_{out}, do CFV1, do CFV2 e a sua recorrência) estiverem disponíveis. Caso contrário, ela não repassará nenhuma ordem para o CSI da intersecção.

4 RESULTADOS E DISCUSSÕES

Como detalhado no item 3.1, o problema a ser solucionado por esta dissertação é como uma RNR e um Controlador Adaptativo podem garantir uma boa regulagem em tempo-real de um sistema integrado de semáforos que controle dois (2) cruzamentos (Figura 4-1) entre uma (1) via principal, a horizontal, e duas (2) secundárias, as verticais.

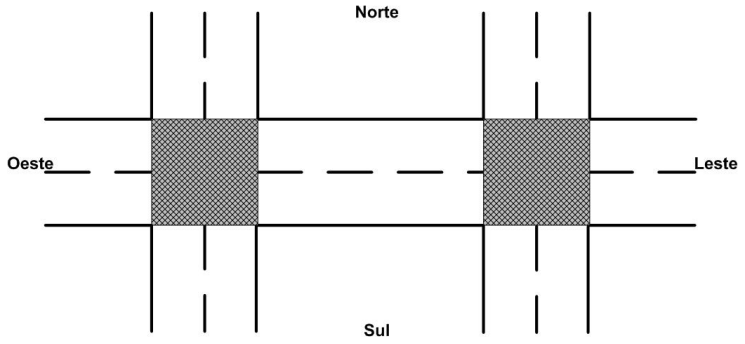


Figura 4-1 – Cruzamento entre Três (3) Vias

4.1 A Arquitetura para a Solução do Problema

Como o processamento deve ser em tempo-real e distribuído entre computadores, optou-se, inicialmente, por uma arquitetura formada por (Figura 4-2):

- Um Controlador Central – CC; e
- Dois (2) Controladores da Intersecção – CI.

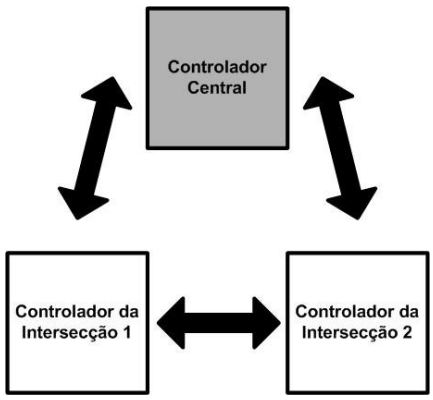


Figura 4-2 – Arquitetura do Sistema Distribuído

Nesta arquitetura, o CC, que é um controlador robusto e é único, será responsável em garantir o sincronismo entre os semáforos da via principal das duas (2) intersecções. Ele receberá, através de mensagem (seta na Figura 4-2), qual o estado atual do semáforo de uma interseção, e após armazená-lo, irá repassar esta informação para o outro CI e vice-versa. Com esta comunicação, um CI será capaz de decidir ou não pela mudança do estado atual do semáforo da via principal do seu cruzamento.

Além do CC, nesta arquitetura proposta haverá dois (2) CI. Cada um destes controladores é o sistema de processamento que efetuará a regulagem dos semáforos de uma (1) interseção de forma autônoma e cooperativa:

- Cooperativa: Pois cada mudança de indicação luminosa do semáforo da via principal sob sua responsabilidade será informada (seta na Figura 4-2), ao mesmo tempo, para CC e para o outro CI. Esta comunicação entre os controladores permitirá o sincronismo entre os semáforos da via principal de ambos os cruzamentos.
- Autônoma: Pois caso a comunicação entre os controladores seja interrompida ou perdida, cada CI será capaz de continuar fazendo uma boa regulagem dos semáforos sob sua responsabilidade.

Da proposta inicial, a informação do estado atual de um semáforo de uma intersecção é repassada, sempre, do seu CI tanto para outro CI como

para o CC, que por sua vez, repassa para o outro CI. Esta redundância visa garantir que a comunicação entre os CI seja mantida. No entanto, se a comunicação entre qualquer CI e CC for perdida ou interrompida, a função principal CC ficará comprometida, o que não ocorre com os CI em caso de falta de comunicação entre eles e/ou o CC. Desta forma, a inexistência do CC não comprometeria o objetivo do sistema.

Considerando que a inexistência do CC na arquitetura inicialmente proposta não comprometerá a função do sistema, a nova arquitetura proposta será composta (Figura 4-3), exclusivamente, de dois (2) Controladores de Intersecção – CI. E cada um dos controladores terá a mesma função atribuída na arquitetura apresentada na Figura 4-2, inclusive com a comunicação com o seu outro par.

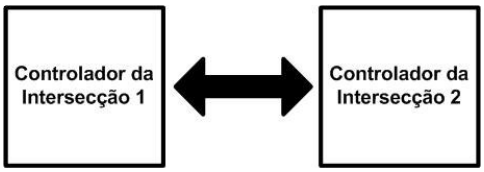


Figura 4-3 – Nova Arquitetura do Sistema Distribuído

Cada um dos CI será, por sua vez, um sistema, como dito anteriormente, autônomo com funções que permitam a regulagem dos semáforos da intersecção sob a sua responsabilidade. Para tanto, este sistema deverá ter capacidade para:

- Identificar as condições do tráfego nas vias que compõem o seu cruzamento;
- Identificar a atual e próxima indicação luminosa de cada um dos semáforos que pertencem ao seu sistema;
- Conhecer a indicação luminosa atual do semáforo da via principal da outra intersecção do sistema geral; e

- Alterar a atual indicação luminosa do semáforo da sua via principal baseado nas condições do trânsito e, se for possível, da indicação luminosa da via principal da outra intersecção.

Considerando o princípio de coesão funcional[19], as capacidades listadas no parágrafo anterior foram agrupadas em três (3) funções distintas:

a) De identificar o fluxo de veículos de cada uma das vias do cruzamento.
b) De conhecer e modificar a indicação luminosa dos semáforos da intersecção, em virtude do tempo de ciclo ter sido atingido ou em conseqüência da alteração do tráfego de veículos das vias.
c) De determinar a partir do fluxo de veículos das vias do cruzamento e dos estados atuais de cada um dos semáforos da via principal, da sua intersecção e da outra intersecção, qual a nova indicação luminosa do semáforo da sua via principal.

Para a resolução do problema da dissertação foi proposto um modelo geral (Figura 4-4) para a nova arquitetura (Figura 4-3), considerando a existência das três (3) funções relacionadas no parágrafo anterior. Este modelo será composto por:

- Controlador de Fluxo de Veículos – CFV, que será responsável pela função descrita na letra “a”. Existirão quatro (4) controladores (CFV11, CFV12, CFV21 e CFV22), sendo dois (2) por via que compõe o cruzamento, ou seja, um (1) controlador para cada faixa da via;

---

[19] Um módulo com coesão funcional, para a metodologia estruturada, contém elementos diferentes, que são necessários e suficientes, para a execução de uma e somente uma função relacionada ao problema (FELICIANO NETO, FURLAN e HIGA, 1988, p.216; PAGE-JONES, 1988, p.130) ou, para a metodologia orientada a objeto, é uma medida de integridade conceitual de uma classe (FURLAN, 1998, p.20).

- Controlador de Semáforos de Intersecção – CSI, que será responsável pela função descrita na letra “b”. Para cada uma das intersecções existirá um (1) único controlador para todos os semáforos de um cruzamento, ou seja, existirão dois (2) controladores (CSI1 e CSI2); e
- A função descrita na letra “c” será de responsabilidade da RRNL: Existirão duas RNL, a RNL1 e a RNL2, e cada uma trabalhará para apenas uma única intersecção.

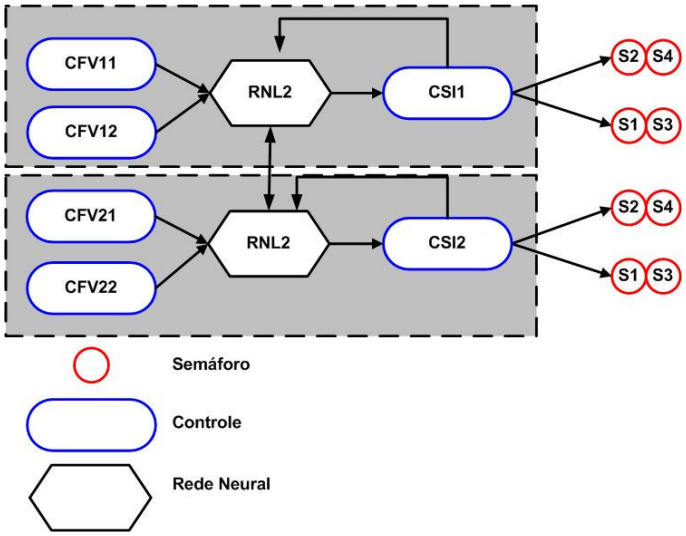


Figura 4-4 – Modelo Geral

O modelo geral da Figura 4-4 foi proposto para resolver o problema de duas (2) intersecções. No entanto, ambos os cruzamentos têm as mesmas características, ou seja, possuem uma (1) via principal e outra, secundária. Desta forma, a solução desenvolvida para uma intersecção poderá ser aplicada completamente para o outro cruzamento, e é por isso que no modelo proposto foram apresentados dois (2) conjuntos semelhantes (a parte cinza da Figura 4-4). Cada um destes conjuntos corresponde a um Controlador de Intersecção – CI (Figura 4-3).

A solução para cada um dos cruzamentos da regulagem em tempo-real de semáforos em um cruzamento ortogonal entre duas vias (Figura 4-5) poderá ser desenvolvido de duas formas:

- Na primeira, como um controle isolado de cruzamento, por se basear no fluxo de veículos de uma única intersecção de vias e não existir a preocupação com as eventuais influências das intersecções adjacentes; e
- Na segunda, como um controle arterial de cruzamento (Rede Aberta), quando o controle dos movimentos de uma via principal (corredor) visa garantir a continuidade do seu fluxo de veículos sobre as intersecções adjacentes (sistema progressivo ou onda verde).

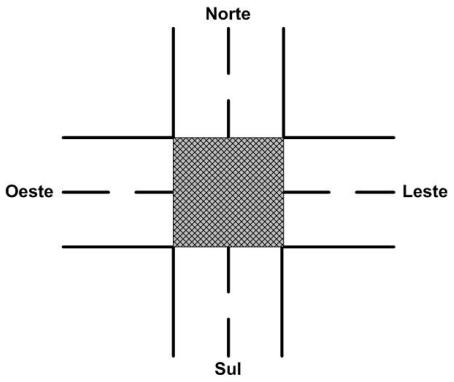


Figura 4-5 – Intersecção entre duas (2) vias

Independente da forma a ser adotada, cada uma das duas vias do cruzamento em estudo (a horizontal e a vertical), contém duas (2) faixas (uma no sentido oeste-leste e outra, no leste-oeste para a via horizontal; e uma no sentido norte-sul e a outra, sul-norte para a via vertical) e uma intersecção. O movimento possível de um veículo em qualquer uma das pistas será ou em frente ou à direita.

A intersecção ou cruzamento das duas vias (a parte quadriculada da Figura 4-1 e Figura 4-5) será controlada por quatro (4) semáforos: o primeiro (S_1), controlará o fluxo da pista no sentido oeste-leste; o segundo (S_2) controlará o tráfego na pista do sentido norte-sul; o terceiro (S_3) controlará a

movimentação dos veículos da pista no sentido leste-oeste; e o último (S_4) controlará o trânsito da pista no sentido sul-norte. Os semáforos estarão sincronizados dois a dois: S_1 e S_3; S_2 e S_4.

4.2 Controlador de Fluxo de Veículos

Das funções descritas no item anterior, a primeira a ser desenvolvida será a do Controlador de Fluxos de Veículos – CFV. Este controlador, como dito anteriormente, terá a responsabilidade de identificar o fluxo de veículos de cada uma das vias do cruzamento. Esta informação é importante para que a RNL possa decidir pela alteração ou não do tempo de ciclo dos semáforos.

Existirão, para cada intersecção, dois (2) controladores deste tipo. Cada um será responsável pela identificação do trânsito das duas (2) pistas que compõem a via. Ambos trabalharão de forma independente, sem interferir e/ou sofrer interferência do outro.

O CFV para poder desenvolver a sua função necessita conhecer, em tempo-real, como está o tráfego de veículos na via sob a sua responsabilidade. Desta forma, a primeira pergunta a ser respondida é como conseguir identificar o trânsito de uma via. Esta pergunta pode ser respondida considerando o trabalho desenvolvido por Lee e Lee-Kwange (1999, p. 264-265).

Lee e Lee-Kwange (1999, p.264-265) propuseram a utilização de detectores veiculares[20](Figura 4-6) para determinar, em tempo-real, qual o fluxo de veículos de uma das pistas da via. Cada uma (1) das pistas da via terá dois (2) detectores, com as seguintes funções:

- O primeiro, o detector de entrada d_e, localizado a uma certa distância da intersecção ou cruzamento das vias, que tem a função de registrar

[20] Detectores de veículos são equipamentos utilizados para registrar a presença ou passagem de veículos em um determinado ponto da via (HOMBURGER, WOLFGANG, LOUTZENHEISER e REILLY, 1989, p.5-2, p.17-9).

a presença dos veículos que estão entrando na via no sentido da intersecção:

OBSERVAÇÃO: Uma das formas de calcular a distância ideal deste detector é multiplicando o comprimento da via vezes 0,375 (LEE e LEE-KWANGE, 1999, p.264-265). A constante 0,375 corresponde ao resultado da divisão de cento e cinqüenta (150) por quatrocentos (400). Os dois valores correspondem, respectivamente, à distância ideal do detector inicial e o comprimento total da via.

- O segundo, o detector de saída d_s, localizado o mais perto possível da intersecção, mas não nela, que tem a função de registrar a presença dos veículos que estão saindo da via em direção à intersecção.

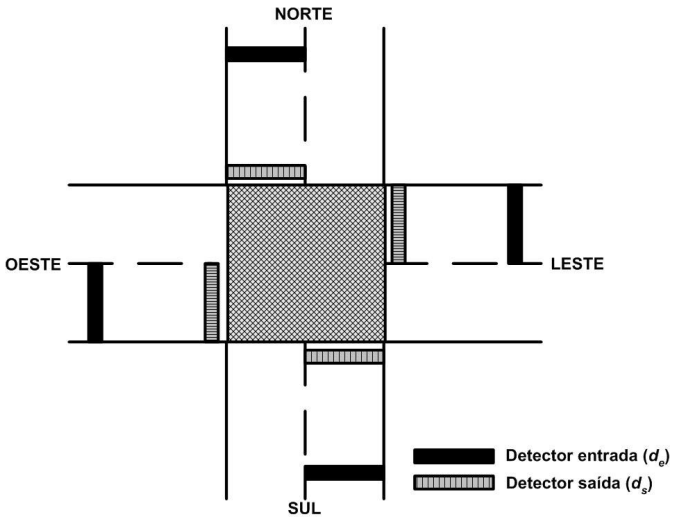


Figura 4-6 – Detectores Veiculares

Para determinar o fluxo de veículos em cada uma das faixas, o detector de entrada (d_e) registrará os veículos que entraram, enquanto que o detector de saída (d_s) registrará os veículos que saíram da via. As informações registradas devem permitir conhecer qual a situação do tráfego (normal, iniciando, parando ou parado) e, também, a quantidade de veículos (q_v) que estão na pista. Esta última informação poderá ser obtida por:

$$q_v = d_e - d_s \tag{4.1}$$

onde:

- q_v é a quantidade de veículos da pista num determinado instante *t*.
- d_e é o valor recuperado pelo detector de entrada.
- d_s é o valor recuperado pelo detector de saída.

Considerando que cada um dos detectores, o detector de entrada (d_e) e o detector de saída (d_s), só poderá assumir, em um determinado instante, um dos dois (2) estados: ativo e inativo. Sendo que estado inicial de cada um deles será inativo, e mudará para o ativo quando o mesmo detectar a passagem de veículos, voltando, imediatamente após, ao seu estado original, a questão a ser respondida é a formula (4.1) permitirá identificar a quantidade de veículos em uma faixa.

A resposta desta pergunta é não, após uma simulação com quarenta (40) entradas aleatórias (item 0) considerando que a representação dos dois (2) estados, ativo e inativo, pudesse ser feita pelos números um (1) e zero (0), respectivamente. O resultado desta simulação demonstrou que a equação (4.1) utilizando as informações dos sensores não permitiu o cálculo da quantidade de veículos da pista, pois apresentou somente quatro (4) valores como resultado final. Dos quatro (4) valores foi possível extrair quatro (4) situações possíveis da pista (Tabela 4-1) e nenhuma delas conclusiva.

Tabela 4-1 – Situações Possíveis de Acordo com os Detectores Veiculares

Situação	d_e	d_s	$q_v = d_e - d_s$
I	0	0	0
II	0	1	-1
III	1	0	1
IV	1	1	0

A inexistência de movimento na pista é representado pela situação I da Tabela 4-1, pois não foi identificada a presença de veículos pelo detector entrada (d_e = 0) e nem pelo de saída (d_s = 0). No entanto, não foi possível

concluir a inexistência de fluxo de veículos da pista (Figura 4-7.a) ou se o trânsito está parado (Figura 4-7.b). Além disto, não foi possível saber qual a quantidade de veículos estavam na pista no momento.

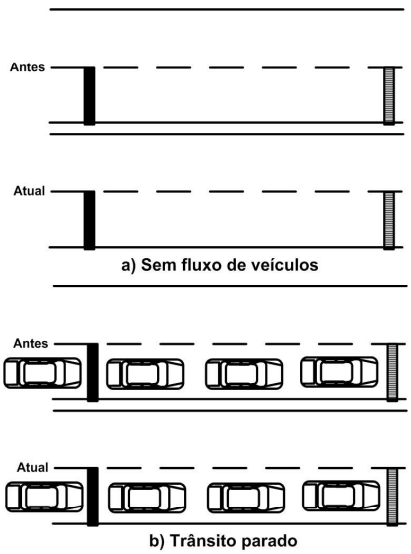


Figura 4-7 – Situação I

Na situação IV da Tabela 4-1 que representa a entrada e, também, a saída de veículos da pista, pois cada um dos detectores registrou a presença de veículos diferentes. Isto pode caracterizar a existência de tráfego (Figura 4-8). Mas, novamente, não é possível saber qual a quantidade de veículos que estão na pista.

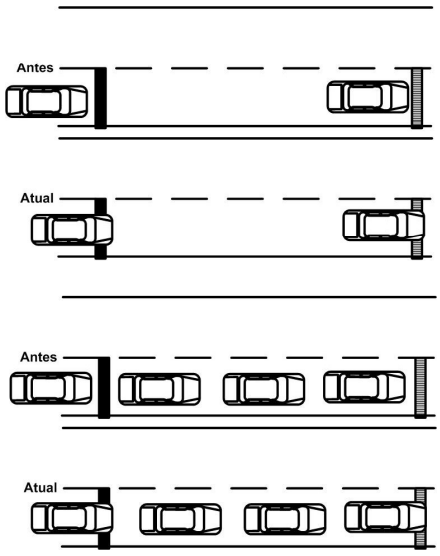


Figura 4-8 – Situação IV

O resultado negativo apresentado na situação II da Tabela 4-1, representa o registro pelo detector de saída de um veículo e, também, a não detecção de veículo pelo detector de entrada (Figura 4-9). Pode-se concluir que há algum movimento na pista, mas não é possível saber quantidade de veículos no momento.

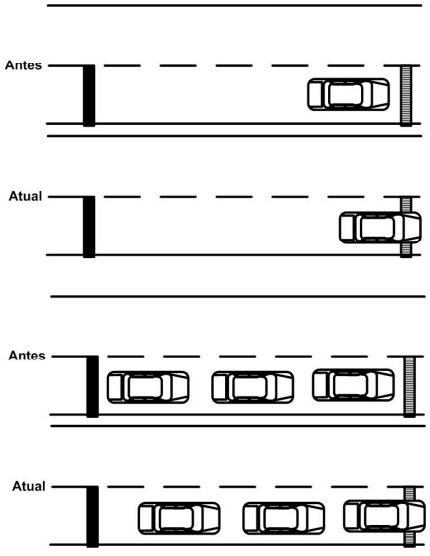


Figura 4-9 – Situação II

Quando o resultado é igual a um (1) (situação III da Tabela 4-1), o que se pode concluir é que um veículo foi registrado pelo detector de entrada e nenhum veículo pelo detector de saída, e que há algum fluxo de veículos na pista (Figura 4-10). Mas não é possível determinar a quantidade de veículos.

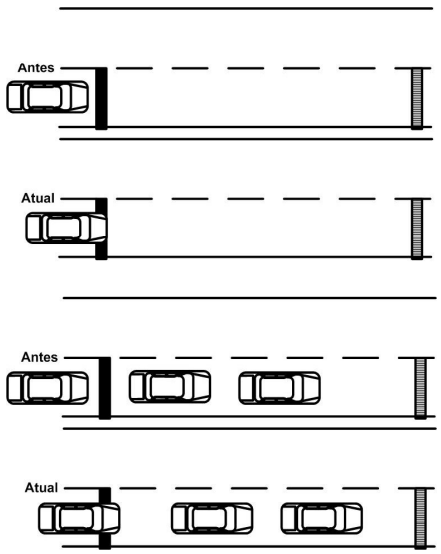


Figura 4-10 – Situação III

As situações apresentadas na Tabela 4-1 representam apenas a identificação de veículos por cada um dos detectores em um determinado instante *t*, e utilizando a equação (4.1) só será possível detectar se veículos entraram e/ou saíram do perímetro estabelecido pelos detectores e nunca a quantidade de veículos da pista, o que é necessário para o experimento.

Por este motivo, a equação (4.1) deve ser alterada para possibilitar saber a quantidade de veículos da pista num instante *t*, com a inclusão da quantidade de veículos que existiam na pista no instante anterior ao momento atual (*t -1*), e será dada por:

$$q_v = q_a + d_e - d_s \tag{4.2}$$

onde:

- q_v é a quantidade de veículos no instante atual *t*.
- q_a é a quantidade de veículos no instante anterior (*t –1*).

- d_e é o valor recuperado pelo detector de entrada no momento *t.*
- d_s é o valor recuperado pelo detector de saída no momento *t*.

Para a validação da equação (4.2) foi efetuado uma simulação com oitenta (80) entradas aleatórias (item 0), que obteve como resultado a quantidade de veículos que estavam na pista no determinado momento. No entanto, com o resultado do cálculo não foi possível determinar como estava o tráfego de veículos em uma pista, ou seja:

- O fluxo está parado?
- O fluxo está parando?
- O fluxo está começando a se movimentar?
- O fluxo está se movendo?

Comparando a quantidade de veículos da pista no momento presente (q_v) com a quantidade de veículos da pista no momento passado (q_a) no resultado da simulação (item 0) foi possível identificar três (3) situações distintas (Tabela 4-2):

Tabela 4-2 – Comparação entre q_v e q_a na Simulação

Situação	Comparação	Variação
I	$q_v = q_a$	$\Delta q = 0$
II	$q_v > q_a$	$\Delta q > 0$
III	$q_v < q_a$	$\Delta q < 0$

A situação II da Tabela 4-2 indica que a quantidade de veículos da faixa no momento presente (q_v) é maior que a quantidade de veículos da pista no momento passado (q_a) e ocorre somente quando o detector de entrada (d_e) registra a presença de veículo.

Na situação III da Tabela 4-2, que ocorre quando somente o detector saída (d_s) registra a presença de um veículo, a quantidade de veículos da faixa

no momento presente (q_v) é menor que a quantidade de veículos da pista no momento passado (q_a).

Quando a quantidade de veículos da faixa a no momento presente (q_v) é igual a quantidade de veículos da pista no momento passado (q_a) tem-se a situação I da Tabela 4-2. Esta situação ocorre quando nenhum dos detectores (o de entrada ou o de saída) registrou a presença de veículos ou quando ambos detectores registram a presença de veículos.

As situações II e III da Tabela 4-2 podem representar duas (2) situações distintas:

- Quando $q_v > q_a$ ($\Delta q > 0$), indica que há mais veículos na pista do que anteriormente, ou seja, mais veículos entraram na pista do que saíram.
- Se $q_v < q_a$ ($\Delta q < 0$), indica que há menos veículos na pista do que anteriormente, ou seja, mais veículos saíram da pista do que entraram, e permite supor que o fluxo de veículos está ficando mais rápido ou está iniciando a se movimentar.

Nas duas situações, não há como concluir a respeito da situação do trânsito, apenas supor que: no primeiro caso, a Situação II, que tráfego de veículos está lento ou está parando; e no segundo, a Situação III, que o fluxo de veículos está ficando mais rápido ou está iniciando.

A situação I da Tabela 4-2 quando $q_v = q_a$ ($\Delta q = 0$) pode ocorrer por causa de dois (2) motivos:

- Quando nenhum dos dois detectores identificou veículos ($d_e = 0$ e $d_s = 0$), que permite supor que o tráfego está parado; ou
- Quando ambos os detectores identificaram veículos ($d_e = 1$ e $d_s = 1$), que permite supor que o fluxo de veículos está fluindo normalmente.

Assim para determinar a situação do fluxo de veículos é necessário calcular a variação da quantidade de veículos na pista, que pode ser dada por:

$$\Delta q = q_v - q_a \tag{4.3}$$

onde:

- Δq é a variação da quantidade de veículos da pista.
- q_v é a quantidade de veículos no instante atual (t).
- q_a é a quantidade de veículos no instante anterior ($t - 1$).

Mas conhecer qual é a variação da quantidade de veículos da pista (Δq) só permite concluir, no caso da Situação I da Tabela 4-2, que o fluxo de veículos está parado, pois não entrou e nem sai veículos, ou seja, nenhum dos dois detectores identificou veículos ($d_e = 0$ e $d_s = 0$). Para as demais situações, é possível supor a respeito do trânsito que:

- Quando $\Delta q = 0$, $d_e \neq 0$ e $d_s \neq 0$, que entrou um veículo e que saiu, outro.
- Quando $\Delta q > 0$, que apenas entrou um veículo.
- Quando $\Delta q < 0$, que apenas saiu um veículo.

Em contrapartida, com a identificação da quantidade de veículos na pista só é possível chegar a duas (2) conclusões:

- Que não há veículos na pista, quando $q_v = 0$.
- Que há veículos, quando $q_v > 0$.

A inexistência de veículos ($q_v = 0$) é conclusiva: o trânsito está parado. No entanto, o conhecimento sobre a existência de veículos ($q_v > 0$), por sua vez, não é significativo. Pois, esta quantidade só terá significado se puder ser determinar se há a pouco ou muito veículos.

Por este motivo, é necessário que se estabeleça uma quantidade de veículos que a pista comporta (q_{max}) para que possa ser possível identificar outras condições do fluxo de veículos da pista, além de sem veículo. A Tabela 4-3 apresenta uma representação qualitativa da quantidade de veículos em relação a q_{max}, e cada linha da coluna Veículos corresponde a um conjunto difuso.

Tabela 4-3 – Quantidade de Veículos

Situação	Condição	Veículos
I	$q = 0$	Nenhum
II	$0 < q_v \leq \frac{q_{\max}}{2}$	Pouco
III	$\frac{q_{\max}}{2} < q_v \leq \frac{2.q_{\max}}{3}$	Normal
IV	$q_v > \frac{2.q_{\max}}{3}$	Muito

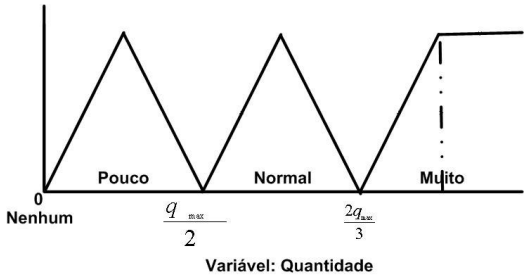


Figura 4-11 – Conjuntos Difusos de Veículos

Apesar de todas as informações já identificadas como necessárias para o controle de fluxo de veículos, ainda não foi possível determinar a fluidez do trânsito, ou seja, se existindo movimento, a velocidade média da via está dentro da sua normalidade ou abaixo do esperado ou acima do esperado. Assim é necessário ter conhecimento, também, da velocidade média da pista (v_m) que expressa por:

$$v_m = \frac{v_s + v_e}{2} \tag{4.4}$$

onde:

- v_m é a velocidade média da pista no momento *t*.
- v_s é a velocidade identificada pelo detector de entrada da pista no momento *t*.
- v_e é a velocidade identificada pelo detector de saída da pista no momento *t*.

OBSERVAÇÃO: Os detectores de entrada e de saída devem ser capazes de fornecer a velocidade do veículo que foi identificado.

Da mesma forma que foi necessário conhecer ou estabelecer uma quantidade máxima de veículos na via (q_{max}) é necessário ter determinado a velocidade média da pista (v_m) para poder identificar a fluidez do trânsito na mesma. Esta fluidez pode ser expressa, também, de forma qualitativa (Tabela 4-4), onde cada um das linhas da coluna Velocidade é um conjunto difuso ():

Tabela 4-4 – Velocidade Média

Situação	Condição	Velocidade
I	$v_m = 0$	Parado
II	$0 < v_m \leq \frac{v_{\max}}{2}$	Baixa
III	$\frac{v_{\max}}{2} < v_m \leq v_{\max}$	Regular
IV	$v_m > v_{\max}$	Alta

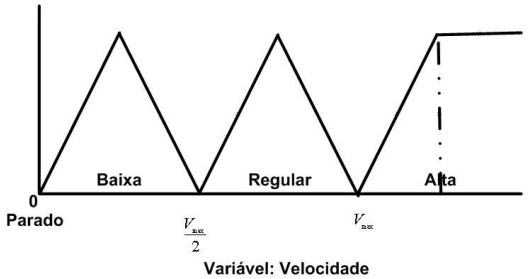


Figura 4-12 – Conjuntos Difusos de Velocidade

Com base em todas essas informações (a variação da quantidade de veículos –Δq –; quantidade veículos da pista – q_v –; a velocidade média – v_m) é possível determinar qual o tráfego de veículos que é uma das entradas da RNL.

Esta informação será expressa, também, qualitativamente, e não apresentará a situação do trafego, mas sim qual a sua interferência para a melhoria da regulagem dos semáforos. Desta forma, essa indicação poderá ser (Tabela 4-5): Nenhum; Baixo; Médio; Alto.

Tabela 4-5 – Indicações para a RNL

Situação	v_m	q_v	Δq	Indicação
I	PARADO	ZERO	ZERO	NENHUM
II	PARADO	POUCO	ZERO	BAIXO
III	PARADO	NORMAL	ZERO	MEDIO
IV	PARADO	MUITO	ZERO	ALTO
...	...	...	...	...
XLVI	REGULAR	MUITO	MOVENDO	ALTO
XLVII	ALTA	POUCO	MOVENDO	MEDIO
XLVIII	ALTA	MUITO	MOVENDO	ALTO
XLIX	ALTA	MUITO	MOVENDO	ALTO

A Tabela 4-5 (Anexo 10)) completa foi gerada a partir de um simulador com quarenta e nove (49) possibilidades. O conteúdo da coluna “Indicação” desta tabela, que será a entrada da RNL, foi estabelecido após a análise de das situações possíveis apresentadas, e que podemos denominar de “base do conhecimento” deste controlador.

A partir desta base do conhecimento foi estabelecido um conjunto de regras, que utilizadas no CFV, desenvolvido em linguagem de programação “C” (Anexo 08), permitirão qualificar o fluxo de veículos:

Regra	Descrição da Regra
R01	Se o Volume for NENHUM, então o Fluxo será NENHUM.
R02	Se a Volume for MUITO e a Velocidade for ALTA, então o Tráfego será BAIXO.
R03	Se a Volume for MUITO e a Velocidade for NORMAL, então o Tráfego será MÉDIO.
R04	Se a Volume for MUITO e a Velocidade for BAIXA, então o Tráfego será ALTO.
R05	Se o Volume for NORMAL e a Velocidade for ALTA, então o Tráfego será BAIXO.
R06	Se o Volume for NORMAL e a Velocidade for NORMAL, então o Tráfego será MÉDIO.
R07	Se o Volume for NORMAL e a Velocidade for BAIXA, então o Tráfego será ALTO.
R08	Se o Volume for POUCO e a Velocidade for ALTA, então o Tráfego será BAIXO.
R09	Se o Volume for POUCO e a Velocidade for NORMAL, então o Tráfego o será MÉDIO.
R10	Se o Volume for POUCO e a Velocidade for BAIXA, então o Tráfego será ALTO.

Além da qualificação do fluxo de veículos, este controlador irá registrar cada uma das velocidades média calculadas e cada uma das quantidades de veículos. Estas informações serão utilizadas futuramente.

4.3 Controlador Semafórico da Intersecção

O Controlador Semafórico da Intersecção tem o objetivo de estabelecer a indicação luminosa dos semáforos do cruzamento sob a sua responsabilidade. Ele pode trabalhar de com três (3) estratégias distintas (Figura 4-13):

- Plano de Tempo Fixo;

- Plano de Tempo Fixo com Verde Variável; e
- Plano de Tempo Variável com Verde Variável.

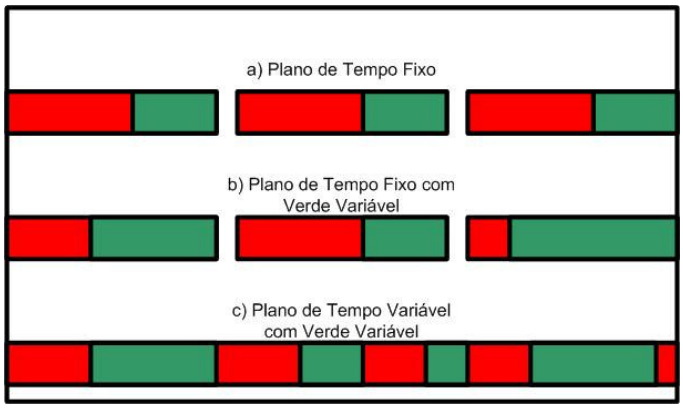


Figura 4-13 – Estratégias de Controle de Semáforos

O Controlador Semafórico de Intersecção – CSI, que será desenvolvido, trabalhará com as duas (2) últimas estratégias, a de Plano de Tempo Fixo com Verde Variável e a de Plano de Tempo Variável com Verde Variável. No entanto, a estratégica de Plano de Tempo Fixo com Verde Variável somente será utilizada quando o CSI não conseguir se comunicar com a RNL da Intersecção e vice-versa, tendo em vista que a inexistência da comunicação não pode prejudicar o trânsito controlado pelos semáforos.

No caso da estratégia de Plano de Tempo Variável com Verde Variável, este controlador deverá receber da RNL qual a indicação luminosa dos semáforos da via principal e qual a indicação luminosa dos semáforos da via secundária. Todas as possibilidades de indicações luminosas do semáforo da via principal e da via secundárias são apresentadas na Tabela 4-6.

Tabela 4-6 – Composição das Indicações Luminosas

Situação	Indicações Luminosas	
	Via Principal	Via Secundária
I	Verde	Vermelho
II	Amarelo	Vermelho
III	Vermelho	Vermelho
IV	Vermelho	Verde
V	Vermelho	Amarelo

Ao analisar a Tabela 4-6 pode-se identificar três (3) situações distintas que são apresentadas na Tabela 4-7:

- A situação I representa a ordem para que o fluxo da via principal esteja liberado e é expresso pela indicação luminosa verde do semáforo desta via, enquanto que o da via secundária deve ficar vermelho;
- As situações IV e V representam a ordem para que o fluxo da via principal esteja bloqueado e é expresso pela indicação luminosa vermelha para o semáforo desta via, enquanto que o da secundária pode ficar vermelho, ou amarelo; e
- A situação III representa a ordem para que ambos os fluxos fiquem bloqueados e é expresso para indicação luminosa vermelha tanto para o semáforo da via principal como o da secundária.

Tabela 4-7 – Ordem versus Composições das Indicações Luminosas

Situação	Indicações Luminosas		Ordem
	Via Principal	Via Secundária	
I	Verde	Vermelho	Fluxo liberado da via principal
II	Amarelo	Vermelho	Fluxo bloqueado para via principal
III	Vermelho	Vermelho	Fluxo bloqueado para ambas as vias
IV	Vermelho	Verde	Fluxo bloqueado para via principal
V	Vermelho	Amarelo	Fluxo bloqueado para via principal

Baseado nas possibilidades de ordens estabelecidas na Tabela 4-7 pode-se concluir que a RNL não necessita enviar dois (2) parâmetros para este controlador, mas apenas um (1) parâmetro para estabelecer:

a) A liberação do fluxo da via principal, neste caso o semáforo desta via ficará verde enquanto o da secundária ficará vermelho;
b) O bloqueio do fluxo da via principal, neste caso o semáforo desta via ficará vermelho enquanto o da secundária ficará verde; e
c) O bloqueio do fluxo de ambas as vias, neste caso tanto o semáforo da via principal como o da secundária ficarão vermelho.

Como saída deste controlador haverá duas informações:

- A primeira é o comando para os semáforos da intersecção que controla, contendo as indicações luminosas de cada um dos pares da intersecção.
- A segunda, a informação para a RNL sobre como está a situação da via principal, podendo ser:
    - Liberado, quando a via principal estiver com a indicação luminosa verde;
    - Bloqueado, quando a via principal estiver com a indicação vermelha ou amarela ou intermitente; e
    - Ambos, quando as duas vias estiverem com o fluxo bloqueado, ou seja, a indicação luminosa dos semáforos estará vermelha.

Para o estabelecimento das regras de funcionamento deste controlador, foi necessário identificar todas as possibilidades de fases de cada um dos pares de semáforos (Tabela 4-8) de uma via. A importância da identificação das fases está em conseguir representar a inteligência do controlador, ou seja, a sua base de conhecimento.

Tabela 4-8 – Fases dos Semáforos da Intersecção

Situação	Par 1 de Sinais			Par 2 de Sinais		
	Anterior	Atual	Próxima	Anterior	Atual	Próxima
I	------	Intermitente	Vermelho	------	Intermitente	Vermelho
II	Intermitente	Vermelho	Verde	Intermitente	Vermelho	Vermelho
III	Vermelho	Verde	Amarelo	Vermelho	Vermelho	Vermelho
IV	Verde	Amarelo	Vermelho	Vermelho	Vermelho	Vermelho
V	Amarelo	Vermelho	Vermelho	Vermelho	Vermelho	Verde
VI	Vermelho	Vermelho	Vermelho	Vermelho	Verde	Amarelo
VII	Vermelho	Vermelho	Vermelho	Verde	Amarelo	Vermelho
VIII	Vermelho	Vermelho	Verde	Amarelo	Vermelho	Vermelho

As situações I e II da Tabela 4-8 representam as duas (2) únicas situações onde não há uma seqüência padrão, como nos demais casos, por isso, cada um representou uma exceção no processamento:

Exceção	Regra	Situação
I	Quando qualquer um dos semáforos estiver Intermitente, a próxima indicação luminosa para ambos os semáforos será Vermelha.	I
II	Quando ambos os semáforos estiverem Vermelho e a indicação luminosa anterior do par de semáforos da via principal for intermitente, a indicação luminosa do par de semáforos da via principal será Verde, enquanto que o par da via secundária será Vermelho.	II

Para as demais situações da Tabela 4-8 será considerado que as indicações luminosas possam ser representadas por valores numéricos: 0, Intermitente; 1, verde; 2, amarelo; e 3, vermelho. Então são estabelecidas as seguintes regras:

Identificação da Próxima Indicação Luminosa dos Semáforos:

Regra 1: Calcular o valor da próxima indicação de cada par de semáforos dada por:

$$p_i = s_i + 1 \tag{4.5}$$

onde:

- p_i é a nova indicação luminosa que o par semáforo *i* deverá ter (coluna Próxima da Tabela 4-8).
- s_i é a indicação que o par semáforo *i* está no momento atual (coluna Atual da Tabela 4-8).

Regra 2: Se o valor calculado p_i for maior que 3 e a indicação luminosa anterior (coluna Anterior da Tabela 4-8) for igual a 2, então o p_i será igual a 3.

Regra 3: Se o valor calculado p_i for maior que 3 e a indicação luminosa atual (coluna Atual da Tabela 4-8) do outro par de semáforo for igual a 1 ou igual a 2, então o p_i será igual a 3.

Regra 4: Se o valor calculado p_i for maior que 3 e as regras 2 e 3 não forem atendidas, então o p_i será igual a 1.

Regra 5: Se não existir uma ordem de alteração das indicações luminosas, a ordem de mudança será o valor identificado da próxima indicação.

Alteração da Indicação Luminosa:

Regra 6: Alterar a indicação luminosa de acordo com as regras anteriores, caso tenha sido uma ordem ou caso o tempo máximo para a fase do semáforo já tenha sido atingida.

- A alteração da indicação luminosa requer que seja recuperado um tempo mínimo de apresentação, que foi estabelecido baseado em estudos anteriores do fluxo de veículos da via.

Regra 7: Se a ordem de alteração para o par 1 de semáforos não for igual à recebida e convertida ou se a ordem de alteração para o par 2 de semáforos não for igual à recebida e convertida, fazer nova alteração da indicação luminosa, a partir da regra 1.

Baseado nas sete (7) regras estabelecidas, um simulador foi desenvolvido (item 0). Após alguns ajustes, o mesmo apresentou-se estável, pois conseguiu entender e aplicar as ordens estabelecidas, mantendo as transições obrigatórias entre cada umas das fases, mas reduzindo o tempo mínimo de cada uma das interseções para um tempo pré-determinado que denominaremos de tempo reduzido (t_{red}).

A importância do estabelecimento do tempo reduzido (t_{red}) está em garantir o atendimento da ordem da RNL, melhorando o fluxo de veículos da via.

Outro ponto importante deste controlador é que todos os tempos dos ciclos utilizados serão armazenados para futura utilização. Com o objetivo de permitir capturar um tempo ideal em alguns momentos quando não houver comunicação com a RNL.

4.4 Rede Neural Local

A Rede Neural Local – RNL, que é o terceiro componente para a solução do problema da dissertação, tem o objetivo de estabelecer qual o próximo estado em que a via principal deve ficar, após o processamento pelo CSI. Por definição, os estados possíveis são:

- Liberado, quando o trânsito da via principal deve ter o direito de passagem enquanto a via secundária fica parada.
- Bloqueado, quando o tráfego da via secundária deve ter o direito de passagem enquanto a via principal fica parada; e

- Ambos, quando tanto o fluxo da via principal como a da secundária devem ficar parados.

Para atingir o seu objetivo, a RNL necessitará ter conhecimento da situação do tráfego das duas (2) vias que compõem o cruzamento, a principal e a via secundária. Estas informações serão fornecidas pelos CFV das vias, o da principal (CFV1) e o da secundária (CFV2), e poderão ser:

- Nenhum, quando não há fluxo de veículos na via;
- Baixo, quando há pouco trânsito na via;
- Médio, quando há um tráfego normal na via; e
- Alto, quando há muito trânsito na via.

A utilização destas duas (2) informações pela RNL irá permitir que a rede estabeleça qual o estado em que a via principal do cruzamento deve ficar. Por isso, foi necessário estabelecer todas as combinações possíveis entre ambas situações identificadas pelos CFV, bem como a possível ordem associada com a situação do trânsito das vias da intersecção (Tabela 4-9):

- Se os fluxos de ambas as vias forem nenhum, então ambos os tráfegos ficarão bloqueados (Situação I);
- Se o fluxo da via principal é inferior ao da secundária, então o seu tráfego será bloqueado para que os veículos da via secundária tenham direito de passagem (Situações II, III, IV, VII, VIII, X e XII);
- Se o fluxo da via principal é superior ao da secundária, então o seu fluxo será liberado enquanto os veículos da via secundária ficaram parados (Situações V, IX, XIII, XIV, XV);
- Se existirem fluxos em ambas as vias e eles forem iguais, então não foi possível estabelecer qual das vias deve ter o direito de passagem enquanto a outra fica parada (Situações VI, XI, XVI).

Tabela 4-9 – Fluxos das Vias versus Saída da RNL

Situação	Fluxo		Ordem
	Via Principal	Via Secundária	
I	Nenhum	Nenhum	Ambos
II	Nenhum	Baixo	Bloqueado
III	Nenhum	Médio	Bloqueado
IV	Nenhum	Alto	Bloqueado
V	Baixo	Nenhum	Liberado
VI	Baixo	Baixo	????
VII	Baixo	Médio	Bloqueado
VIII	Baixo	Alto	Bloqueado
IX	Médio	Nenhum	Liberado
X	Médio	Baixo	Bloqueado
XI	Médio	Médio	????
XII	Médio	Alto	Bloqueado
XIII	Alto	Nenhum	Liberado
XIV	Alto	Baixo	Liberado
XV	Alto	Médio	Liberado
XVI	Alto	Alto	????

As ordens estabelecidas na Tabela 4-9 foram baseadas, exclusivamente, nas informações sobre a situação do trânsito em cada uma das vias do cruzamento. Mas há algumas situações – VI, XI e XVI – onde não foi possível determinar qual das vias deve ter o seu fluxo liberado ou bloqueado. Por isso, é necessário identificar alternativas que permitam estabelecer a ordem para estas situações:

a) Deixar que a via principal sempre tenha o direito de passagem, enquanto que a via secundária fica bloqueada; ou
b) Deixar que a via que tem o direito de passagem, ou seja, a indicação luminosa para ela é verde, continue com o seu fluxo liberado enquanto a outra fica bloqueada.

c) Deixar que o controlador dos semáforos da intersecção continuasse com o seu processamento e determinasse qual o próximo estágio ou intervalo dos dois (2) semáforos.

Na primeira alternativa, a RNL ordenará ao CSI que o fluxo da via principal fique liberado, mantendo o atual estágio do semáforo, se ele for verde, ou acelerando o ciclo até que o próximo intervalo seja verde. No entanto, esta solução poderá gerar um congestionamento na via secundária, pois o semáforo da via secundária ficará verde em um estágio menor do que o necessário.

Para que a manutenção da liberação do trânsito da via que está com o estágio do semáforo igual a verde, a segunda alternativa, a RNL ordenará ao CSI que mantenha o estágio atual através do acréscimo do intervalo no ciclo deste semáforo. Isto também poderá ocorrer o mesmo problema apresentando do parágrafo anterior, ou seja, gerar o congestionamento da outra via. Se este congestionamento for na via principal do cruzamento, será um grande problema.

Nas duas (2) alternativas apresentadas anteriormente, um congestionamento pode ser gerado, por causa, principalmente, de uma regulagem ruim do sistema de semáforos do cruzamento. Portanto, pode-se concluir que estas duas (2) alternativas não são boas. Assim sendo, resta a última alternativa a ser analisada: deixar que o ciclo se complete normalmente no CSI, sem interferência da RNL.

A última alternativa, a mais simples de todas, a ordem da RNL para o CSI nas situações VI, XI e XVI da Tabela 4-9 será NENHUMA, ou seja, não haverá ordem da RNL para o CSI. Desta forma, este controlador interpretará como a inexistência de comunicação e adotará, por definição, a estratégia de Plano de Tempo Fixo com Verde Variável. Esta situação poderá gerar congestionamento também, mas pelo menos será garantido que cada uma das vias tenha um ciclo completo.

Considerando que a última alternativa será adotada, além das ordens anteriores (liberado, bloqueado e ambos) que a RNL poderia repassar ao CSI, haverá mais uma, a NENHUMA. Como o significado desta ordem é informar ao controlador que o atual ciclo dos semáforos deve completado sem a interferência da RNL, a Tabela 4-10 é a Tabela 4-9 complementada com esta nova ordem:

Tabela 4-10 – Fluxos das Vias versus Saída da RNL

Situação	Fluxo		Ordem
	Via Principal	Via Secundária	
I	Nenhum	Nenhum	Ambos
II	Nenhum	Baixo	Bloqueado
III	Nenhum	Médio	Bloqueado
IV	Nenhum	Alto	Bloqueado
V	Baixo	Nenhum	Liberado
VI	Baixo	Baixo	**NENHUMA**
VII	Baixo	Médio	Bloqueado
VIII	Baixo	Alto	Bloqueado
IX	Médio	Nenhum	Liberado
X	Médio	Baixo	Bloqueado
XI	Médio	Médio	**NENHUMA**
XII	Médio	Alto	Bloqueado
XIII	Alto	Nenhum	Liberado
XIV	Alto	Baixo	Liberado
XV	Alto	Médio	Liberado
XVI	Alto	Alto	**NENHUMA**

A Tabela 4-10 apresenta as situações possíveis do trânsito de ambas as vias que compõem a intersecção ou cruzamento, bem como qual a ordem que deve ser repassada ao controlador semafórico da intersecção – CSI. No entanto, se a ordem a ser repassada representar o atual estado do semáforo da via principal, a informação que deveria ser encaminhada ao CSI é NENHUMA, ou seja, que o atual ciclo deve ser mantido até segunda ordem.

Considerando a premissa expressa na última frase do parágrafo anterior, a Tabela 4-10 poderia ter mais uma coluna que representasse o atual estado da via principal da intersecção (Tabela 4-11 que é uma parte da tabela existente no item 0). Desta forma se o novo estado do semáforo da vai principal a ser repassado ao CSI fosse o seu atual estado, esta ordem não necessitaria ser encaminhada:

Tabela 4-11 – Fluxos da Via, Estado Atual e Saída da RNL

Situação	Fluxo		Estado Atual	Ordem Prevista	Ordem Real
	Via Principal	Via Secundária			
I	Nenhum	Nenhum	Ambos	Ambos	**NENHUMA**
II	Nenhum	Nenhum	Liberado	Ambos	Ambos
III	Nenhum	Nenhum	Bloqueado	Ambos	Ambos
IV	Nenhum	Baixo	Ambos	Bloqueado	Bloqueado
V	Nenhum	Baixo	Liberado	Bloqueado	Bloqueado
VI	Nenhum	Baixo	Bloqueado	Bloqueado	**NENHUMA**
VII	Nenhum	Médio	Ambos	Bloqueado	Bloqueado
VIII	Nenhum	Médio	Liberado	Bloqueado	Bloqueado
IX	Nenhum	Médio	Bloqueado	Bloqueado	**NENHUMA**
...	....	...	...	...	...
XLIII	Alto	Médio	Ambos	Liberado	Liberado
XLIV	Alto	Médio	Liberado	Liberado	**NENHUMA**
XLV	Alto	Médio	Bloqueado	Liberado	Liberado
XLVI	Alto	Alto	Ambos	Nenhuma	**NENHUMA**
XLVII	Alto	Alto	Liberado	Nenhuma	**NENHUMA**
XLVIII	Alto	Alto	Bloqueado	Nenhuma	**NENHUMA**

A Tabela 4-11 completa com quarenta e oito (48) situações é a base de dados que será utilizada na validação da RNL:

- As colunas Fluxo da Via Principal e Fluxo da Via Secundária correspondem, respectivamente, às informações fornecidas pelos

Controladores de Fluxo de Veículos da via principal (CFV1) e o da via secundária (CFV2);

- O Estado Atual corresponde ao estado atual do semáforo da via principal e é esta informação que pode ser repassada a RNL pelo Controlador de Semáforos da Intersecção ou pela própria RNA;
- A Ordem Prevista é o estado que a RNA deverá determinar após o seu processamento, não considerando o estado atual do semáforo; e
- A Ordem Real é o estado que a RNA repassará ao Controlador de Semáforos da Intersecção após o processamento e considerando qual o estado atual do semáforo da via principal do cruzamento.

Com relação ao parâmetro Estado Atual do semáforo da via principal, optou-se que a mesma será repassada pela própria rede, quando encaminhar esta ordem ao CSI. Assim sendo, a RNL a ser desenvolvida deverá ser, obrigatoriamente, uma RNA recorrente – RNR com três (3) camadas (Figura 4-14), com apenas uma (1) camada intermediária além das camadas de entrada e de saída. A opção de utilizar uma (1) única camada intermediária está relacionada com a capacidade desta rede em aprender tudo que uma rede *n* camadas intermediária aprende (BRASIL, 2002, p.42).

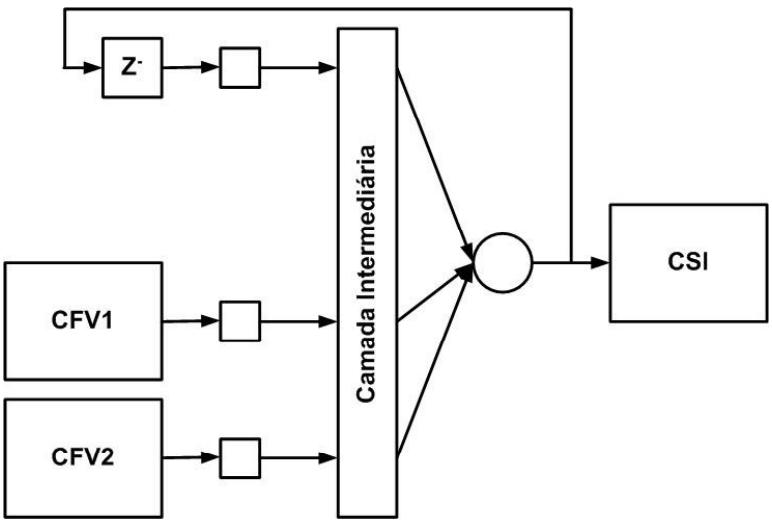


Figura 4-14 – Rede Neural Local Inicial

Na rede proposta (Figura 4-14), a camada de entrada terá três (3) neurônios, enquanto que a de saída, apenas um (1). Os neurônios da camada de entrada correspondem, respectivamente, ao fluxo de veículos da via principal (CFV1), ao fluxo de veículos da via secundária (CFV2) e da Ordem Real repassada ao CSI da intersecção. Está última informação será originada do único neurônio da camada de saída.

Tendo em vista que a RNA necessita ser recorrente, foi necessário definir qual o algoritmo que a mesma deveria utilizar, e, dentre os vários identificados na literatura, optou-se pelo *Backpropagation Time Through* – BPTT, que é uma extensão do *Backpropagation* – BP com a aplicação de janelas de tempo. A justificativa para a escolha pelo BPTT como o algoritmo da rede está na possibilidade de utilizar o *Backpropagation*, no processo de validação da rede, e depois, após a sua validação, incluir as janelas de tempo para o seu processamento recorrente.

Com a definição da quantidade de camadas da RNR e do algoritmo que a mesma deve utilizar, o próximo passo foi estabelecer como a rede seria e testada. Para atingir este objetivo, foram estabelecidas duas (2) possibilidades, uma através da utilização de ferramentas disponíveis no mercado e a outra, através do desenvolvimento de um *software* utilizando uma linguagem de programação. Tendo em vista que há, no mercado, vários *softwares* disponíveis e testados, optou-se, num primeiro momento, pela primeira alternativa.

Para a avaliação de *softwares* foi necessário identificar aqueles que tivessem cópia de avaliação disponível para *download* através da *Internet* e que pudessem ser executados sob o sistema operacional *Windows XP*, e, o mais importante, que permitisse implementar uma RNR. A escolha recaiu sobre três (3) produtos: o EasyNN, o Matlab e o *Neuro Solutions*. A massa de dados a ser utilizada nesta avaliação será a Tabela 4-10.

O primeiro *software* avaliado foi o EasyNN Versão 6.0 da *Neural Planner Software*. Este produto, segundo pesquisa realizada na sua ferramenta de ajuda, permite criar redes totalmente conectadas, mas não contém informação se é possível criar redes recorrentes ou não.

Esta ferramenta só permite a criação de uma nova rede, após a importação ou definição de uma massa de dados que será utilizada pela mesma. Como na importação ou na criação desta massa é definida a quantidade de neurônios da camada de entrada e, também, a da camada de saída, na criação da nova rede só é necessário estabelecer quantas camadas intermediárias serão necessárias, variando de um (1) e três (3), bem como, a quantidade de neurônios de cada uma das camadas.

Durante o processo de criação da RNA, não foi possível definir que a mesma deveria ser recorrente. Baseado no resultado gráfico da RNA criada foi possível concluir que é uma MLP com três (3) camadas: a de entrada com dois (2) neurônios e de cor amarela; a intermediária com quatro (4) neurônios e de cor azul; e a de saída, com um (1) neurônio e de cor roxa. Como não foi possível criar nesta ferramenta uma RNA recorrente, ela foi desconsiderada.

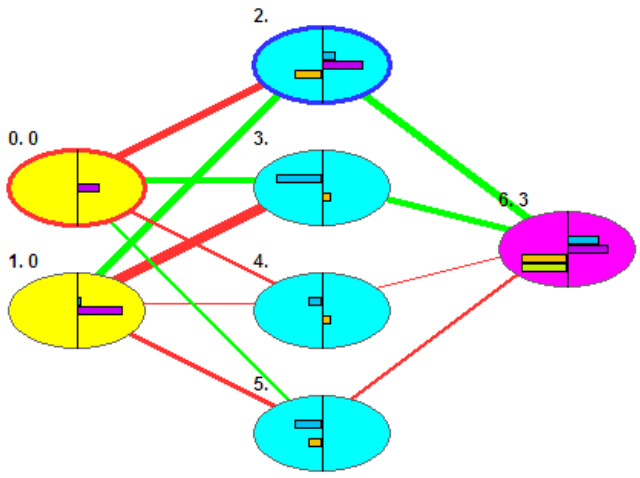


Figura 4-15 – RNA Criada pelo EasyNN

A segunda ferramenta avaliada foi a *Neuron Solutions* Versão 5.1.2600.0 da *NeroDimension, Inc*. Neste produto, segundo as informações na sua ferramenta de ajuda, é possível criar todos os tipos de RNA, inclusive uma recorrente, completa ou parcialmente conectada, utilizando o algoritmo *backpropagation through time*.

Baseado nos dados da Tabela 4-10 foi criado uma RNA parcialmente recorrente, com dois (2) neurônios na camada de entrada, um (1) na camada de saída. A própria ferramenta sugeriu que a camada intermediária, neste caso definida como uma (1), tivesse quatro (4) neurônios. A recorrência parcial desta rede partia da camada intermediária e não foi possível, como estabelecido para a solução do problema, alterar a RNA para que a recorrência partisse da camada de saída (Figura 4-14). Por causa disto, esta ferramenta foi desconsiderada.

A última ferramenta avaliada foi o *Matlab* Versão 7.0 da *The MathWorks Inc*. Este *software* é um dos produtos mais referenciados e utilizados em trabalhos acadêmicos sobre RNA. No entanto, para esta dissertação ele foi desconsiderado por, segundo a sua ferramenta de ajuda, só permitia a criação de dois tipos de redes recorrentes, a Elman e a Hopfield, que apresentam arquitetura diferente da necessária para a resolução do problema, composta por duas (2) camadas, uma de entrada e outra de saída.

Como não foi possível identificar um *software* no mercado que funcionasse no ambiente operacional *Windows XP* para a construção da RNA recorrente proposta na Figura 4-14, optou-se pelo desenvolvimento de uma aplicação para a resolução do problema. Como esta aplicação necessitaria ser executada em qualquer máquina sob o sistema operacional *Windows XP*, sem a necessidade de instalação do *software* e de outros componentes, foi escolhida a linguagem de programação “C” sem interface gráfica do ambiente integrado de desenvolvimento DEV-CPP Versão 4.9.9.2. A utilização desta linguagem e deste ambiente para o desenvolvimento dos dois (2) controladores, o CFV e o CSI, foi o principal motivo desta escolha.

Tendo o desenvolvimento da aplicação como alternativa escolhida, a próxima etapa foi descrever a lógica de funcionamento deste *software*, que deverá permitir a criação de uma RNR com várias camadas utilizando algoritmo *Backpropagation Through Time – BPTT*. Como este modelo pode ser implementado a partir de uma RNA *Backpropagation*, que é estática, com a utilização de janelas de tempo, em que a entrada da rede utiliza trechos dos dados temporais como se eles formassem um padrão estático (BRAGA, CARVALHO e LUDEMIR, 2000, p.209; HAYKIN, 2001, p. 49), optou-se pelo desenvolvimento da solução em duas etapas:

1. Na primeira etapa, o *software* permitirá a criação de uma RNA MLP não recorrente com algoritmo *backpropagation*; e
2. Na segunda etapa, após a validação do primeiro aplicativo, será inserida a parte de recorrência, ou seja, a janela de tempo.

Na primeira parte, a solução deveria permitir a construção de uma RNA MLP estática com algoritmo BP, conforme a Figura 4-16. Esta RNA terá quatro (4) entradas e uma (1) única saída. E a sua construção foi baseada em alguns trabalhos de vários autores disponibilizados na Internet, dentre os quais: John Bullinaria[21]; Karsten Kutza[22]; Beiro R. Tschanggelar[23]; Kiyoshi Kawaguchi[24]. O resultado final foi um aplicativo onde a quantidade de camadas da rede e de neurônios em cada uma das camadas seriam parâmetros recuperados do primeiro registro do arquivo da massa de dados estabelecida (Anexo 11).

---

[21] John Bullinaria (http:/www.cs.bham.ac.uk/~jxb/NN/nn.html em 03/03/2006).

[22] Karsten Kutza (hhtp://www.neural-networks-at-your-fingertips.com/bpn.html em 03/03/2006).

[23] Beiro R. Tschanggelar (http://www.ibrctses.com/delphi/neuralnets.html em 03/03/2006).

[24] Kyoshi Kawaguchi (http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node73.html)

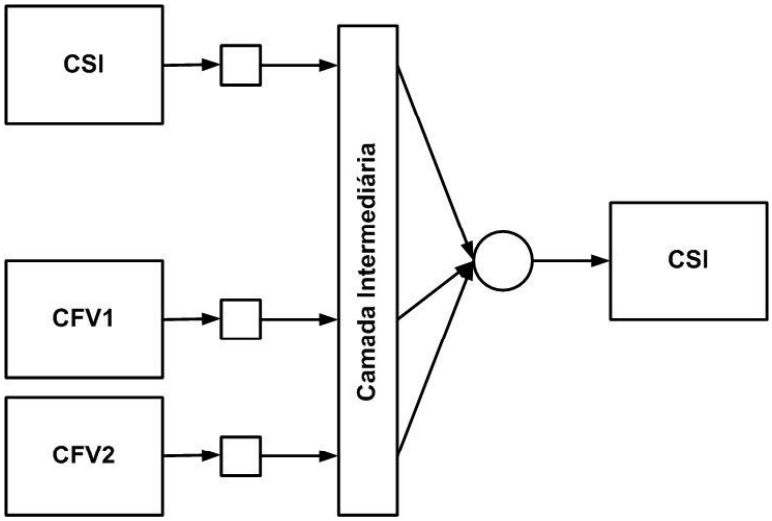


Figura 4-16 – Rede Neural Local Inicial sem Recorrência

Esta solução deveria ser genérica, ou seja, deveria ter a capacidade de criar e treinar uma RNA com vários neurônios de entrada, várias camadas intermediárias e a camada de saída com vários neurônios. Por isso, utilizou-se a facilidade de construção de matrizes unidimensionais ou bidimensionais ou tridimensionais dinâmicas, ou seja, sem pré-determinação da quantidade de ocorrências de cada dimensão:

- Uma (1) dimensão:

```
int * G01_Qtd_Neuronio; // definição de uma matriz unidimensional

// alocação dinâmica das ocorrências da matriz
G01_Qtd_Neuronio = (int *) calloc (G04_Qtd_Camadas, sizeof(int));
```

- Duas (2) dimensões:

```
double ** G01_Dado; // definição de uma matriz bidimensional

// Alocação dinâmica de n ocorrências para a 1ª dimensão da matriz
G01_Dado = (double **) calloc (G04_Qtd_Camadas, sizeof(double *));

// Alocação dinâmica de n ocorrências para a 2ª dimensão de uma
//  determinada 1ª dimensão
G01_Dado[p040_01_Indx] = (double *)
         calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
         sizeof(double));
```

- Três (3) dimensões:

```
double *** G01_Peso_Recuperado;
      // definição de uma matriz tridimensional

// Alocação dinâmica de n ocorrências para a 1ª dimensão da matriz
G01_Peso_Recuperado = (double ***) calloc (G04_Qtd_Camadas,
sizeof(double **));

// Alocação dinâmica de n ocorrências para a 2ª dimensão de uma
//  determinada 1ª dimensão
G01_Peso_Recuperado[p040_01_Indx]= (double **)
   calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
   sizeof(double *));

// Alocação dinâmica de n ocorrências para a 3ª dimensão de uma
//  determinada 2ª dimensão de uma 1ª dimensão.

G01_Peso_Recuperado[p040_01_Indx][p040_02_Indx] =
   (double *) calloc (G05_Qtd_Neuronios[p040_01_Indx -1] + 1,
 sizeof(double));
```

Como o algoritmo *backpropagation* tem duas (2) fases: a *forward* e a *backward*. Esta aplicação deve contemplar as duas. No *forward*, primeiro calcula-se a função net_i^P ou somatório, que é um dos parâmetros para ser calculado a função de ativação.

A função net_i^P é somatório dos produtos do peso de cada entrada de um neurônio vezes o respectivo valor da entrada do neurônio, e é expresso por (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.77; ZURADA, 1992, p. 177):

$$net_i^P = \sum_j w_{ij} u_j^p + \theta_i \tag{4.6}$$

onde:

- w_{ij} é o peso da conexão do neurônio *j* para o neurônio *i*;
- u_j^p são as entradas para o neurônio *j*;
- net_i^P é a função de entrada para o neurônio *i*; e
- θ_i é o bias, sempre apresenta a entrada +1 e um peso adaptativo w_0.

Esta função net_i^P , a equação (4.6), foi codificada como:

```
G01_Dado[p100_01_Indx][p100_02_Indx] +=
   G01_Dado[p100_01_Indx - 1][p100_03_Indx] *
   G01_Peso_Recuperado[p100_01_Indx][p100_02_Indx][p100_02_Indx];
```

Para que a função net_i^P possa ser calculada é necessário que os pesos da rede sejam os seus valores definidos, aleatoriamente (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.88). Esta definição aleatória foi codificada em uma função a ser chamada:

```
// a função para recuperar o peso aleatoriamente
double p030_Encontra_Real(float p030_Menor_Valor, float
p030_Maior_Valor)
{
   double p030_Aux = 0;
   p030_Aux = ((((double)(rand()%10))/10) - 0.5);
   return (p030_Aux);
}
// a chamada da rotina para recuperar um peso aleatório
G01_Peso_Recuperado[p050_01_Indx][p050_02_Indx][p050_03_Indx]
                  = p030_Encontra_Real(-0.5, 0.5);
```

O resultado da Equação (4.6) é um dos componentes da função de ativação. A função escolhida foi a sigmóide que é expressa por (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.87; ZURADA, 1992, p. 179):

$$f(net_i^P) = \frac{1}{1 + e^{-net_i^P}} \tag{4.7}$$

onde:

- $f(net_i^P)$ é a função de ativação. $f(net_i^P) = y_i^p$.
- net_i^P é o somatório expresso pela Equação (4.6).

A função de ativação $f(net_i^P)$, Equação (4.7), no aplicativo, foi codificada como uma função interna:

```
float p080_Sigmoide (float p080_Dado)
{
   return (1.0 / (1.0 + exp(-p080_Dado)));
}
```

Tanto a função net_i^P, equação (4.6), e a função $f(net_i^P)$, Equação (4.7), devem ser aplicadas a cada um dos neurônios de cada uma das camadas da rede, inclusive a camada de saída. Neste ponto, conclui-se a fase *forward* do BP.

A fase *backward* inicia-se com o cálculo do erro quadrático (E). Este cálculo é necessário pois para cada entrada, a saída da rede y^p, pode ser diferente do valor desejado d^p. O erro quadrático representa a soma dos erros (E) do nodos de saída da rede que é expresso por (BRAGA, CARVALHO e LUDEMIR, 2001, p. 83; DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p.84; ZURADA, 1992, p. 176):

$$E^p = \frac{1}{2}\sum_{j=1}^{k}(d_i^p - y_i^p)^2 \tag{4.8}$$

onde:

- d_i^p é a saída desejada para a unidade *i* quando o padrão *p* é fixado;
- y_i^p é a saída calculada para a unidade *i* quando o padrão *p* é fixado.

O erro quadrático, Equação (4.8), foi codificado como:

```
G01_Erro += 0.5 * raizquadrada
(G01_Saida_Esperada[p100_02_Indx]-G01_Saida_Calculada[p100_02_Indx]);
```

Após o cálculo do erro quadrático, a equação (4.8), inicia-se o processo de ajuste de cada um dos pesos da entrada através da aplicação da seguinte equação (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 79; ZURADA, 1992, p. 177):

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{4.9}$$

onde:

- η é a constante de é proporcionalidade;
- ∂E é a derivada do erro quadrático; e
- ∂w_{kj} é a derivada do peso entre os neurônios j e k.

A Equação (4.9) pode ser expressa como (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 79; ZURADA, 1992, p. 177; BRAGA, CARVALHO e LUDEMIR, 2001, p.90):

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial (net_k)} \frac{\partial (net_k)}{\partial w_{kj}} \tag{4.9}$$

De acordo com Zurada (1992, p.177), e De Azevedo, Brasil e De Oliveira (2000, p. 84), o resultado do desenvolvimento do segundo fator da Equação (4.9) será:

$$\frac{\partial (net_k)}{\partial w_{kj}} = y_j \tag{4.10}$$

onde:

- y_j é a entrada do neurônio da rede.

O primeiro termo da Equação (4.9) pode ser definido como δ_i^p (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 84):

$$\delta_J^P = \frac{\partial E}{\partial (net_J^P)} \tag{4.11}$$

e assim se obtém uma regra atualizada, que é equivalente a regra delta que resultará em um gradiente descendente na superfície de erro de forma que o peso mudará de acordo com (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 84):

$$\Delta_p w_{kj} = -\eta \delta_i^p u_j^p \tag{4.12}$$

Na Equação (4.12), o principal problema para o cálculo da variação do peso está em identificar qual δ_i^p deve ser para cada uma das unidades i na rede. Assim sendo, para determinar o δ_i^p é necessário aplicar a regra da cadeia para descrever esta derivada parcial como o produto de dois fatores, onde um deles reflete a mudança no erro com uma função de saída da unidade e o outro reflete a mudança da saída com uma função de mudança na entrada. Então teremos que (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 85):

$$\delta_i^p = -\frac{\partial E^p}{\partial(net_i^p)} = \frac{\partial E^p}{\partial y_i^p}\frac{\partial y_i^p}{\partial(net_i^p)} \tag{4.13}$$

O segundo fator da Equação (4.13), considerando a equação $f(net_i^P) = y^p$, pode ser expresso por (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 85):

$$\frac{\partial y_i^p}{\partial(net_i^j)} = f`(net_i^p) \tag{4.14}$$

Para o computar o primeiro fator da Equação (4.13) é necessário considerar duas situações para a unidade i (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 85; ZURADA, 1992, p. 183):

- Se for uma unidade de saída da rede; e
- Se não for é uma unidade de saída da rede.

Quando a unidade i for uma unidade de saída da rede, ela será dada por (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 85; ZURADA, 1992, p. 183):

$$\frac{\partial E_i^p}{\partial y_i^j)} = -(d_o^p - y_o^p) \tag{4.15}$$

sendo o δ_i^p será dado por

$$\delta_i^p = -(-(d_i^p - y_i^p).f`(net_i^p) \Rightarrow \delta_i^p = (d_i^p - y_i^p).f`(net_i^p) \tag{4.16}$$

e o $\Delta\ w_{ij_i}^{\ p}$ será dado por:

$$\Delta_p w_{ij} = \eta(d_i^p - y_i^p) f`(net_i^p) u_j^p \tag{4.17}$$

que foi codificada como:

```
G01_Delta[p140_01_Indx] =
   (G01_Saida_Esperada[p140_02_Indx - 1] -
   G01_Saida_Calculada [p140_02_Indx - 1])*
   G01_Saida_Calculada [p140_02_Indx - 1] *
      (1 - G01_Saida_Calculada [p140_02_Indx - 1]);
```

Na segunda situação, quando a unidade i não é uma unidade de saída da rede e sim uma unidade intermediária $k = h$, não é possível saber qual a contribuição da unidade para o erro da saída da rede. No entanto, a medida do erro pode ser escrita como uma função das entradas da rede na camada intermediária para a da saída. Então, $E^p = E^p(net_1^p, net_2^p, net_3^p, ..., net_n^p)$ e utilizando a regra da cadeia pode ser escrito como (DE AZEVEDO, BRASIL e DE OLIVEIRA, 2000, p. 86):

$$\frac{\partial E^p}{\partial y_i^p} = \sum_h \frac{\partial E^p}{\partial (net_h^p)} \frac{\partial (net_h^p)}{\partial y_i^p} \tag{4.18}$$

$$\frac{\partial E^p}{\partial y_i^p} = \sum_h \frac{\partial E^p}{\partial (net_h^p)} \frac{\partial}{\partial y_i^p} \sum w_{hk} y_k^p \tag{4.19}$$

$$\frac{\partial E^p}{\partial y_i^p} = \sum_h \frac{\partial E^p}{\partial (net_h^p)} w_{hi} \tag{4.20}$$

$$\frac{\partial E^p}{\partial y_i^p} = \sum_h \delta_h^p w_{hi} \tag{4.21}$$

Desta forma δ_i^p será dado por:

$$\delta_i^p = f`(net_i^p).\sum_h \delta_h^p w_{hi} \tag{4.22}$$

e o $\Delta\ w_{ij_i}{}^p$ será dado por:

$$\Delta_p w_{ij} = u_j^p f`(net_i^p).\sum_h \delta_h^p w_{hi} \tag{4.23}$$

que foi codificada em duas partes. Na primeira, correspondendo a Equação (4.21):

```
p140_03_Aux += 1;
   G01_Peso_Recuperado[p140_01_Indx][p140_02_Indx][p140_05_Indx]
   * G01_Delta [p140_01_Indx + 1];
```

e a outra, Equação (4.23), onde p140_03_Aux corresponde a Equação (4.21):

```
G01_Delta[p140_01_Indx] = p140_03_Aux *
   G01_Dado [p140_01_Indx][p140_02_Indx] *
   (1 - G01_Dado [p140_01_Indx][p140_02_Indx]);
```

Após o cálculo do $\Delta_p w_{kj}$, deve-se iniciar o processo de a atualização dos pesos dos neurônios entre uma camadas com a camada posterior, que no programa foi codificado como:

```
for (p140_01_Indx=0;p140_01_Indx<G04_Qtd_Camadas-1;p140_01_Indx++)
{
   for (p140_02_Indx=0;p140_02_Indx<G05_Qtd_Neuronios[p140_01_Indx];
   p140_02_Indx++)
   {
      for (p140_05_Indx = 1; p140_05_Indx <=
            G05_Qtd_Neuronios[p140_01_Indx + 1]; p140_05_Indx++)
      {
        G01_Peso_Variacao[p140_01_Indx][p140_02_Indx][p140_05_Indx] =
            G07_Taxa_Aprendizado * G01_Delta [p140_01_Indx + 1] +
            G06_Momento *
         G01_Peso_Recuperado
          [p140_01_Indx][p140_02_Indx][p140_05_Indx];
         G01_Peso_Recuperado
          [p140_01_Indx][p140_02_Indx][p140_05_Indx] =
          G01_Peso_Variacao[p140_01_Indx][p140_02_Indx][p140_05_Indx];
      }
   }
}
```

Após a criação do programa da RNA utilizando o algoritmo *backpropagation* tradicional, iniciou-se o processo de teste e de validação do mesmo. Para tanto, utilizou-se como massa de dados a Tabela 4-12 e foi criado uma rede com a seguinte característica:

- Três (3) camadas na rede;
- Dois (2) neurônios na camada de entrada;
- Dois (2) neurônios na camada intermediária; e
- Um (1) neurônio na camada de saída.

Tabela 4-12 – Dados para Teste da Rede

x_1	x_2	y_d
0	0	0
1	0	1
0	1	1
1	1	0

Como parâmetros para o teste desta rede foram definidos:

- Ciclos = 100;
- Taxa de Aprendizado = 0,6;
- Momento = 0,8; e
- Erro Alvo = 0,05.

Durante a fase de teste, a quantidade de neurônios da camada intermediária foi aumentada para três (3), quatro (4) e cinco (5), e o melhor resultado verificou-se com a rede com dois (2) neurônios (Tabela 4-13). Este resultado comparado com o treinamento de uma rede similar em uma das ferramentas avaliada mostrou-se semelhante.

Tabela 4-13 – Erro no Teste da Rede

Neurônios na camada intermediária	Erro		
	Máximo	Mínimo	Média
2	0,519346	0,504939	0,512539
3	0,520294	0,507844	0,513756
4	0,516146	0,511718	0,511718
5	0,519107	0,508706	0,514334

A validação do programa construído foi considerada satisfatória. Então, o próximo passo é estabelecer a arquitetura da rede que será validada pelo programa e os seus parâmetros. Esta rede também terá três (3) camadas e a quantidade de neurônios da camada intermediária poderia ser definida

empiricamente, conforme relatam Braga, Carvalho e Ludemir (2000, p. 55). No entanto, utilizando o Teorema de Kolmogorov-Nielsen (KOVÁCS, 2002, p.107), e sabendo que há quatro (4) neurônios (*n*) na camada de entrada, foi possível determinar que a camada intermediária terá, no máximo, nove (9) neurônios:

$$2n+1 \Rightarrow n=4 \Rightarrow 2.4+1=9$$

O Teorema de Kolmogorov-Nielsen permite determinar a quantidade máxima de neurônios da camada intermediária e não a quantidade ideal de neurônios desta camada, que só pode ser identificada após simulações. Na simulação efetuada, com os parâmetros abaixo relacionados, o número ideal de neurônios foi quatro (4) (Tabela 4-14):

- Momento = 0,8.
- Taxa de Aprendizagem = 0,1;
- Épocas = 500;
- Elementos de Teste = 200.

Tabela 4-14 – Neurônios da Camada Intermediária

Neurônios	Acertos	Erros	% Acertos
1	0	200	0,0
2	7	193	3,5
3	8	192	4,0
4	10	190	5,0
5	0	200	0,0
6	1	198	0,5
7	0	200	0,0
8	0	200	0,0
9	0	200	0,0

Com a determinação da quantidade de neurônios da camada intermediária, inicia-se os procedimentos de ajuste dos demais parâmetros da

RNA. O primeiro parâmetro a sofre ajuste foi a taxa de erro que foi estabelecida em 0,000001 (Tabela 4-15), segundo os parâmetros abaixo relacionados:

- Momento = 0,8.
- Taxa de Aprendizagem = 0,6.
- Neurônios da Camada Intermediária = 4.
- Épocas = 500;
- Elementos de Teste = 200.

Tabela 4-15 – Taxa de Erro

Erro	Acertos	Erros	% Acertos
0,1	10	190	5,0
0,01	7	193	3,5
0,001	8	192	4,0
0,0001	10	190	5,0
0,00001	10	190	5,0
0,000001	11	189	5,5

A quantidade de épocas, o segundo parâmetro, foi determinada em 15.000 (Tabela 4-16), com os seguintes parâmetros:

- Momento = 0,8.
- Taxa de Aprendizagem = 0,6.
- Neurônios da Camada Intermediária = 4.
- Erro = 0,000001.
- Elementos de Teste = 200.

Tabela 4-16 – Número de Épocas

Épocas	Acertos	Erros	% Acerto
5.000	7	193	3,5
10.000	8	192	4,0
15.000	11	189	5,5
20.000	6	194	3,0

A Taxa de Aprendizagem, terceiro parâmetro, foi identificado como 0,2 (Tabela 4-17) de acordo os parâmetros abaixo relacionados:

- Momento = 0,8.
- Neurônios da Camada Intermediária = 4.
- Épocas = 15.000.
- Erro = 0,000001.
- Elementos de Teste = 200.

Tabela 4-17 – Taxa de Aprendizagem

Taxa Aprendizagem	Acertos	
	Mínimo	Máximo
0,6	191	194
0,5	192	195
0,4	192	193
0,3	192	194
0,2	193	195
0,1	192	195

O valor do último parâmetro ajustado, o Momento, foi determinado em 0,8 (Tabela 4-18), de acordo com os parâmetros abaixo:

- Neurônios da Camada Intermediária = 4.
- Épocas = 15.000.
- Erro = 0,000001.
- Taxa de Aprendizagem = 0,2.
- Elementos de Teste = 200.

Tabela 4-18 – Momento

Momento	Acertos	
	Mínimo	Máximo
0,9	5	9
0,8	191	195
0,7	189	194
0,6	6	11
0,5	x-x	x-x
0,1	7	10

Com todos os parâmetros ajustados, relacionados abaixo, iniciou-se o processo de teste da da RNA recorrente com cento e trinta e seis (136) simulações, onde apenas vinte e oito (28) testes, aproximadamente 20,59%, apresentou mais de nove (9) erros em duzentos (200) elementos de teste (Tabela 4-19)

- Neurônios da Camada Intermediária = 4.
- Épocas = 15.000.
- Erro = 0,000001.
- Momento = 0,8.
- Taxa de Aprendizagem = 0,2.
- Elementos de Teste = 200.

Tabela 4-19 – Simulações na RNA

	Número de Simulações	Simulação		
		Elementos	Erros	% Acertos
	16	200	4	98,0
	29	200	5	97,5
	32	200	6	97,0
	8	200	7	96,5
	10	200	8	96,0
	13	200	9	95,5
	28	200	> 9	-X-X
Total	136			

Em mais de 79% das simulações, a quantidade de acertos foi superior a 95,0%, assim sendo o aplicativo contendo uma RNA estática MLP com algoritmo BP pode ser considerado válida. E o próximo passo foi transformar esta RNA em uma recorrente.

Para tanto, foi incluída a janela de tempo de tempo de $\frac{1}{100}$ segundos, no aplicativo validado, com o objetivo de ter a recorrência na rede. Este tempo foi estabelecido como aceitável para que a regulagem dos semáforos ocorra em tempo-real.

A RNA recorrente (Figura 4-17) desenvolvida terá na camada de entrada terá quatro (4) nós:

- O primeiro, é a informação repassada pelo controlador de fluxo de veículos da via principal, o CFV1;
- O segundo, é a informação repassada pelo controlador de fluxo de veículos da via secundária, o CFV2;
- O terceiro, é a informação repassada pelo controlador semafórico da outra intersecção, o CSI_{out}; e

- O quarto e último, é a informação referente a ordem repassada para o controlador semafórico da sua intersecção, o CSI. Esta informação representa a recorrência da rede.

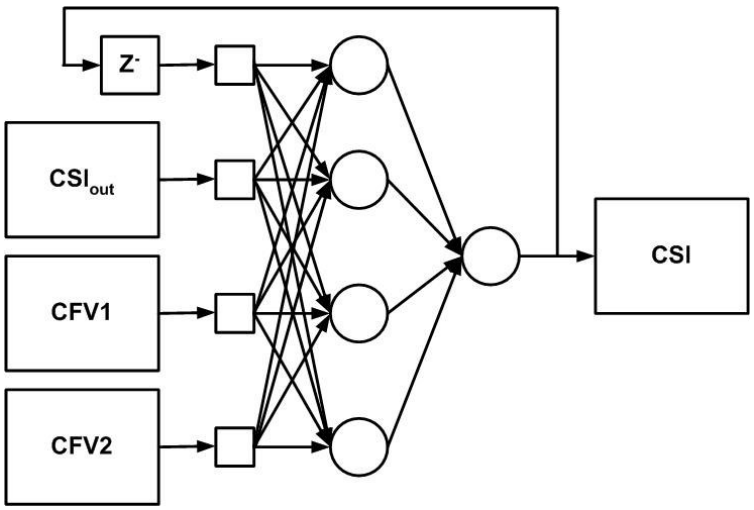


Figura 4-17 – Rede Neural Local – RNL

De acordo com as informações de entrada, a RNL deverá determinar as seguintes ordens para o CSI da Intersecção:

- Liberado – Neste caso, após o processamento pelo CSI, o semáforo da via principal deverá estar com a indicação luminosa VERDE, enquanto que o da via secundária deverá estar VERMELHO;
- Bloqueado – Neste caso, após o processamento pelo CSI, o semáforo da via principal deverá estar com a indicação luminosa VERMELHA e o da via secundária, VERDE.
- Ambos – Neste caso, após o processamento pelo CSI, ambos os semáforos, o da via principal e o da via secundária, devem estar com a indicação luminosa VERMELHA.

A RNL só será capaz de determinar qual o estado do semáforo da via principal se as entradas esperadas (do CSI_{out}, do CFV1, do CFV2 e a sua recorrência) estiverem disponíveis. Caso contrário, ela não repassará nenhuma ordem para o CSI da intersecção.

5 CONCLUSÃO

O objetivo deste trabalho era definir e desenvolver uma RNA recorrente e um controle adaptativo para a regulagem de um sistema integrado de semáforos em tempo-real para melhoria do tempo de cor verde. Para que pudesse ser alcançado seria necessário:

- Identificar a melhor arquitetura de uma RNA recorrente para a resolução do problema.
- Identificar o melhor controle dinâmico para, em conjunto com a RNA, ajudar na resolução do problema.
- Demonstrar que a arquitetura de RNA recorrente proposta em conjunto com o controle dinâmico pode ser utilizada para a resolução do problema.
- Desenvolver um mecanismo para armazenamento e recuperação de todas as regulagens aplicadas ao sistema integrado de semáforos.

A arquitetura da RNA recorrente utilizada neste trabalho para atender o objetivo principal, foi composta de três (3) camadas:

- Uma (1) de entrada com quatro (4) neurônios:
  - O primeiro, é a informação repassada pelo controlador de fluxo de veículos da via principal, o CFV1;
  - O segundo, é a informação repassada pelo controlador de fluxo de veículos da via secundária, o CFV2;
  - O terceiro, é a informação repassada pelo controlador semafórico da outra intersecção, o CSI_{out}; e
  - O quarto e último, é a informação referente a ordem repassada para o controlador semafórico da sua intersecção, o CSI. Esta informação representa a recorrência da rede.
- Uma (1) intermediária com quatro (4) neurônios; e

- Uma (1) de saída com um (1) neurônio que corresponde a ordem que o CSI da Intersecção receberá.

O algoritmo de aprendizado escolhido foi o *Backpropagation Through Time* – BPTT, que é uma extensão do *Backpropagation* – BP tradicional E por não ser possível implementá-la em uma das ferramentas disponível no mercado que foram avaliadas, optou-se pelo seu desenvolvimento na linguagem de programação C (em ambiente não gráfico) dentro do ambiente integrado de desenvolvimento DEV-CPP.

O desenvolvimento do programa foi feito em duas (2) partes. Na primeira, construiu-se um código para tratamento de uma rede MLP com algoritmo para BP. Na segunda, após a sua validação, teve a inclusão da janela de tempo, transformando-se em uma BPTT.

A função da RNA criada era, basicamente, a partir da situação do fluxo de veículos das vias que compõem a intersecção, estabelecer se o fluxo de veículos da via principal ficaria bloqueado ou liberado, ou se as duas vias do cruzamento ficariam com o trânsito bloqueado. Para tanto, foi necessário desenvolver um controlador de fluxo de veículos, um controlador adaptativo, cuja responsabilidade era determinar como estava o tráfego das vias.

Além do controlador de fluxo de veículos, outro controlador adaptativo foi desenvolvido para controlar o ciclo dos semáforos da intersecção, o CSI. Este controlador tem a capacidade de trabalhar com as estratégias de plano de tempo fixo com verde variável e de plano de tempo variável com verde variável. A adoção das duas estratégias está na necessidade de manter o controle dos semáforos da intersecção mesmo que a comunicação entre este controlador e a RNA seja interrompida ou perdida.

Ambos os controladores têm a capacidade de armazenar as informações que geram no seu processamento. O CFV registrará as informações sobre do fluxo de veículos da via, incluindo o volume e a velocidade dos veículos das

pistas de cada uma das vias, e o CSI, os ciclos aplicados aos pares de semáforos da intersecção. Com estas informações, o conhecimento explícito, será possível conhecer as situações do trânsito nas intersecções sob o controle do sistema. Para tanto será necessário desenvolver futuramente um mecanismo de mineração de dados.

O conhecimento recuperado, através da mineração de dados, poderá ser utilizado na determinação dos ciclos nos novos semáforos instalados, onde a regulagem em tempo-real não será aplicada, inicialmente. Desta forma, pode-se utilizar os ciclos registrados de outras intersecções cuja característica do trânsito seja igual ou semelhante ao do cruzamento onde os novos semáforos serão instalados.

O CSI deve, ainda, ter a capacidade de identificar e recuperar a melhor regulagem que já aplicou em um determinado horário, e reutilizá-la quando não existir, por qualquer motivo, a comunicação com a RNR. Como esta informação dinâmica, ou seja, atualizada a cada ciclo dos semáforos, a sua utilização irá permitir um bom fluxo de veículos no cruzamento.

De acordo com a proposta deste trabalho, a utilização de uma RNA recorrente em conjunto com controladores adaptativos pode efetuar uma boa regulagem do sistema de semáforos de duas (2) intersecções em tempo-real. No entanto, a solução apresentada foi estabelecida para o fluxo de veículos em duas fases. Portanto, como estudo futuro é proposto a aplicação desta rede para um conjunto de vias com mais de duas (2) fases.

Além do estudo proposto, é um dos objetivos do autor, após a conclusão do mestrado, finalizar o desenvolvimento do sistema integrado de controle de semáforos com processamento distribuído e em tempo-real, cujos componentes iniciais são os dois (2) controladores e a RNR desenvolvidos neste trabalho. Espera-se que este produto esteja pronto até o final deste ano, e que possa ser comercializado a partir do início do próximo ano.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTP – Associação Nacinal de Transportes Públicos. *Transporte Humano: Cidades com Qualidade de Vida*. ANTP, [2000?].

BELLEMANS, T.; DE SCHUTTER, B.; DE MOOR, B. *Models for Traffic Control*. Journal A, vol. 43, no. 3-4, pp. 13-22, 2002.

BIORA, F. *at al*. *A Best Practice Manual for Innovative UTC Schemes*. Relatório de Priority Management for Vehicle Efficiency, Environment and Road Safety on Arterials - ITS University of Leeds - Mizar Automazione SpA - Città di Torino, December 1995.

BORGES, Tauler Teixeira; DE AZEVEDO, Haroldo R. *Controlador Fuzzy para Motor a Relutância*. Universidade Católica de Goiás; Universidade Federal de Uberlândia. [ca1993].

BRAGA, Antônio de P.; CARVALHO, André C. P. de L. F.; LUDEMIR, T. B. *Redes Neurais Artificiais; Teoria e Aplicações*. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora SA, 2000.

BRASIL. *Código de Trânsito Brasileiro*. Lei n° 9.503 de 23 de setembro de 1997.

BRASIL, Lourdes. M. *Inteligência Artificial: Modelos Conexionistas*. (Material de Estudos). Paraíba: Curso de Ciência da Computação do Departamento de Informática - DI da Universidade Federal da Paraíba - UFPB, 2002.

CAMPONOGARA, Eduardo; SOUZA, Silvia G.; KRAUS JUNIOR, Werner. *A Matemathical Programming Model for Urban Traffic Control*. Universidade Federal de Santa Catarina e Universidade Federal do Paraná, 1995.

CHIN, Daniel C.; SMITH, Richard H. *A Neural Network Application in Signal Tunning Control*. IEEE, 1996.

CHOY, Min C.; CHEU, Ruey Long; SRINIVASAN, Dipti; LOGI, Filippo; *Real-Time Coordinated Signal Control Using Agents with Online Reinforcement Learning*. 82nd Annual Meeting of the Transportation Research Board, November, 2002.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. *Distributed Systems: Concepts and Designs*. London: Addison-Wesley, 2001.

DA SILVA, Danyela M. LINDAU, Luís A. *Potencial dos Sistemas Avançados (APTS) no Transporte Coletivo Urbano por Ônibus*. in: Congresso de Pesquisa e Ensino em Transportes, 11, Rio de Janeiro, 1997, Anais. Rio de Janeiro, ANPET, 1997, v.1, p.510-518.

DA SILVEIRA, Paulo R.; DOS SANTOS, Winderson E. *Automação e Controle Discreto*. São Paulo: Erica, 2004 (6ª edição).

DAVIS, L. C. *Effect of adaptive cruise control systems on traffic flow*. Physical Review E 69 006110, 2004.

DAVENPORT, Thomas H.; PRUSAK, Lawrence. *Ecologia da informação: porque só a tecnologia não basta para o sucesso na era da informação*. São Paulo: Futura, 1998.

DE AMORIM, Bruno P. *Descoberta de uma Plataforma Híbrida para Descoberta de Conhecimento em Bases de Dados*. Dissertação de mestrado em Ciência da Computação da Universidade Federal de Pernambuco, 2004.

DE AZEVEDO, Fernando M. *Contribution to the Study of Neural Networks in Dynamical Expert System*. Tese de doutorado na Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, 1993.

DE AZEVEDO, Fernando M.; BRASIL, Lourdes M.; OLIVEIRA, Roberto C. L. *Redes Neurais com aplicação em controles e em sistemas especialistas*. Florianopólis, SC: Bookstores, 2000.

DENATRAN - Departamento Nacional de Trânsito. *Manual de Semáforos*. 2ª edição. Brasília: DENATRAN, 1984.

DENATRAN – Departamento Nacional de Trânsito. *Sistema Nacional de Estatísticas de Trânsito e Departamentos Estaduais de Trânsito.* (www.denatran.gov.br em 07/02/2006).

DE OLIVEIRA, Roberto C. L. *Rede Neural com Dinâmica Interna Aplicada a Problemas de Identificação e Controle Não-Linear*. Tese de doutorado em Engenharia Elétrica da Universidade Federal de Santa Catarina, Florianopólis-SC, 1999.

DE RE, Angelita Maria. *Sistemas Conexionistas Adaptativos Aplicados a Problemas de Controle de Tráfego Urbano*. Dissertação de Mestrado em Ciências da Computação da Universidade Federal de Santa Catarina, Florianópolis-SC, 1995.

DETRAN/DF – Departamento de Trânsito do Distrito Federal. *Estatísticas*. (www.detran.df.gov.br em 07/02/2006).

DIAS, Marina R. B.; BARROS, Laércio C. *O Uso de Controladores Fuzzy para Resolução de Equações Diferenciais Ordinárias*. Bio Matemática, IMECC, UNICAMP, nº 15, 2005, (p.147-156).

DORF, Richard C; BISHOP, Robert H. *Sistemas de Controle Moderno*. Rio de Janeiro: LTC – Livros Técnicos e Científicos Ltda, 2001.

FELICIANO NETO, Acácio; FURLAN, José D.; HIGA, Wilson. *Engenharia da Informação: Metodologia, técnicas e ferramentas*. São Paulo: McGraw-Hill, 1988 (p. 209-234).

FERNANDEZ, Marcial P.; PEDROZA, Aloysio de C. P.; DE REZENDE, José Ferreira. *Otimização de Controlador Fuzzy para Provisionamento de Recursos em ambiente DiffServ através de Algoritmo Genético*. COPPE/PEE – Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, [ca2001].

FERNANDEZ, Marcial P.; PEDROZA, Aloysio de C. P.; DE REZENDE, José F. *Implementação de Políticas de Gerenciamento com Lógica Fuzzy e Algoritmo Genético Visando à Melhoria da Qualidade de Serviço (QOS).* Revista da Sociedade Brasileira de Telecomunicações, Volume 18, Número 2, Outubro/2003.

FERREIRA, Enrique D.; SUBBRAHMANIAN, Eswaram; MANSTETTEN, Dietrich. *Intelligente agents in descentralized traffic control*. 2005 IEEE Intelligent Transportation Systems Conference Procceding. Oakland (CA), USA, 25-29/Agosto/2001.

FGVSP. *Gestão do Conhecimento*. (http://www.fgvsp.br/conhecimenot/home.htm em 23/02/2006).

FURLAN, José D. *Modelagem de Objetos através da UML: Análise e Desenho Orientados a Objetos*. São Paulo: Makron Books, 1998.

GARBER, Nicholas J.; HOEL, Lester A. *Traffic and Highway Engineering*. Revised Second Edition. USA: Brooks/Cole Publishing Company, 1999.

GARDNER, Mark P. *Highway traffic Monitoring*. A2B08 Committee Traffic Monitoring. South Dakota, USA, (2000?).

HALL, Fred L. *Traffic Stream Characteristics*. (Cap. 2) (2000?).

HAYKIN, S. *Redes Neurais: Princípios e Práticas*. Porto Alegre: Bookman, 2001.

HAYKIN, Simon; VAN VEEN, Barry. *Sinais e Sistemas*. Porto Alegre: Bookman, 2001.

HERNÁNDEZ, Josefa; CUENA, José; MOLINA, Matin. *Real-time Traffic Management Through Knowledge-Based Models: The TRYS Aprroach*. Departament of Artificial Intelligence Techical, University of Madrid, 1997(?).

HO, Fu-Sheng; IOANNOU, Petros. *Traffic Flow Modeling and Control Using Artificial Neural Networks*. IEEE Controls Systems. Outubro/1996 (p.16-21).

HOMBURGER, Wolfgang S.; HALL, Jerome W.; LOUTZENHEISER, Roy C. RELLY, Willian R. *Fundamentals of Traffic Engineering*. Berkeley, USA: Institute of Transportatin Studies, University of Califórnia, 1989.

HUANG, Hui-Min. *An Architecture and a Methodology for Intelligent Control*. Intelligent Control, IEEE Expert, 1996 (p.46-55).

KENNEDY, Norman; KELL, James H.; HOMBURGER, Wolfgang. *Fundamentals of Traffic Engineering*. 6ª edição. The Institute of Transportation and Traffic Engineering, University of Califórnia, Berkeley, 1966.

KIRSCHFINK, Heribert; HERNÁNDEZ, Josefa; BOERO, Marco. *Intelligent Traffic Management Models*. ESIT, ESIT 2000, 14-15 September 2000, Aachen, Germany.

KOVÁCS, Z. L. *Redes Neurais Artificiais: Fundamentos e Aplicações: Um texto básico*. São Paulo: Editora Livraria da Física, 2002.

LEDOUX, Corinne. *An Urban Traffic Flow Model Integrating Neural Networks*. Transpn Res.-C, Vol. 5, No. 5, pp. 287-300, 1997. Elsevier Sciense Ltd.

LEE, Jee-Hyong; LEE-KWANG, Hyung. *Distributed and Cooperative Fuzzy Controllers for Traffic Intersections Group*. IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews, Vol. 29, nº 2, maio/1999.

LO, Hong K.; CHAN, Y. C.; CHOW, Andy H. F. *A New Dynamic Traffic Control System: Performance of Adaptive Control Strategies for Over-saturated Traffic*. Intelligent Transportation Systems Conference Proceeding, Oakland, Califórnia, USA, 25-29 de agosto de 2001.

LUGER, G. F. *Artificial Intelligence: structures and strategies of complex problem solving*. Londres, Inglaterra: Pearson Education Limited, 2002 (Cap.10 - p.417-468).

MANIKONDA, Vikram; *at al. Autonomous Agentes for Traffic Simulation and Control*. Transportation Research Record 1774, 2001 (p.1-10).

MEIRELLES, Alexandre A. de C. *Sistemas de Transportes Inteligentes: Aplicação da Telemática na Gestão do Trânsito Urbano*. [ca1998].

MELO, Josué M. de. *Gerenciando o conhecimento*. Revista Treinamento e Desenvolvimento, abril/1998 (pág. 18-20).

MIRCHANDANI, Pitu B.; NOBE, Steven A.; WU, Wenji W. *Online Turning Proportion Estimation in Real-Time Traffic-Adaptive Signal Control*. Transportation Research Record 1748, 2001 (p.80-86).

MOLINA, Martín; ROBLEDO, Mónica; FERNÁNDEZ, Alberto. *A Propose-and-revise System for Real-time Traffic Management*. ESIT 2000, 14-15 September 2000, Aachen, Germany.

MORESI, Eduardo (Organizador). *Metodologia da Pesquisa*. Universidade Católica de Brasília, 2004 (Material da disciplina Metodologia da Pesquisa do Mestrado de Gestão do Conhecimento e Tecnologia da Informação).

MULLER, Theo H. J.; MISKA, Marc.; VAN ZUYLEN, Henk J. *Monitoring traffic under congestion: base for dynamic assigniment in online predict modells*. TRB 2005 Anual Meeting, 2005.

NAKAMITI, G.; FREITAS, R. *Adaptive, Real-Time Traffic Control Management*. International Journal of Automotive Technology, Vol. 3, N°. 3, 2002, (p. 89-94).

NAUMANN, Rolf; RASCHE, Rainer; TACKEN, Jurgen. *Managing autonomous vehicles at intersections*. IEEE Intelligent Systems (p.82-86). Maio/Junho 1998.

NASCIMENTO JÚNIOR, Cairo L.; YONEYAMA, Takashi. *Inteligência Artificial em Controle e Automação*. São Paulo: Editora Edgard Blucher, 2000.

NISEMBAUM, Hugo. *Gestão do conhecimento: como é possível as empresas administrarem o que sabem*. Revista Ser Humano, Ano XXXII, número 135, Agosto/1988 (pág. 54-55).

NONAKA, Ikujiro; TAKEUCHI, Hirotaka. *Criação do conhecimento na empresa: Como as empresas japonesas geram a dinâmica da inovação*. Rio de Janeiro: Campus, 1997.

OGATA, Katsuhiko. *Engenharia de Controle Moderno*. São Paulo: Prentice Hall, 2003 (4ª. edição).

OLIVEIRA, Rômulo S. de; FARGESY, Jean-Loup; MOREIRA, Guilherme O.; KRAUS JR., Werner. *Controle Ótimo de um Cruzamento Automatizado de Tráfego Urbano*. XIV Congresso Brasileiro de Automática, Natal-RN, 2002.

OLIVEIRA JR, Hime A. e. *Lógica Difusa: Aspectos práticos e aplicações*. Rio de Janeiro: Interciência, 1999.

OSSOWSKI, Sascha; CUENA, José; GARCIA-SERRANO, Ana. *A Case of Multiagent Decision Support: Using Autonomous Agents for Urban Traffic Control*. 6ª Conferência Ibero-americana sobre Inteligência Artificial: Progresso da Inteligência Artificial. Madri,ES: 1998.

PAGE-JONES, Meilir. *Projeto Estruturado de Sistemas*. São Paulo: McGraw-Hill, 1988 (p.107-149).

PAPACOSTAS, Constantinos S.; PREVEDOUROS, Panos D. *Transportation Engineering and Planning*. Second Edition. USA: Prentice-Hall, Inc, 1993.

PARK, Byungkyu; MESSER, Carroll J.; URBANIK II, Thomas. *A Genetic Algorithm-Based Signal Optimization Program for Oversaturated Intersections*. Texas Transportation Institute, College Station. [ca1998].

PHILLIPS, Charles L.; HARBOR, Royce D. *Sistemas de Controle e Realimentação*. São Paulo: Makron Books, 1996.

RICH, Elaine; KNIGHT, Kevin. *Inteligência Artificial*. São Paulo: Makron Books, 1993.

ROOZEMOND, Danko A.; ROGIER, Jan L. *Agent Controlled Traffic Lights*. ESIT-2000, Aachen, Germany, 2000.

RUSSEL, Stuart; NORVIG, Peter. *Inteligência Artificial*. 2a ed. Rio de Janeiro: Elsevier, 2004.

SADEK, Adel W.; SMITH, Brian L.; DEMETSKY, Michael J. *Dynamic Traffic Assignment*. Transportation Research Record, 1588, 1998.

SAITO, Mitsuru; FAN, Jianzhong. *Artificial Neural Network-Based Heuristic Optimal Traffic Signal Timing*. Computer-Aided Civil and Infrastructure Engineering, 2000 (p. 281-291).

SCHÖNHOF, Martin; HELBING, Dirk. *Traffic States and Their Implications for Traffic Modeling*. Institute for Economics and Traffic, Dresden University of Technology, Dresden, Germany, 2004.

SCHOPPEK, Wolfgang. *Examples, Rules, and Strategies in the Controle of Dynamic Systems*. University of Bayreuth, Germany. [ca2001].

SENNE, Edson L. F. *Primeiro Curso de Programação em C*. Florianópolis: Bookstore Livraria Ltda, 2003.

SERRA, Maurício R. G. *Aplicações de Aprendizagem por Reforço em Controle de Tráfego Veicular Urbano*. Dissertação do Mestrado em Engenharia Elétrica do Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina - UFSC, 2004.

SHAW, Alan. C. *Sistemas em Tempo Real*. Porto Alegre: Bookman, 2003.

STEWART, J. A.; LEPIK, K.; VAN AERDE, M. *Benefit Sensitivities of Adaptive Traffic Control Strategies at Isolated Traffic Signals*. Transportation Research Record 1692, 1998 (p.173-182).

SVEIBY, Karl Erik. *A nova riqueza da organização: gerenciando e avaliando patrimônios de conhecimento*. Rio de Janeiro: Campus, 1998.

TANEMBAUM, Andrew S.; VAN STEEN, Maarten. *Distributed Systems: Principles and Paradigms*. New Jersey, USA; Printice Hall, 2002.

VILANOVA, Luís M. *O Controle dos Semáforos em Tempo Real no Brasil*. Fundação Carlos Chagas (http://www.fcc.org.br;pesquisa/detranIdeias2.html disponível em 22/12/2004).

WEI, Junhua; WANG, Anlin; DU, Nianci. *Study of self-organizing controle of traffic signals in na urban network based on cellular automata*. IEEE transactions on vehicular technology, vol. 54, no.2, março/2005.

WIIG, Karl M. *Gestão do conhecimento: de onde veio e para onde vai?* Revista Empresas & Tendências, ano V, número 48, 1998 (pág. 6-18).

ZURADA, Jacek M. *Introduction to Artificial Neural Systemas*. Saint Paul, MN: West Publishing Company, 1992.

ANEXOS

Anexo 01 – 1º Simulador da Quantidade de Veículos da Via

```
#include <stdio.h>
#include <stdlib.h>

FILE *G_01_Arquivo;
char G_02_Registro [80] = "\0";
char *G_03_Endereco_Arquivo = "simulador_001.txt";

int P010_Num_Randomicos (int P01_Intervalo)
{
   return rand()% P01_Intervalo;
}

int main(int argc, char *argv[])
{
   int Ax_01_Index;
   int Ax_02_Detector_Inicial;
   int Ax_03_Detector_Final;
   int Ax_04_Quantidade_Veiculos;

   system("CLS");

   G_01_Arquivo = fopen (G_03_Endereco_Arquivo, "w");
   if (!G_01_Arquivo)
   {
      printf ("Com problema");
      return;
   }

   for (Ax_01_Index = 0; Ax_01_Index < 40; Ax_01_Index++)
   {
      Ax_02_Detector_Inicial = P010_Num_Randomicos (2);
      Ax_03_Detector_Final = P010_Num_Randomicos (2);
      Ax_04_Quantidade_Veiculos =
         Ax_02_Detector_Inicial - Ax_03_Detector_Final;
      printf ("Inicial=%d Final=%d Quantidade=%d\n",
         Ax_02_Detector_Inicial, Ax_03_Detector_Final,
         Ax_04_Quantidade_Veiculos);

      sprintf
         (G_02_Registro, "%02d% ; %d ; %d ; %-d ;\n",
         Ax_01_Index + 1, Ax_02_Detector_Inicial,
         Ax_03_Detector_Final,
         Ax_04_Quantidade_Veiculos);
      fputs (G_02_Registro, G_01_Arquivo);
   }

      fclose(G_01_Arquivo);
   system("PAUSE");
   return 0;
}
```

Anexo 02 – Tabela com o Resultado da Execução do 1º Simulador da Quantidade de Veículos da Via

d_e = detector de entrada

d_s= detector de saída

q_v = quantidade de veículos

Simulação	d_e	d_s	q_v	Situação
01	1	1	0	IV
02	0	0	0	I
03	1	0	1	III
04	0	0	0	I
05	0	0	0	I
06	1	1	0	IV
07	1	1	0	IV
08	1	1	0	IV
09	1	0	1	III
10	1	0	1	III
11	1	0	1	III
12	0	1	-1	II
13	0	0	0	I
14	1	0	1	III
15	0	1	-1	II
16	1	0	1	III
17	1	0	1	III
18	1	0	1	III
19	1	1	0	IV
20	1	0	1	III
21	1	1	0	IV
22	0	1	-1	II

Simulação	d_e	d_s	q_v	Situação
23	1	0	1	III
24	1	1	0	IV
25	1	0	1	III
26	1	0	1	III
27	0	1	-1	II
28	1	1	0	IV
29	1	1	0	IV
30	1	0	1	III
31	0	1	-1	II
32	0	0	0	I
33	0	0	0	I
34	0	0	0	I
35	0	0	0	I
36	0	1	-1	II
37	0	1	-1	II
38	0	0	0	I
39	0	1	-1	II
40	1	0	1	III

Anexo 03 – 2º Simulador da Quantidade de Veículos da Via

```
#include <stdio.h>
#include <stdlib.h>

FILE *G_01_Arquivo;
char G_02_Registro [80] = "\0";
char *G_03_Endereco_Arquivo = "simulador_002.txt";

int P010_Num_Randomicos (int P01_Intervalo)
{
   return rand()% P01_Intervalo;
}

int main(int argc, char *argv[])
{
   int Ax_01_Index;
   int Ax_02_Detector_Inicial;
   int Ax_03_Detector_Final;
   int Ax_04_Quantidade_Veiculos_Anterior;
   int Ax_05_Quantidade_Veiculos_Atual;


   system("CLS");

   G_01_Arquivo = fopen (G_03_Endereco_Arquivo, "w");
   if (!G_01_Arquivo)
   {
      printf ("Com problema");
      return;
   }

   for (Ax_01_Index = 0; Ax_01_Index < 80; Ax_01_Index++)
   {
      Ax_02_Detector_Inicial = P010_Num_Randomicos (2);
      Ax_03_Detector_Final = P010_Num_Randomicos (2);

      Ax_05_Quantidade_Veiculos_Atual =
         Ax_04_Quantidade_Veiculos_Anterior +
         Ax_02_Detector_Inicial - Ax_03_Detector_Final;

      printf
       ("Inicial=%d Final=%d Veiculos (t-1)=%d Veiculos (t)=%d\n",
         Ax_02_Detector_Inicial, Ax_03_Detector_Final,
         Ax_04_Quantidade_Veiculos_Anterior,
         Ax_05_Quantidade_Veiculos_Atual);

      sprintf
      (G_02_Registro, "%02d% ; %d ; %d ; %d ; %d\n",
      Ax_01_Index + 1, Ax_02_Detector_Inicial, Ax_03_Detector_Final,
      Ax_04_Quantidade_Veiculos_Anterior,
      Ax_05_Quantidade_Veiculos_Atual);

      fputs (G_02_Registro, G_01_Arquivo);

      Ax_04_Quantidade_Veiculos_Anterior =
         Ax_05_Quantidade_Veiculos_Atual;
```

```
   }

      fclose(G_01_Arquivo);

   system("PAUSE");
   return 0;
}
```

Anexo 04 – Tabela com o Resultado da Execução do 2º Simulador da Quantidade de Veículos da Via

d_e = detector de entrada

d_s = detector de saída

q_v = quantidade de veículos atual

q_a = quantidade de veículos anterior

Simulação	d_e	d_s	q_a	q_v
01	1	1	0	0
02	0	0	0	0
03	1	0	0	1
04	0	0	1	1
05	0	0	1	1
06	1	1	1	1
07	1	1	1	1
08	1	1	1	1
09	1	0	1	2
10	1	0	2	3
11	1	0	3	4
12	0	1	4	3
13	0	0	3	3
14	1	0	3	4
15	0	1	4	3
16	1	0	3	4
17	1	0	4	5
18	1	0	5	6
19	1	1	6	6
20	1	0	6	7
21	1	1	7	7
22	0	1	7	6

Simulação	d_e	d_s	q_a	q_v
23	1	0	6	7
24	1	1	7	7
25	1	0	7	8
26	1	0	8	9
27	0	1	9	8
28	1	1	8	8
29	1	1	8	8
30	1	0	8	9
31	0	1	9	8
32	0	0	8	8
33	0	0	8	8
34	0	0	8	8
35	0	0	8	8
36	0	1	8	7
37	0	1	7	6
38	0	0	6	6
39	0	1	6	5
40	1	0	5	6
41	1	1	6	6
42	0	0	6	6
43	0	0	6	6
44	0	0	6	6
45	1	0	6	7
46	0	1	7	6
47	0	1	6	5
48	1	0	5	6
49	0	0	6	6
50	1	1	6	6
51	1	1	6	6
52	1	0	6	7

Simulação	d_e	d_s	q_a	q_v
53	0	0	7	7
54	1	0	7	8
55	1	0	8	9
56	1	1	9	9
57	0	0	9	9
58	0	1	9	8
59	1	1	8	8
60	1	0	8	9
61	0	0	9	9
62	1	0	9	10
63	1	1	10	10
64	1	0	10	11
65	1	0	11	12
66	0	0	12	12
67	1	0	12	13
68	0	0	13	13
69	1	1	13	13
70	1	1	13	13
71	1	1	13	13
72	1	1	13	13
73	1	1	13	13
74	1	0	13	14
75	1	0	14	15
76	0	0	15	15
77	0	0	15	15
78	1	0	15	16
79	0	1	16	15
80	0	1	15	14

Anexo 05 – Tabela com o Resultado da Execução do 2º Simulador da Quantidade de Veículos da Via

d_e = detector de entrada

d_s = detector de saída

q_v = quantidade de veículos atual

q_a = quantidade de veículos anterior

Simulação	d_i	d_f	Q_a	Q_v	Situação
01	1	1	0	0	Igual
02	0	0	0	0	Igual
03	1	0	0	1	Maior
04	0	0	1	1	Igual
05	0	0	1	1	Igual
06	1	1	1	1	Igual
07	1	1	1	1	Igual
08	1	1	1	1	Igual
09	1	0	1	2	Maior
10	1	0	2	3	Maior
11	1	0	3	4	Maior
12	0	1	4	3	Menor
13	0	0	3	3	Igual
14	1	0	3	4	Maior
15	0	1	4	3	Menor
16	1	0	3	4	Maior
17	1	0	4	5	Maior
18	1	0	5	6	Maior
19	1	1	6	6	Igual
20	1	0	6	7	Maior
21	1	1	7	7	Igual
22	0	1	7	6	Menor

Simulação	d_i	d_f	Q_a	Q_v	Situação
23	1	0	6	7	Maior
24	1	1	7	7	Igual
25	1	0	7	8	Maior
26	1	0	8	9	Maior
27	0	1	9	8	Menor
28	1	1	8	8	Igual
29	1	1	8	8	Igual
30	1	0	8	9	Maior
31	0	1	9	8	Menor
32	0	0	8	8	Igual
33	0	0	8	8	Igual
34	0	0	8	8	Igual
35	0	0	8	8	Igual
36	0	1	8	7	Menor
37	0	1	7	6	Menor
38	0	0	6	6	Igual
39	0	1	6	5	Menor
40	1	0	5	6	Maior
41	1	1	6	6	Igual
42	0	0	6	6	Igual
43	0	0	6	6	Igual
44	0	0	6	6	Igual
45	1	0	6	7	Maior
46	0	1	7	6	Menor
47	0	1	6	5	Menor
48	1	0	5	6	Maior
49	0	0	6	6	Igual
50	1	1	6	6	Igual
51	1	1	6	6	Igual
52	1	0	6	7	Maior

Simulação	d_i	d_f	Q_a	Q_v	Situação
53	0	0	7	7	Igual
54	1	0	7	8	Maior
55	1	0	8	9	Maior
56	1	1	9	9	Igual
57	0	0	9	9	Igual
58	0	1	9	8	Menor
59	1	1	8	8	Igual
60	1	0	8	9	Maior
61	0	0	9	9	Igual
62	1	0	9	10	Maior
63	1	1	10	10	Igual
64	1	0	10	11	Maior
65	1	0	11	12	Maior
66	0	0	12	12	Igual
67	1	0	12	13	Maior
68	0	0	13	13	Igual
69	1	1	13	13	Igual
70	1	1	13	13	Igual
71	1	1	13	13	Igual
72	1	1	13	13	Igual
73	1	1	13	13	Igual
74	1	0	13	14	Maior
75	1	0	14	15	Maior
76	0	0	15	15	Igual
77	0	0	15	15	Igual
78	1	0	15	16	Maior
79	0	1	16	15	Menor
80	0	1	15	14	Menor

Anexo 06 – 3º Simulador da Quantidade de Veículos da Via e Fluxo da Via

```
#include <stdio.h>
#include <stdlib.h>

FILE *G_01_Arquivo;
char G_02_Registro [80] = "\0";
char *G_03_Endereco_Arquivo = "simulador_003.txt";

int P010_Num_Randomicos (int P01_Intervalo)
{
   return rand()% P01_Intervalo;
}

int main(int argc, char *argv[])
{
   int Ax_01_Index;
   int Ax_02_Detector_Inicial = 0;
   int Ax_03_Detector_Final = 0;
   int Ax_04_Quantidade_Veiculos_Anterior = 0;
   int Ax_05_Quantidade_Veiculos_Atual = 0;
   int Ax_06_Variacao_Veiculos = 0;
   char Ax_07_Variacao [5]="\0";
   char Ax_08_Situacao [6]="\0";

   system("CLS");

   G_01_Arquivo = fopen (G_03_Endereco_Arquivo, "w");
   if (!G_01_Arquivo)
   {
      printf ("Com problema");
      return;
   }

   for (Ax_01_Index = 0; Ax_01_Index < 160; Ax_01_Index++)
   {
      Ax_02_Detector_Inicial = P010_Num_Randomicos (2);
      Ax_03_Detector_Final = P010_Num_Randomicos (2);

      Ax_05_Quantidade_Veiculos_Atual =
         Ax_04_Quantidade_Veiculos_Anterior +
         Ax_02_Detector_Inicial - Ax_03_Detector_Final;

      Ax_06_Variacao_Veiculos = Ax_05_Quantidade_Veiculos_Atual -
         Ax_04_Quantidade_Veiculos_Anterior;

      if (Ax_06_Variacao_Veiculos == 0)
      {
         strcpy (Ax_07_Variacao, "IGUAL");
         if (Ax_02_Detector_Inicial == 1)
         {
               strcpy (Ax_08_Situacao, "NORMAL");
         }
         else
         {
               strcpy (Ax_08_Situacao, "PARADO");         }
      }
```

```
      else
         if (Ax_06_Variacao_Veiculos > 0)
         {
               strcpy (Ax_07_Variacao, "MAIOR");
               strcpy (Ax_08_Situacao, "LENTO");
         }
         else
         {
            strcpy (Ax_07_Variacao, "MENOR");
            strcpy (Ax_08_Situacao, "RAPIDO");
         }

      printf ("Index=%04d Di=%d Df=%d Qa=%03d Qv=%03d SQ=%s SIT=%s\n",
      Ax_01_Index + 1, Ax_02_Detector_Inicial, Ax_03_Detector_Final,
      Ax_04_Quantidade_Veiculos_Anterior,
      Ax_05_Quantidade_Veiculos_Atual,
      Ax_07_Variacao, Ax_08_Situacao);

      sprintf
       (G_02_Registro, "%04d% ; %d ; %d ; %03d ; %03d; %s; %s\n",
       Ax_01_Index + 1, Ax_02_Detector_Inicial, Ax_03_Detector_Final,
       Ax_04_Quantidade_Veiculos_Anterior,
       Ax_05_Quantidade_Veiculos_Atual,
       Ax_07_Variacao, Ax_08_Situacao);

      fputs (G_02_Registro, G_01_Arquivo);

      Ax_04_Quantidade_Veiculos_Anterior =
       Ax_05_Quantidade_Veiculos_Atual;
   }

   fclose(G_01_Arquivo);

   system("PAUSE");
   return 0;
}
```

Anexo 07 – Tabela com o Resultado da Execução do 3º Simulador da Quantidade de Veículos da Via e Fluxo da Via

d_e = detector de entrada

d_s = detector de saída

q_v = quantidade de veículos atual

q_a = quantidade de veículos anterior

ΔQ = variação da quantidade de veículos da via

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0001	1	1	000	000	IGUAL	NORMAL
0002	0	0	000	000	IGUAL	PARADO
0003	1	0	000	001	MAIOR	LENTO
0004	0	0	001	001	IGUAL	PARADO
0005	0	0	001	001	IGUAL	PARADO
0006	1	1	001	001	IGUAL	NORMAL
0007	1	1	001	001	IGUAL	NORMAL
0008	1	1	001	001	IGUAL	NORMAL
0009	1	0	001	002	MAIOR	LENTO
0010	1	0	002	003	MAIOR	LENTO
0011	1	0	003	004	MAIOR	LENTO
0012	0	1	004	003	MENOR	RAPIDO
0013	0	0	003	003	IGUAL	PARADO
0014	1	0	003	004	MAIOR	LENTO
0015	0	1	004	003	MENOR	RAPIDO
0016	1	0	003	004	MAIOR	LENTO
0017	1	0	004	005	MAIOR	LENTO
0018	1	0	005	006	MAIOR	LENTO
0019	1	1	006	006	IGUAL	NORMAL
0020	1	0	006	007	MAIOR	LENTO
0021	1	1	007	007	IGUAL	NORMAL

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0022	0	1	007	006	MENOR	RAPIDO
0023	1	0	006	007	MAIOR	LENTO
0024	1	1	007	007	IGUAL	NORMAL
0025	1	0	007	008	MAIOR	LENTO
0026	1	0	008	009	MAIOR	LENTO
0027	0	1	009	008	MENOR	RAPIDO
0028	1	1	008	008	IGUAL	NORMAL
0029	1	1	008	008	IGUAL	NORMAL
0030	1	0	008	009	MAIOR	LENTO
0031	0	1	009	008	MENOR	RAPIDO
0032	0	0	008	008	IGUAL	PARADO
0033	0	0	008	008	IGUAL	PARADO
0034	0	0	008	008	IGUAL	PARADO
0035	0	0	008	008	IGUAL	PARADO
0036	0	1	008	007	MENOR	RAPIDO
0037	0	1	007	006	MENOR	RAPIDO
0038	0	0	006	006	IGUAL	PARADO
0039	0	1	006	005	MENOR	RAPIDO
0040	1	0	005	006	MAIOR	LENTO
0041	1	1	006	006	IGUAL	NORMAL
0042	0	0	006	006	IGUAL	PARADO
0043	0	0	006	006	IGUAL	PARADO
0044	0	0	006	006	IGUAL	PARADO
0045	1	0	006	007	MAIOR	LENTO
0046	0	1	007	006	MENOR	RAPIDO
0047	0	1	006	005	MENOR	RAPIDO
0048	1	0	005	006	MAIOR	LENTO
0049	0	0	006	006	IGUAL	PARADO
0050	1	1	006	006	IGUAL	NORMAL
0051	1	1	006	006	IGUAL	NORMAL

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0052	1	0	006	007	MAIOR	LENTO
0053	0	0	007	007	IGUAL	PARADO
0054	1	0	007	008	MAIOR	LENTO
0055	1	0	008	009	MAIOR	LENTO
0056	1	1	009	009	IGUAL	NORMAL
0057	0	0	009	009	IGUAL	PARADO
0058	0	1	009	008	MENOR	RAPIDO
0059	1	1	008	008	IGUAL	NORMAL
0060	1	0	008	009	MAIOR	LENTO
0061	0	0	009	009	IGUAL	PARADO
0062	1	0	009	010	MAIOR	LENTO
0063	1	1	010	010	IGUAL	NORMAL
0064	1	0	010	011	MAIOR	LENTO
0065	1	0	011	012	MAIOR	LENTO
0066	0	0	012	012	IGUAL	PARADO
0067	1	0	012	013	MAIOR	LENTO
0068	0	0	013	013	IGUAL	PARADO
0069	1	1	013	013	IGUAL	NORMAL
0070	1	1	013	013	IGUAL	NORMAL
0071	1	1	013	013	IGUAL	NORMAL
0072	1	1	013	013	IGUAL	NORMAL
0073	1	1	013	013	IGUAL	NORMAL
0074	1	0	013	014	MAIOR	LENTO
0075	1	0	014	015	MAIOR	LENTO
0076	0	0	015	015	IGUAL	PARADO
0077	0	0	015	015	IGUAL	PARADO
0078	1	0	015	016	MAIOR	LENTO
0079	0	1	016	015	MENOR	RAPIDO
0080	0	1	015	014	MENOR	RAPIDO
0081	0	1	014	013	MENOR	RAPIDO

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0082	0	1	013	012	MENOR	RAPIDO
0083	0	1	012	011	MENOR	RAPIDO
0084	1	1	011	011	IGUAL	NORMAL
0085	0	0	011	011	IGUAL	PARADO
0086	1	0	011	012	MAIOR	LENTO
0087	0	0	012	012	IGUAL	PARADO
0088	0	1	012	011	MENOR	RAPIDO
0089	0	1	011	010	MENOR	RAPIDO
0090	0	0	010	010	IGUAL	PARADO
0091	1	0	010	011	MAIOR	LENTO
0092	1	1	011	011	IGUAL	NORMAL
0093	0	0	011	011	IGUAL	PARADO
0094	0	0	011	011	IGUAL	PARADO
0095	1	1	011	011	IGUAL	NORMAL
0096	0	1	011	010	MENOR	RAPIDO
0097	0	1	010	009	MENOR	RAPIDO
0098	1	1	009	009	IGUAL	NORMAL
0099	0	1	009	008	MENOR	RAPIDO
0100	1	0	008	009	MAIOR	LENTO
0101	1	0	009	010	MAIOR	LENTO
0102	1	1	010	010	IGUAL	NORMAL
0103	0	1	010	009	MENOR	RAPIDO
0104	1	0	009	010	MAIOR	LENTO
0105	0	1	010	009	MENOR	RAPIDO
0106	0	0	009	009	IGUAL	PARADO
0107	0	1	009	008	MENOR	RAPIDO
0108	1	0	008	009	MAIOR	LENTO
0109	1	1	009	009	IGUAL	NORMAL
0110	1	1	009	009	IGUAL	NORMAL
0111	1	1	009	009	IGUAL	NORMAL

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0112	0	1	009	008	MENOR	RAPIDO
0113	0	0	008	008	IGUAL	PARADO
0114	0	0	008	008	IGUAL	PARADO
0115	0	0	008	008	IGUAL	PARADO
0116	0	1	008	007	MENOR	RAPIDO
0117	1	0	007	008	MAIOR	LENTO
0118	1	1	008	008	IGUAL	NORMAL
0119	0	0	008	008	IGUAL	PARADO
0120	0	0	008	008	IGUAL	PARADO
0121	0	1	008	007	MENOR	RAPIDO
0122	0	1	007	006	MENOR	RAPIDO
0123	0	1	006	005	MENOR	RAPIDO
0124	1	0	005	006	MAIOR	LENTO
0125	0	1	006	005	MENOR	RAPIDO
0126	0	0	005	005	IGUAL	PARADO
0127	0	1	005	004	MENOR	RAPIDO
0128	0	0	004	004	IGUAL	PARADO
0129	0	0	004	004	IGUAL	PARADO
0130	1	1	004	004	IGUAL	NORMAL
0131	1	0	004	005	MAIOR	LENTO
0132	0	0	005	005	IGUAL	PARADO
0133	1	0	005	006	MAIOR	LENTO
0134	0	1	006	005	MENOR	RAPIDO
0135	1	1	005	005	IGUAL	NORMAL
0136	1	0	005	006	MAIOR	LENTO
0137	0	1	006	005	MENOR	RAPIDO
0138	1	0	005	006	MAIOR	LENTO
0139	0	0	006	006	IGUAL	PARADO
0140	1	1	006	006	IGUAL	NORMAL
0141	1	0	006	007	MAIOR	LENTO

Simulação	d_i	d_f	Q	Q_v	ΔQ	Fluxo
0142	1	1	007	007	IGUAL	NORMAL
0143	1	1	007	007	IGUAL	NORMAL
0144	0	1	007	006	MENOR	RAPIDO
0145	0	1	006	005	MENOR	RAPIDO
0146	0	0	005	005	IGUAL	PARADO
0147	1	1	005	005	IGUAL	NORMAL
0148	0	0	005	005	IGUAL	PARADO
0149	1	0	005	006	MAIOR	LENTO
0150	1	1	006	006	IGUAL	NORMAL
0151	0	1	006	005	MENOR	RAPIDO
0152	0	0	005	005	IGUAL	PARADO
0153	1	1	005	005	IGUAL	NORMAL
0154	0	1	005	004	MENOR	RAPIDO
0155	1	0	004	005	MAIOR	LENTO
0156	1	0	005	006	MAIOR	LENTO
0157	0	1	006	005	MENOR	RAPIDO
0158	1	1	005	005	IGUAL	NORMAL
0159	1	0	005	006	MAIOR	LENTO
0160	1	1	006	006	IGUAL	NORMAL

Anexo 08 – Controlador de Fluxo de Veículos

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define MAXIMO 49

FILE *G_01_Arquivo;
char G_02_Registro [80] = "\0";
char *G_03_Endereco_Arquivo = "CFV-E.TXT";

FILE *G_04_Arquivo;
char G_05_Registro [80] = "\0";
char *G_06_Endereco_Arquivo = "CFV-S.TXT";

typedef struct
{
   int G_07_Sequencia;
   int G_07_Vi; // velocidade de entrada
   int G_07_Vs; // velocidade de saida
   int G_07_Vm; // velocidade media
   int G_07_Dq; // variacao da quantidade de veiculos na via
   int G_07_Q; // quantidade de veiculos na via
   int G_07_Risco; // risco
} G_07_MASSA;

G_07_MASSA *G_08_Massa;

void p010_abre_arquivo()
{
   G_01_Arquivo = fopen (G_03_Endereco_Arquivo, "r");
   if (!G_01_Arquivo)
   {
      printf ("Com problema - Abertura 01");
      return;
   }
}

void p020_fecha_arquivo()
{
   fclose(G_01_Arquivo);
}

int p030_carrega_massa ()
{
   int P020_01_Index = 0, P020_02_Index = 0;

   p010_abre_arquivo();

//    while (!feof(G_01_Arquivo))
//    {
//      fgets (G_02_Registro, 80, G_01_Arquivo);
//      L_06_Index++;
//   }
//   L_06_Index--;
```

```
   G_08_Massa = (G_07_MASSA *) calloc (MAXIMO, sizeof(G_07_MASSA));

      while (!feof(G_01_Arquivo))
      {
      fgets (G_02_Registro, 80, G_01_Arquivo);
//      puts(G_02_Registro);
      P020_01_Index = atoi(strtok(G_02_Registro,";"));
      P020_02_Index = P020_01_Index;
      G_08_Massa[P020_02_Index].G_07_Sequencia = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Vi = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Vs = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Vm = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Dq = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Q = P020_01_Index;
      P020_01_Index = atoi(strtok(NULL,";"));
      G_08_Massa[P020_02_Index].G_07_Risco = P020_01_Index;
   }

   p020_fecha_arquivo();

   return(MAXIMO);
}

void p040_Inicializar_Numero_Aleatorio ()
{
   srand((unsigned) time(NULL));
}

int p050_Numero_Aleatorio (int P050_Numero)
{
   return rand() % P050_Numero;
}

int p060_Qual_Situacao(int P060_Qtd_Massa)
{
   int P060_01_Index = 0, P060_02_Index = 0;

   P060_01_Index = p050_Numero_Aleatorio(P060_Qtd_Massa - 1);
   printf ("Aleatorio=%d ", P060_01_Index);
   P060_02_Index = G_08_Massa[P060_01_Index].G_07_Risco;
   printf ("Risco=%d ", P060_02_Index);
   return (P060_02_Index);
}

int p070_Maior (int P070_01_Valor, int P070_02_Valor)
{
   return ((P070_01_Valor > P070_02_Valor) ? P070_01_Valor :
P070_02_Valor);
}

void p080_abre_arquivo()
{
   G_04_Arquivo = fopen (G_06_Endereco_Arquivo, "w");
   if (!G_04_Arquivo)
   {
```

```
      printf ("Com problema - Abertura 02");
      return;
   }
}

void p090_fecha_arquivo()
{
   fclose(G_04_Arquivo);
}

void p100_Grava(int P100_Index)
{
   int P100_01_Index = 0, P100_02_Pista = 0, P100_03_Pista = 0,
P100_04_Resultado = 0;

   p080_abre_arquivo();

   for (P100_01_Index = 0; P100_01_Index < P100_Index;
P100_01_Index++)
   {
      P100_02_Pista = p060_Qual_Situacao(MAXIMO);
      P100_03_Pista = p060_Qual_Situacao(MAXIMO);
      P100_04_Resultado = p070_Maior (P100_02_Pista, P100_03_Pista);
      printf ("Risco = %d\n", P100_04_Resultado);

      sprintf
       (G_02_Registro, "%03d% ; %d ;\n", P100_01_Index + 1,
P100_04_Resultado);

      fputs (G_02_Registro, G_04_Arquivo);
   }

   p090_fecha_arquivo();
}

//*****************************************************************
**********
// programa principal
//*****************************************************************
**********
int main(int argc, char *argv[])
{
   int L_01_Index = 0, L_02_Index = 0, L_03_Index, L_04_Index;
   int L_05_Pista_1 = 0, L_06_Pista_2 = 0;
   int L_06_Index = 0, L_07_Index;

   L_03_Index = p030_carrega_massa();

   p040_Inicializar_Numero_Aleatorio ();

   printf ("Quantas simulacoes ? ");
   scanf ("%d", &L_01_Index);

   p100_Grava(L_01_Index);

   system("PAUSE");

   return 0;
}
```

Anexo 09 – Controlador de Semáforos da Intersecção

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define MULTIPLO 5
#define ESCOLHIDO 2

FILE *G_01_Arquivo;
char G_02_Registro [80] = "\0";
char *G_03_Endereco_Arquivo = "CSI-TMPE.TXT";
char *G_04_Endereco_Arquivo = "CSI-ORD.TXT";
char *G_05_Endereco_Arquivo = "CSI-ACAO.TXT";

int *G_06_Tempo_Verde;
int *G_06_Tempo_Amarelo;
int *G_06_Tempo_Vermelho;

int *G_07_Ordem;
//int *G_07_Sinal_2;

//time_t G_08_Inicio;
//time_t G_09_Fim;
//time_t G_10_Inicio;
//time_t G_11_Fim;

void p010_Recupera_Guarda_Tempo()
{
   int P010_01_Aux = 0, P010_02_Index = 0;

   G_06_Tempo_Verde = (int *) calloc (24, sizeof(int));
   G_06_Tempo_Amarelo = (int *) calloc (24, sizeof(int));
   G_06_Tempo_Vermelho = (int *) calloc (24, sizeof(int));

   G_01_Arquivo = fopen (G_03_Endereco_Arquivo, "r");
   if (!G_01_Arquivo)
   {
      printf ("Com problema - Abertura 01");
      return;
   }

   while (!feof(G_01_Arquivo))
   {
      fgets(G_02_Registro, 80, G_01_Arquivo);
//      puts(G_02_Registro);
      P010_01_Aux = atoi(strtok(G_02_Registro,";"));
      P010_02_Index = P010_01_Aux;
      P010_01_Aux = atoi(strtok(NULL,";"));
      G_06_Tempo_Verde[P010_02_Index] = P010_01_Aux;
      P010_01_Aux = atoi(strtok(NULL,";"));
      G_06_Tempo_Amarelo[P010_02_Index] = P010_01_Aux;
      P010_01_Aux = atoi(strtok(NULL,";"));
      G_06_Tempo_Vermelho[P010_02_Index] = P010_01_Aux;
   }
   fclose(G_01_Arquivo);
}
```

```
void p020_Inicializar_Numero_Aleatorio ()
{
   srand((unsigned) time(NULL));
}

int p030_Numero_Aleatorio (int P040_Numero)
{
   return rand() % P040_Numero;
}

void p040_Recupera_Ordem(int P040_Ordem)
{
   int P040_01_Aux = 0, P040_02_Index = 0;

   G_07_Ordem = (int *) calloc (P040_Ordem, sizeof(int));

   G_01_Arquivo = fopen (G_04_Endereco_Arquivo, "r");
   if (!G_01_Arquivo)
   {
      printf ("Com problema - Abertura 01");
      return;
   }

   while (!feof(G_01_Arquivo))
   {
      fgets(G_02_Registro, 80, G_01_Arquivo);
//      puts(G_02_Registro);
      P040_01_Aux = atoi(strtok(G_02_Registro,";"));
      P040_02_Index = P040_01_Aux;
      if (P040_02_Index > P040_Ordem)
         break;
      P040_01_Aux = atoi(strtok(NULL,";"));
      G_07_Ordem[P040_02_Index] = P040_01_Aux;
   }
   fclose(G_01_Arquivo);
}

int p090_Mudar_Sinal (int P090_Ordem_1, int P090_Ordem_2)
{
   static P090_01_Sinal_1 = 0;
   static P090_02_Sinal_2 = 0;
   static P090_03_Proximo_1 = 0;
   static P090_04_Proximo_2 = 0;
   static P090_05_Antes_1 = 0;
   static P090_06_Antes_2 = 0;
   int P090_07_Qual_Tempo = 0;
   int P090_08_Tempo_Recuperado = 0;
   time_t P090_09_Inicio;
   time_t P090_10_Fim;
   int P090_11_Diferenca = 0;
   int P090_12_Fez_Recursividade = 0;

   // achar qual o proximo ciclo
   if (P090_01_Sinal_1 == 0 && P090_01_Sinal_1 == 0) // intermitente
   {
      P090_03_Proximo_1 = 3;
      P090_04_Proximo_2 = 3;
   }
   else
```

```
   {
      P090_03_Proximo_1 = P090_01_Sinal_1 + 1;
      P090_04_Proximo_2 = P090_02_Sinal_2 + 1;
      if (P090_03_Proximo_1 > 3)
      {
         if (P090_03_Proximo_1 > 3)
         {
            if (P090_05_Antes_1 == 2)
               P090_03_Proximo_1 = 3;
            else
            {
               if (P090_02_Sinal_2 == 1 || P090_02_Sinal_2 == 2)
                  P090_03_Proximo_1 = 3;
               else
                  P090_03_Proximo_1 = 1;
            }
         }
      }

      if (P090_04_Proximo_2 > 3)
      {
         if (P090_04_Proximo_2 > 3)
         {
            if (P090_06_Antes_2 == 2)
               P090_04_Proximo_2 = 3;
            else
            {
               if (P090_01_Sinal_1 == 1 || P090_01_Sinal_1 == 2)
                  P090_04_Proximo_2 = 3;
               else
               {
                  if (P090_05_Antes_1 == 0)
                     P090_04_Proximo_2 = 3;
                  else
                     P090_04_Proximo_2 = 1;
               }
            }
         }
      }
   }

   if (P090_Ordem_1 == 0 && P090_Ordem_2 == 0) // nao eh uma ordem
   {
      // faltando o tratamento para mudar ao horario intermitente

      P090_Ordem_1 = P090_03_Proximo_1;
      P090_Ordem_2 = P090_04_Proximo_2;
   }

   printf ("Mudar S1=%d S2=%d P1=%d P2=%d\n",
   P090_01_Sinal_1, P090_02_Sinal_2, P090_03_Proximo_1,
      P090_04_Proximo_2);
//   system ("PAUSE");

   // mudar o valor do sinal
   P090_05_Antes_1 = P090_01_Sinal_1;
   P090_06_Antes_2 = P090_02_Sinal_2;
   P090_01_Sinal_1 = P090_03_Proximo_1;
   P090_02_Sinal_2 = P090_04_Proximo_2;
```

```
   switch(P090_01_Sinal_1)
   {
      case 1 : printf("Sinal 1=%d Verde        ", P090_01_Sinal_1);
      break;
      case 2 : printf("Sinal 1=%d Amarelo      ", P090_01_Sinal_1);
       break;
      case 3 : printf("Sinal 1=%d Vermelho     ", P090_01_Sinal_1);
      break;
      default: printf("Sinal 1=%d Intermitente ", P090_01_Sinal_1);
       break;
   }

   switch(P090_02_Sinal_2)
   {
      case 1 : printf("Sinal 2=%d Verde       \n", P090_02_Sinal_2);
      break;
      case 2 : printf("Sinal 2=%d Amarelo     \n", P090_02_Sinal_2);
      break;
      case 3 : printf("Sinal 2=%d Vermelho    \n", P090_02_Sinal_2);
      break;
      default: printf("Sinal 2=%d Intermitente\n", P090_02_Sinal_2);
      break;
   }

   if (P090_01_Sinal_1 != P090_Ordem_1 || P090_02_Sinal_2 !=
P090_Ordem_2)
   {
      printf ("Recursividade\n");
//      system ("PAUSE");
      P090_09_Inicio = time(NULL);
      while (P090_11_Diferenca <= 5) // cinco segundos
      {
         P090_10_Fim = time(NULL);
         P090_11_Diferenca = difftime(P090_10_Fim, P090_09_Inicio);
      }

      p090_Mudar_Sinal (P090_Ordem_1, P090_Ordem_2);
      P090_12_Fez_Recursividade = 1;
   }

   if (P090_12_Fez_Recursividade == 0)
   {
      P090_07_Qual_Tempo = p030_Numero_Aleatorio (24);
      P090_07_Qual_Tempo++;
      if (P090_01_Sinal_1 == 1)
         P090_08_Tempo_Recuperado =
         G_06_Tempo_Verde[P090_07_Qual_Tempo];
      else
         if (P090_01_Sinal_1 == 2)
            P090_08_Tempo_Recuperado =
            G_06_Tempo_Amarelo[P090_07_Qual_Tempo];
         else
            P090_08_Tempo_Recuperado =
            G_06_Tempo_Vermelho[P090_07_Qual_Tempo];

      printf ("Tempo Sinal 1(%d)=%d\n\n", P090_07_Qual_Tempo,
         P090_08_Tempo_Recuperado);
   //   system("PAUSE");

      return(P090_08_Tempo_Recuperado);
```

```
   }
   else
      return(0);
}

int p100_Processa(int P100_Qtd_Ordem, int P100_Qtd_Ciclo)
{
   time_t P100_01_Inicio;
   time_t P100_02_Fim;
   int P100_03_Ha_Ordem = 0;
   int P100_04_Resultado = 0;
   int P100_05_Qual_Ordem = 0;
   int P100_06_Ordem_1 = 0;
   int P100_07_Ordem_2 = 0;
   int P100_08_Diferenca = 0;
   int P100_09_Tempo_Ciclo = 0;
   int P100_10_Qtd_Ordem = 0;
   int P100_11_Qtd_Ciclo = 0;

   p020_Inicializar_Numero_Aleatorio();

   while (1==1)
   {
      // recuperar um numero aleatorio para identificar qual a ordem
      P100_05_Qual_Ordem = p030_Numero_Aleatorio (P100_Qtd_Ordem);
      if (G_07_Ordem[P100_05_Qual_Ordem] == 0)
      {
         P100_06_Ordem_1 = 0;
         P100_07_Ordem_2 = 0;
      }
      else
      {
         if (G_07_Ordem[P100_05_Qual_Ordem] == 1)
         // sinal1=verde e sinal2=vermelho
         {
            P100_06_Ordem_1 = 1; // verde = 1
            P100_07_Ordem_2 = 3; // vermelho = 3
         }
         else
         {
            if (G_07_Ordem[P100_05_Qual_Ordem] == 2)
            // sinal1=vermelho e sinal2=verde
            {
               P100_06_Ordem_1 = 3; // vermelho = 3
               P100_07_Ordem_2 = 1; // verde = 1
            }
            else
            {
               if (G_07_Ordem[P100_05_Qual_Ordem] == 3)
                  // sinal1=vermelho e sinal2=vermelho
                  P100_06_Ordem_1 = P100_07_Ordem_2 = 3;
                  // vermelho = 3
            }
         }
         P100_01_Inicio = time(NULL);
         P100_09_Tempo_Ciclo = -1;
      }

      P100_02_Fim = time(NULL);
      P100_08_Diferenca = difftime(P100_02_Fim, P100_01_Inicio);
```

```
      if (P100_08_Diferenca > P100_09_Tempo_Ciclo)
      {
         P100_11_Qtd_Ciclo++;
         if (P100_06_Ordem_1 != 0 && P100_07_Ordem_2 != 0)
         {
            P100_10_Qtd_Ordem++;
            printf ("Ordem Nr.=%d Ordem1=%d Ordem2=%d\n",
            P100_10_Qtd_Ordem, P100_06_Ordem_1, P100_07_Ordem_2);
         }
         else
            printf ("Ciclo Nr.=%d Ordem1=%d Ordem2=%d\n",
               P100_11_Qtd_Ciclo, P100_06_Ordem_1, P100_07_Ordem_2);
//       system ("PAUSE");

         P100_09_Tempo_Ciclo = p090_Mudar_Sinal (P100_06_Ordem_1,
            P100_07_Ordem_2);

         P100_01_Inicio = time(NULL);
      }

//      if (P100_10_Qtd_Ordem >= P100_Qtd_Ordem ||
            P100_11_Qtd_Ciclo >= P100_Qtd_Ciclo)
      if (P100_11_Qtd_Ciclo >= P100_Qtd_Ciclo)
         break;
   }
}

//*********************************************************************
// programa principal
//*********************************************************************
int main(int argc, char *argv[])
{
   int L_01_Qtd_Ordem = 0;
   int L_02_Qtd_Ciclo = 0;

   p010_Recupera_Guarda_Tempo();

   printf ("Informe a quantidade de ordens a processar ? ");
   scanf ("%d", &L_01_Qtd_Ordem);

   p040_Recupera_Ordem(L_01_Qtd_Ordem);

//   printf ("Informe a quantidade de ciclos a processar ? ");
//   scanf ("%d", &L_02_Qtd_Ciclo);

   L_02_Qtd_Ciclo = L_01_Qtd_Ordem;
   p100_Processa(L_01_Qtd_Ordem, L_02_Qtd_Ciclo);

   system("PAUSE");

   return 0;
}
```

Anexo 10 – Parâmetros da RNL

Situação	Fluxo		Estado Atual	Ordem Prevista	Ordem Real
	Via Principal	Via Secundária			
I	Nenhum	Nenhum	Ambos	Ambos	Nenhuma
II	Nenhum	Nenhum	Liberado	Ambos	Ambos
III	Nenhum	Nenhum	Bloqueado	Ambos	Ambos
IV	Nenhum	Baixo	Ambos	Bloqueado	Bloqueado
V	Nenhum	Baixo	Liberado	Bloqueado	Bloqueado
VI	Nenhum	Baixo	Bloqueado	Bloqueado	Nenhuma
VII	Nenhum	Médio	Ambos	Bloqueado	Bloqueado
VIII	Nenhum	Médio	Liberado	Bloqueado	Bloqueado
IX	Nenhum	Médio	Bloqueado	Bloqueado	Nenhuma
X	Nenhum	Alto	Ambos	Bloqueado	Bloqueado
XI	Nenhum	Alto	Liberado	Bloqueado	Bloqueado
XII	Nenhum	Alto	Bloqueado	Bloqueado	Nenhuma
XIII	Baixo	Nenhum	Ambos	Liberado	Liberado
XIV	Baixo	Nenhum	Liberado	Liberado	Nenhuma
XV	Baixo	Nenhum	Bloqueado	Liberado	Liberado
XVI	Baixo	Baixo	Ambos	Nenhuma	Nenhuma
XVII	Baixo	Baixo	Liberado	Nenhuma	Nenhuma
XVIII	Baixo	Baixo	Bloqueado	Nenhuma	Nenhuma
XIX	Baixo	Médio	Ambos	Bloqueado	Bloqueado
XX	Baixo	Médio	Liberado	Bloqueado	Bloqueado
XXI	Baixo	Médio	Bloqueado	Bloqueado	Nenhuma
XXII	Baixo	Alto	Ambos	Bloqueado	Bloqueado
XXIII	Baixo	Alto	Liberado	Bloqueado	Bloqueado
XXIV	Baixo	Alto	Bloqueado	Bloqueado	Nenhuma
XXV	Médio	Nenhum	Ambos	Liberado	Liberado
XXVI	Médio	Nenhum	Liberado	Liberado	Nenhuma
XXVII	Médio	Nenhum	Bloqueado	Liberado	Liberado

Situação	Fluxo		Estado Atual	Ordem Prevista	Ordem Real
	Via Principal	Via Secundária			
XXVIII	Médio	Baixo	Ambos	Liberado	Liberado
XXIX	Médio	Baixo	Liberado	Liberado	Nenhuma
XXX	Médio	Baixo	Bloqueado	Liberado	Liberado
XXXI	Médio	Médio	Ambos	Nenhuma	Nenhuma
XXXII	Médio	Médio	Liberado	Nenhuma	Nenhuma
XXXIII	Médio	Médio	Bloqueado	Nenhuma	Nenhuma
XXXIV	Médio	Alto	Ambos	Bloqueado	Bloqueado
XXXV	Médio	Alto	Liberado	Bloqueado	Bloqueado
XXXVI	Médio	Alto	Bloqueado	Bloqueado	Nenhuma
XXVII	Alto	Nenhum	Ambos	Liberado	Liberado
XXVIII	Alto	Nenhum	Liberado	Liberado	Nenhuma
XXIX	Alto	Nenhum	Bloqueado	Liberado	Liberado
XL	Alto	Baixo	Ambos	Liberado	Liberado
XLI	Alto	Baixo	Liberado	Liberado	Nenhuma
XLII	Alto	Baixo	Bloqueado	Liberado	Liberado
XLIII	Alto	Médio	Ambos	Liberado	Liberado
XLIV	Alto	Médio	Liberado	Liberado	Nenhuma
XLV	Alto	Médio	Bloqueado	Liberado	Liberado
XLVI	Alto	Alto	Ambos	Nenhuma	Nenhuma
XLVII	Alto	Alto	Liberado	Nenhuma	Nenhuma
XLVIII	Alto	Alto	Bloqueado	Nenhuma	Nenhuma

Anexo 11 – Rede Neural em C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define raizquadrada(x) ((x) * (x))

int * G01_Qtd_Neuronio;
double * G01_Entrada;
double ** G01_Dado;
double *** G01_Peso_Recuperado;
double *** G01_Peso_Salvo;
double *** G01_Peso_Variacao;
double * G01_Saida_Esperada;
double * G01_Saida_Calculada;
double G01_Erro_Padrao;
double * G01_Delta;

// area para guardar os dados do arquivo
typedef struct
{
   int G02_Seq;
   int * G02_Parametro;
   int G02_Ordem;
} ENTRADA;

ENTRADA * G02_Entrada;

int G03_Qtd_Registro;
int G04_Qtd_Camadas;
int * G05_Qtd_Neuronios;
double G06_Momento;
double G07_Taxa_Aprendizado;
double G08_Erro_Alvo;
int G09_Epoca;
int G10_Funcao_Ativacao;
int G11_Saida_Errada = 0;

FILE * G99_01_Arquivo;
char G99_02_Nome_Arquivo[20]="\0";
char G99_03_Registro[80]="\0";
int G99_04_Tem_Log = 0;
time_t G99_05_Tempo;
struct tm *G99_06_Tempo;
int G99_07_Ano,
   G99_08_Mes,
   G99_09_Dia,
   G99_10_Hora,
   G99_11_Minuto,
   G99_12_Segundo;
char G99_13_Buffer[256];

//*****************************************************************
*******//
int p010Le_Entrada() // OK
```

```
{
   FILE * p010_01_Arquivo;
   char p010_02_Registro[80]="\0";
   char p010_03_Nome_Arquivo[12]="\0";
   int p010_04_Indx = 0.;
   int p010_05_Aux = 0;
   int p010_06_Indx = 0;
   // recebendo o nome do arquivo de entrada
   fflush(stdin);
   printf ("Qual o arquivo de entrada:");
   gets(p010_03_Nome_Arquivo);

   // abrindo o arquivo de entrada
   p010_01_Arquivo = fopen (p010_03_Nome_Arquivo, "r");
   if (!p010_01_Arquivo)
   {
      printf ("Erro na Abertura do Arquivo %s\n\n",
p010_03_Nome_Arquivo);
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Erro na Abertura do Arquivo
%s\n\n",
            p010_03_Nome_Arquivo);
      }
      return(9);
   }
   for (p010_04_Indx = 0; p010_04_Indx < 12; p010_04_Indx++)
   {
      p010_03_Nome_Arquivo[p010_04_Indx] =
         toupper(p010_03_Nome_Arquivo[p010_04_Indx]);
   }
   if (G99_04_Tem_Log == 1)
   {
      fprintf (G99_01_Arquivo, "Arquivo=%s\n\n",
p010_03_Nome_Arquivo);
   }

   // lendo o 1o. registro do arquivo de entrada
   fgets(p010_02_Registro, 80, p010_01_Arquivo);
   if (!feof(p010_01_Arquivo))
   {  // recuperando os parametros para montagem da rede
      G03_Qtd_Registro = atoi(strtok(p010_02_Registro,";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Registros=%d ", G03_Qtd_Registro);
      }
      G04_Qtd_Camadas = atoi(strtok(NULL,";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Camadas=%d ", G04_Qtd_Camadas);
      }
      G05_Qtd_Neuronios = (int *) calloc (G04_Qtd_Camadas,
sizeof(int));
      for (p010_04_Indx = 0; p010_04_Indx < G04_Qtd_Camadas;
p010_04_Indx++)
      {
         G05_Qtd_Neuronios[p010_04_Indx] = atoi(strtok(NULL,";"));
         if (G99_04_Tem_Log == 1)
         {
            fprintf (G99_01_Arquivo, "Neuronios[%d]=%d",
```

```
               p010_04_Indx+1, G05_Qtd_Neuronios[p010_04_Indx]);
         }
      }
      G06_Momento = atof (strtok(NULL,";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "\n\n");
         fprintf (G99_01_Arquivo, "Momento=%f ", G06_Momento);
      }
      G07_Taxa_Aprendizado = atof (strtok(NULL,";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Tx.Aprendizado=%f ",
G07_Taxa_Aprendizado);
      }
      G08_Erro_Alvo = atof (strtok(NULL, ";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Erro=%f ", G08_Erro_Alvo);
      }
      G09_Epoca = atoi (strtok(NULL, ";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Epoca=%d ", G09_Epoca);
      }
      G10_Funcao_Ativacao = atoi (strtok(NULL, ";"));
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Funcao Ativacao=%d\n\n",
G10_Funcao_Ativacao);
      }
   }
   else
   {
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo, "Erro na 1a. Leitura do Arquivo
%s\n\n",
            p010_03_Nome_Arquivo);
      }
      printf("Problema na 1a. Leitura do %s\n\n",
p010_03_Nome_Arquivo);
      return(8);
   }
   // alocando o espaco necessario para conter os registros de entrada
   G02_Entrada = (ENTRADA *) calloc (G03_Qtd_Registro,
sizeof(ENTRADA));
   for (p010_04_Indx = 0; p010_04_Indx < G03_Qtd_Registro;
p010_04_Indx++)
      G02_Entrada[p010_04_Indx].G02_Parametro = (int *)
         calloc (G05_Qtd_Neuronios[p010_04_Indx], sizeof(int));
   // Carregando os registros de entrada
   while (!feof(p010_01_Arquivo))
   {
      fgets(p010_02_Registro, 80, p010_01_Arquivo);
      p010_05_Aux = atoi(strtok(p010_02_Registro, ";"));
      p010_04_Indx = p010_05_Aux-1;
      G02_Entrada[p010_04_Indx].G02_Seq = p010_05_Aux;
      p010_05_Aux = atoi(strtok(NULL, ";"));
      G02_Entrada[p010_04_Indx].G02_Parametro[0] = p010_05_Aux;
```

```
      if (G05_Qtd_Neuronios[0] > 1)
      {
         p010_05_Aux = atoi(strtok(NULL, ";"));
         G02_Entrada[p010_04_Indx].G02_Parametro [1] = p010_05_Aux;
      }
      if (G05_Qtd_Neuronios[0] > 2)
      {
         p010_05_Aux = atoi(strtok(NULL, ";"));
         G02_Entrada[p010_04_Indx].G02_Parametro [2] = p010_05_Aux;
      }
     if (G05_Qtd_Neuronios[0] > 3)
      {
         p010_05_Aux = atoi(strtok(NULL, ";"));
         G02_Entrada[p010_04_Indx].G02_Parametro[3] = p010_05_Aux;
      }
      p010_05_Aux = atoi(strtok(NULL, ";"));
      G02_Entrada[p010_04_Indx].G02_Ordem = p010_05_Aux;
   }
   fclose(p010_01_Arquivo);
   return(0);
}

//*****************************************************************
*******//
void p020_Inicializa_Randomizacao() // OK
{
   srand((unsigned) time(NULL));
}

//*****************************************************************
*******//
double p030_Encontra_Real(float p030_Menor_Valor, float
p030_Maior_Valor)
{ // OK
   double p030_Aux = 0;
   p030_Aux = ((((double)(rand()%10))/10) - p030_Maior_Valor);
   return (p030_Aux);
}

//*****************************************************************
*******//
void p040_Cria_Rede()
{
   int p040_01_Indx = 0,
      p040_02_Indx = 0,
      p040_03_Indx = 0,
      p040_04_Aux = 0;
   // define a rede de acordo com a quantidade de camadas
   G01_Qtd_Neuronio = (int *) calloc (G04_Qtd_Camadas, sizeof(int));
   for (p040_01_Indx = 0; p040_01_Indx < G04_Qtd_Camadas;
p040_01_Indx++)
   {
      G01_Qtd_Neuronio [p040_01_Indx] = G05_Qtd_Neuronios
[p040_01_Indx];
   }
/*   G01_Entrada = (double *) calloc (G05_Qtd_Neuronios[0]+1,
sizeof(double));
   for (p040_01_Indx = 0; p040_01_Indx <= G05_Qtd_Neuronios [0];
p040_01_Indx++)
   {
```

```
      if (p040_01_Indx == 0) // bias
         G01_Entrada [p040_01_Indx] = 1.0;
      else
         G01_Entrada [p040_01_Indx] = 0.0;
   }
*/
   G01_Dado = (double **) calloc (G04_Qtd_Camadas, sizeof(double *));
   for (p040_01_Indx = 0; p040_01_Indx < G04_Qtd_Camadas;
p040_01_Indx++)
   {
      p040_04_Aux = 1;
      G01_Dado[p040_01_Indx] = (double *)
         calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
         sizeof(double));
   }
   for (p040_01_Indx = 0; p040_01_Indx < G04_Qtd_Camadas;
p040_01_Indx++)
   {
      for (p040_02_Indx = 0; p040_02_Indx <=
G05_Qtd_Neuronios[p040_01_Indx];
         p040_02_Indx++)
      {
         if  (p040_02_Indx == 0)
            G01_Dado[p040_01_Indx][p040_02_Indx] = 1.0;
         else
            G01_Dado[p040_01_Indx][p040_02_Indx] = 0.0;
      }
   }
   G01_Saida_Esperada = (double *)
      calloc (G05_Qtd_Neuronios [G04_Qtd_Camadas - 1],
sizeof(double));
   G01_Saida_Calculada = (double *)
      calloc (G05_Qtd_Neuronios [G04_Qtd_Camadas - 1],
sizeof(double));
   for (p040_01_Indx = 0; p040_01_Indx <
G05_Qtd_Neuronios[G04_Qtd_Camadas - 1];
      p040_01_Indx++)
   {
      G01_Saida_Esperada [p040_01_Indx] = 0.;
      G01_Saida_Calculada [p040_01_Indx] = 0.;
   }
   G01_Peso_Recuperado = (double ***) calloc (G04_Qtd_Camadas - 1,
      sizeof(double **));
   G01_Peso_Salvo = (double ***) calloc (G04_Qtd_Camadas - 1,
      sizeof(double **));
   G01_Peso_Variacao = (double ***) calloc (G04_Qtd_Camadas - 1,
      sizeof(double **));
   for (p040_01_Indx = 0; p040_01_Indx < G04_Qtd_Camadas - 1;
p040_01_Indx++)
   {
      p040_04_Aux = 1;
      G01_Peso_Recuperado[p040_01_Indx]= (double **)
         calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
         sizeof(double *));
      G01_Peso_Salvo[p040_01_Indx] = (double **)
         calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
         sizeof(double *));
      G01_Peso_Variacao[p040_01_Indx] = (double **)
         calloc (G05_Qtd_Neuronios[p040_01_Indx] + p040_04_Aux,
         sizeof(double *));
```

```
      for (p040_02_Indx = 0;
         p040_02_Indx <= G05_Qtd_Neuronios[p040_01_Indx];
p040_02_Indx++)
      {
         G01_Peso_Recuperado[p040_01_Indx][p040_02_Indx] =
            (double *) calloc (G05_Qtd_Neuronios[p040_01_Indx + 1],
sizeof(double));
         G01_Peso_Salvo[p040_01_Indx][p040_02_Indx] =
            (double *) calloc (G05_Qtd_Neuronios[p040_01_Indx + 1],
sizeof(double));
         G01_Peso_Variacao[p040_01_Indx][p040_02_Indx] =
            (double *) calloc (G05_Qtd_Neuronios[p040_01_Indx + 1],
sizeof(double));
      }
   }
   G01_Delta = (double *) calloc (G04_Qtd_Camadas, sizeof (double));
   for (p040_01_Indx = 0; p040_01_Indx < G04_Qtd_Camadas;
p040_01_Indx++)
   {
      G01_Delta[p040_01_Indx] = 0.;
   }
/*
   for (p040_01_Indx = 1; p040_01_Indx < G04_Qtd_Camadas;
p040_01_Indx++)
   {
      for (p040_02_Indx = 0; p040_02_Indx <=
G05_Qtd_Neuronios[p040_01_Indx]; p040_02_Indx++)
      {
         if (p040_01_Indx == (G04_Qtd_Camadas - 1) &&
            p040_02_Indx == G05_Qtd_Neuronios[p040_01_Indx])
            ;
         else
         {
            for (p040_03_Indx = 0; p040_03_Indx <=
G05_Qtd_Neuronios[p040_01_Indx -1];
               p040_03_Indx++)
            {
               printf ("G01_Peso_Recuperado[%d][%d][%d]=%f\n",
                  p040_01_Indx, p040_02_Indx, p040_03_Indx,

G01_Peso_Recuperado[p040_01_Indx][p040_02_Indx][p040_03_Indx]);
            }
         }
      }
   }
*/
}

//*****************************************************************
*******//
void p050_Recupera_Peso(char *p050_Nome_Arquivo)
{
   int p050_01_Indx, p050_02_Indx, p050_03_Indx;
   FILE * p050_04_Arquivo;
   char p050_05_Registro[80]="\0";

   p050_04_Arquivo = fopen (p050_Nome_Arquivo, "r");
   if (!p050_04_Arquivo)
   {
      p050_04_Arquivo = fopen (p050_Nome_Arquivo, "w");
```

```
   }
   // Soh deve existir ateh a camada oculta antes da camada de saida
   for (p050_01_Indx = 0; p050_01_Indx < G04_Qtd_Camadas - 1;
p050_01_Indx++)
   {
      for (p050_02_Indx = 0; p050_02_Indx <=
G01_Qtd_Neuronio[p050_01_Indx];
         p050_02_Indx++)
      {
         for (p050_03_Indx = 0;
            p050_03_Indx <= G01_Qtd_Neuronio[p050_01_Indx + 1];
p050_03_Indx++)
         {

G01_Peso_Recuperado[p050_01_Indx][p050_02_Indx][p050_03_Indx]
               = p030_Encontra_Real(-0.5, 0.5);
            G01_Peso_Salvo[p050_01_Indx][p050_02_Indx][p050_03_Indx] =
0.;

G01_Peso_Variacao[p050_01_Indx][p050_02_Indx][p050_03_Indx] = 0.;
            fprintf(p050_04_Arquivo, "Peso[%d][%d][%d]=%f\n",
               p050_01_Indx, p050_02_Indx, p050_03_Indx,

G01_Peso_Recuperado[p050_01_Indx][p050_02_Indx][p050_03_Indx]);
         }
      }
   }
   fclose(p050_04_Arquivo);
}

//*****************************************************************
*******//
int p060_Recupera_Inteiro(int p060_Menor_Valor, int p060_Maior_Valor)
{
   int p060_01_Aux =0;
   p060_01_Aux = rand() % ((p060_Maior_Valor - p060_Menor_Valor)+ 1);
   p060_01_Aux += p060_Menor_Valor;
 return (p060_01_Aux);
}

//*****************************************************************
*******//
void p070_Seleciona_Massa(char *p070_Nome_Arquivo, double
p070_Percentual)
{
   FILE * p070_01_Arquivo;
   char p070_02_Registro[80]="\0";
   int p070_04_Indx = 0;
   float p070_05_Perc = 0.;
   int * p070_06_Tab;
   int p070_07_Indx = 0;
   // criando a tabela para ter os registros selecionados;
   p070_06_Tab = (int *) calloc (G03_Qtd_Registro, sizeof(int));
   for (p070_04_Indx = 0; p070_04_Indx < G03_Qtd_Registro;
p070_04_Indx++)
      p070_06_Tab[p070_04_Indx] = 0;
   // abrindo o arquivo de treino
   p070_01_Arquivo = fopen (p070_Nome_Arquivo, "r");
   if (!p070_01_Arquivo)
   {
```

```
      p070_01_Arquivo = fopen (p070_Nome_Arquivo, "w");
      p070_05_Perc = G03_Qtd_Registro * p070_Percentual;
      p070_04_Indx = p070_05_Perc;
      while (1==1)
      {
         p070_07_Indx = p060_Recupera_Inteiro(1, G03_Qtd_Registro);
         if (p070_06_Tab[p070_07_Indx - 1] == 0)
         {
            p070_04_Indx--;
            if (p070_04_Indx <= 0)
               break;
            p070_06_Tab[p070_07_Indx - 1] = 1;
         }
      }
      for (p070_04_Indx = 0; p070_04_Indx < G03_Qtd_Registro;
p070_04_Indx++)
      {
         if (p070_06_Tab[p070_04_Indx] == 1)
            fprintf(p070_01_Arquivo, "%03d%\n", p070_04_Indx);
      }
      fclose(p070_01_Arquivo);
   }
}

//*********************************************************************
*******//
float p080_Funcao_Ativacao (float p080_Dado, int p080_Funcao)
{
   if (p080_Funcao == 1) // sigmoide
      return (1.0 / (1.0 + exp(-p080_Dado)));
   else // tangente hiperbolica
      return (0);
}

//*********************************************************************
*******//
float p090_Derivada_Funcao_Ativacao (float p090_Dado, int p090_Funcao)
{
   float p090_Resultado = 0;
   // faz calculo da sigmoide
   p090_Resultado = p080_Funcao_Ativacao (p090_Dado, p090_Funcao);
   if (p090_Funcao == 1) // sigmoide
      return (p090_Resultado * (1.0 - p090_Resultado));
   else // tangente hiperbolica
      return (0);
}

//*********************************************************************
*******//
void p100_Limpa_Dados_Entrada ()
{
   int p100_01_Indx =0,
      p100_02_Indx = 0;
   // limpa os dados de entrada
   for (p100_01_Indx = 0; p100_01_Indx < G04_Qtd_Camadas;
p100_01_Indx++)
   {
      for (p100_02_Indx = 0; p100_02_Indx <=
G05_Qtd_Neuronios[p100_01_Indx];
         p100_02_Indx++)
```

```
      {
         if (p100_02_Indx == 0)
            G01_Dado[p100_01_Indx][p100_02_Indx] = 1.0;
         else
            G01_Dado[p100_01_Indx][p100_02_Indx] = 0.0;
      }
   }
}

//*********************************************************************
*******//
void p110_Recupera_Dados_Entrada(FILE * p110_Arquivo, int p110_Indx)
{
   int p110_01_Indx = 0,
      p110_02_Indx = 0;

   p110_02_Indx = 0;
   for (p110_01_Indx = 1; p110_01_Indx <= G05_Qtd_Neuronios[0];
p110_01_Indx++)
   {
      if (p110_01_Indx == 1)
      {
         G01_Dado[p110_02_Indx][p110_01_Indx - 1] = 1.0;
//         fprintf(p110_Arquivo, "B%d=%f ", p110_01_Indx - 1,
//            G01_Dado[p110_02_Indx][p110_01_Indx - 1]);
      }
      G01_Dado[p110_02_Indx][p110_01_Indx] = ((double)
         G02_Entrada[p110_Indx - 1].G02_Parametro[p110_01_Indx - 1]);
//      fprintf(p110_Arquivo, "I%d=%f ", p110_01_Indx,
//            G01_Dado[p110_02_Indx][p110_01_Indx]);
   }
}

//*********************************************************************
*******//
void p120_Recupera_Dados_Saida (FILE * p120_Arquivo, int p120_Indx)
{
   int p120_01_Indx = 0, p120_02_Indx = 0;
   p120_02_Indx = G04_Qtd_Camadas - 1;
   for (p120_01_Indx = 1; p120_01_Indx <=
G05_Qtd_Neuronios[p120_02_Indx]; p120_01_Indx++)
   {
      G01_Saida_Esperada [p120_01_Indx] = ((double)
         G02_Entrada[p120_Indx].G02_Ordem / 3) ;
//      fprintf(p120_Arquivo, "O%d=%f %d\n", p120_01_Indx,
//         G01_Saida_Esperada[p120_01_Indx],
//         G02_Entrada[p120_Indx].G02_Ordem);
   }
}

//*********************************************************************
*******//
void p130_Backpropagation_Forward (FILE * p130_Arquivo)
{
   int p130_01_Indx = 0, p130_02_Indx = 0, p130_03_Indx = 0;
   double p130_04_Aux = 0.;

   for (p130_01_Indx = 1; p130_01_Indx < G04_Qtd_Camadas;
p130_01_Indx++)
   {
```

```
      for (p130_02_Indx = 1; p130_02_Indx <=
G01_Qtd_Neuronio[p130_01_Indx];
         p130_02_Indx++)
      {
         for(p130_03_Indx = 0; p130_03_Indx <=
G01_Qtd_Neuronio[p130_01_Indx - 1];
            p130_03_Indx++)
         {
            if (G99_04_Tem_Log == 1)
            {
               fprintf (G99_01_Arquivo,
                  "01=%d 02=%d 03=%d\n", p130_01_Indx, p130_02_Indx,
p130_03_Indx);
            }
            // calcula o somatorio do produto do neuronio pelo peso
            p130_04_Aux = G01_Dado[p130_01_Indx][p130_02_Indx];
//            fprintf (p130_Arquivo, " Ant(%d, %d)=%f ",
//               p130_01_Indx, p130_02_Indx, p130_04_Aux);
            G01_Dado[p130_01_Indx][p130_02_Indx] =
               p130_04_Aux +
               ((double) G01_Dado[p130_01_Indx - 1][p130_03_Indx] *
               G01_Peso_Recuperado[p130_01_Indx -
1][p130_03_Indx][p130_02_Indx]);
//               fprintf (p130_Arquivo, "Dado(%d,%d)=%f *
Peso(%d,%d,%d)=%f Atual(%d,%d)=%f",
//               p130_01_Indx - 1, p130_03_Indx,
//               G01_Dado[p130_01_Indx - 1][p130_03_Indx],
//               p130_01_Indx - 1, p130_03_Indx, p130_02_Indx,
//               G01_Peso_Recuperado[p130_01_Indx -
1][p130_03_Indx][p130_02_Indx],
//               p130_01_Indx, p130_02_Indx,
//               G01_Dado[p130_01_Indx][p130_02_Indx]);
         }
//         fprintf(p130_Arquivo, "\n");
//         fprintf (p130_Arquivo, "Somatorio(%d,%d)=%f ",
//            p130_01_Indx, p130_02_Indx,
//            G01_Dado[p130_01_Indx][p130_02_Indx]);
         G01_Dado[p130_01_Indx][p130_02_Indx] =
            p080_Funcao_Ativacao
               (G01_Dado[p130_01_Indx][p130_02_Indx],
G10_Funcao_Ativacao);
//         fprintf (p130_Arquivo, "Resultado(%d,%d)=%f\n",
//            p130_01_Indx, p130_02_Indx,
//            G01_Dado[p130_01_Indx][p130_02_Indx]);
      }
   }
}

//*********************************************************************
*******//
void p140_Backpropagation_Backward (FILE * p140_Arquivo)
{
   int p140_01_Indx = 0,
      p140_02_Indx = 0;
   double p140_03_Aux =0.;
   int p140_04_Saida_Errada = 0,
      p140_05_Indx = 0;

   G11_Saida_Errada++;
   p140_01_Indx = G04_Qtd_Camadas - 1;
```

```
   for (p140_02_Indx = 1;
      p140_02_Indx <= G01_Qtd_Neuronio[p140_01_Indx]; p140_02_Indx++)
   {
      G01_Saida_Calculada [p140_02_Indx - 1] =
         G01_Dado [p140_01_Indx][p140_02_Indx];
//      fprintf (p140_Arquivo, "Saida(%d)=%f\n",
//         p140_02_Indx, G01_Dado[p140_01_Indx][p140_02_Indx]);
      if (G01_Saida_Calculada [p140_02_Indx - 1] !=
         G01_Saida_Esperada [p140_02_Indx])
      {
         G11_Saida_Errada++;
         p140_04_Saida_Errada = 1;
         G01_Erro_Padrao +=  0.5 * raizquadrada
            (G01_Saida_Esperada[p140_02_Indx] -
            G01_Saida_Calculada [p140_02_Indx - 1]);
         fprintf (p140_Arquivo,
//            "Saida Esperada[%d]=%f Calculada(%d)=%f\n",
//            p140_02_Indx, G01_Saida_Esperada[p140_02_Indx],
//            p140_02_Indx, G01_Saida_Calculada [p140_02_Indx - 1]);
            "%f;%f\n",
            G01_Saida_Esperada[p140_02_Indx],
            G01_Saida_Calculada [p140_02_Indx - 1]);
         // achar o delta erro da camada de saida
         G01_Delta[p140_01_Indx] =
            (G01_Saida_Esperada[p140_02_Indx - 1] -
            G01_Saida_Calculada [p140_02_Indx - 1])*
            G01_Saida_Calculada [p140_02_Indx - 1] *
            (1 - G01_Saida_Calculada [p140_02_Indx - 1]);
      }
      else
      {
         p140_04_Saida_Errada = 0;
      }
      // achar o delta das camadas intermediarias
      p140_03_Aux = 0;
      for (p140_01_Indx = G04_Qtd_Camadas - 2; p140_01_Indx > 0;
         p140_01_Indx--);
      {
         for(p140_02_Indx = 0; p140_02_Indx <=
G05_Qtd_Neuronios[p140_01_Indx];
            p140_02_Indx++)
         {
            for (p140_05_Indx = 0; p140_05_Indx <
               G05_Qtd_Neuronios[p140_01_Indx + 1]; p140_05_Indx++)
            {
               p140_03_Aux += 1;

G01_Peso_Recuperado[p140_01_Indx][p140_02_Indx][p140_05_Indx]
                     * G01_Delta [p140_01_Indx + 1];
            }
         }
         G01_Delta[p140_01_Indx] = p140_03_Aux *
            G01_Dado [p140_01_Indx][p140_02_Indx] *
            (1 - G01_Dado [p140_01_Indx][p140_02_Indx]);
      }
   }
   // atualiza os pesos da camada de entrada para a camada
intermediaria
   for (p140_01_Indx = 0; p140_01_Indx < G04_Qtd_Camadas - 1;
p140_01_Indx++)
```

```
   {
      for (p140_02_Indx = 0; p140_02_Indx <
G05_Qtd_Neuronios[p140_01_Indx];
         p140_02_Indx++)
      {
         for (p140_05_Indx = 1; p140_05_Indx <=
            G05_Qtd_Neuronios[p140_01_Indx + 1]; p140_05_Indx++)
         {

G01_Peso_Variacao[p140_01_Indx][p140_02_Indx][p140_05_Indx] =
               G07_Taxa_Aprendizado * G01_Delta [p140_01_Indx + 1] +
               G06_Momento *
               G01_Peso_Recuperado
[p140_01_Indx][p140_02_Indx][p140_05_Indx];
            G01_Peso_Recuperado
[p140_01_Indx][p140_02_Indx][p140_05_Indx] =

G01_Peso_Variacao[p140_01_Indx][p140_02_Indx][p140_05_Indx];
         }
      }
   }
//   fprintf (p140_Arquivo, "\n");
}

//*****************************************************************
*******//
void p190_Recupera_Data_Hora()
{
   G99_05_Tempo = time(NULL);
   G99_06_Tempo = localtime(&G99_05_Tempo);
   strftime (G99_13_Buffer, 256, "%Y", G99_06_Tempo);
   G99_07_Ano = atoi(G99_13_Buffer);
   strftime (G99_13_Buffer, 256, "%m", G99_06_Tempo);
   G99_08_Mes = atoi(G99_13_Buffer);
   strftime (G99_13_Buffer, 256, "%d", G99_06_Tempo);
   G99_09_Dia = atoi(G99_13_Buffer);
   strftime (G99_13_Buffer, 256, "%H", G99_06_Tempo);
   G99_10_Hora = atoi(G99_13_Buffer);
   strftime (G99_13_Buffer, 256, "%M", G99_06_Tempo);
   G99_11_Minuto = atoi(G99_13_Buffer);
   strftime (G99_13_Buffer, 256, "%S", G99_06_Tempo);
   G99_12_Segundo = atoi(G99_13_Buffer);
}

//*****************************************************************
*******//
void p200_Treina_Rede(char *p200_Nome_Entrada, char *p200_Nome_Saida)
{
   int p200_01_Indx = 0, p200_02_Indx = 0, p200_03_Indx = 0;
   FILE * p200_04_Arquivo;
   char p200_05_Registro[80]="\0";
   int p200_06_Indx = 0;
   FILE * p200_07_Arquivo;
   int p200_07_Indx = 0,
      p200_08_Epoca = 0;
   char p200_09_Inicio[20];
   char p200_10_Final[20];
   // abre os arquivos
   p200_04_Arquivo = fopen (p200_Nome_Entrada, "r");
   p200_07_Arquivo = fopen (p200_Nome_Saida, "w");
```

```
   // inicia a leitura do arquivo de entrada
   p200_08_Epoca = 1;
   //
   p190_Recupera_Data_Hora();
   sprintf (p200_09_Inicio, "%04d/%02d/%02d=%02d:%02d:%02d",
      G99_07_Ano, G99_08_Mes, G99_09_Dia, G99_10_Hora, G99_11_Minuto,
      G99_12_Segundo);
   //
   while (p200_08_Epoca <= G09_Epoca)
   {
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo,
            "Epoca=%04d\n\n", p200_08_Epoca);
      }
      rewind (p200_04_Arquivo);
      // le arquivo de entrada
      fgets(p200_05_Registro, 80, p200_04_Arquivo);
      while (!feof(p200_04_Arquivo))
      {
         p100_Limpa_Dados_Entrada();
         p200_06_Indx = atoi(strtok(p200_05_Registro, ";"));
//         fprintf (p200_07_Arquivo, "Reg.=%03d ", p200_06_Indx);
         // recupera os dados de entrada da rede
         p110_Recupera_Dados_Entrada(p200_07_Arquivo, p200_06_Indx);
         // recupera os dados esperados de saida
         p120_Recupera_Dados_Saida (p200_07_Arquivo, p200_06_Indx);
         // faz a parte forward do backpropagation
         p130_Backpropagation_Forward (p200_07_Arquivo);
         // faz a parte backward do backpropagation
         p140_Backpropagation_Backward (p200_07_Arquivo);
         // le arquivo de entrada
         fgets(p200_05_Registro, 80, p200_04_Arquivo);
      }
/*      if (G01_Erro_Padrao < G08_Erro_Alvo)
      {
         // tem que ter um local para guardar os pesos
         G08_Erro_Alvo = G01_Erro_Padrao;
      }
      else
      {
         if (G01_Erro_Padrao > (G08_Erro_Alvo * 1.2))
         {
            if (G99_04_Tem_Log == 1)
            {
               fprintf (G99_01_Arquivo,
                  "Erro Padrao=%f Erro Esperado=%f\n\n",
                  G01_Erro_Padrao, G08_Erro_Alvo);
            }
            printf ("\n\nProcessamento interrompido por causa do
erro\n\n");
            p200_08_Epoca = G09_Epoca + 5;
            break;
         }
      }
*/      p200_08_Epoca++;
   }
   p190_Recupera_Data_Hora();
   sprintf (p200_10_Final, "%04d/%02d/%02d=%02d:%02d:%02d",
      G99_07_Ano, G99_08_Mes, G99_09_Dia, G99_10_Hora, G99_11_Minuto,
```

```
      G99_12_Segundo);
   if (G99_04_Tem_Log == 1)
   {
      if (p200_08_Epoca > G09_Epoca + 5)
         fprintf (G99_01_Arquivo,
            "Processamento Interrompido - Inicio=%s Final=%s\n\n",
            p200_09_Inicio, p200_10_Final);
      else
         fprintf (G99_01_Arquivo,
            "Epoca=%04d Inicio=%s Final=%s\n\n",
            p200_08_Epoca - 1, p200_09_Inicio, p200_10_Final);
   }
   //
   fclose (p200_04_Arquivo);
   fclose (p200_07_Arquivo);
}

//*****************************************************************
******//
int p210_Avalia_Rede()
{
}

//*****************************************************************
*******//
int main(int argc, char *argv[])
{
   float L01_Entrada = 2;
   float L02_Saida = 0.;
   char L03_Nome_Arquivo [32];
   char L04_Nome_Arquivo [32];
   char L05_Nome_Arquivo_Aux [32] = "\0";

   printf ("Gravar Log do Processamento (0=Nao / 1=Sim) ?");
   scanf ("%d", &G99_04_Tem_Log);
   printf ("\n\n");

   p190_Recupera_Data_Hora();
   sprintf (L05_Nome_Arquivo_Aux, "%04d%02d%02d_%02d%02d%02d_",
      G99_07_Ano, G99_08_Mes, G99_09_Dia, G99_10_Hora, G99_11_Minuto,
      G99_12_Segundo);
   if (G99_04_Tem_Log == 1)
   {
      sprintf (G99_02_Nome_Arquivo,
"%04d%02d%02d_%02d%02d%02d_LOG.TXT",
         G99_07_Ano, G99_08_Mes, G99_09_Dia, G99_10_Hora,
G99_11_Minuto,
         G99_12_Segundo);
      G99_01_Arquivo = fopen (G99_02_Nome_Arquivo, "w");
   }

   if (p010Le_Entrada()==0)
   {
      p020_Inicializa_Randomizacao();
      p040_Cria_Rede();
      //
      strcpy (L03_Nome_Arquivo, L05_Nome_Arquivo_Aux);
      strcat (L03_Nome_Arquivo, "PR.TXT");
      if (G99_04_Tem_Log == 1)
      {
```

```
         fprintf (G99_01_Arquivo,
            "Peso Recuperado=%s\n\n", L03_Nome_Arquivo);
      }
      p050_Recupera_Peso(L03_Nome_Arquivo);
      //
      strcpy (L03_Nome_Arquivo, L05_Nome_Arquivo_Aux);
      strcat (L03_Nome_Arquivo, "DADO_TRE.TXT");
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo,
            "Dado Teino=%s\n\n", L03_Nome_Arquivo);
      }
      p070_Seleciona_Massa(L03_Nome_Arquivo, 0.4);
      //
      strcpy (L03_Nome_Arquivo, L05_Nome_Arquivo_Aux);
      strcat (L03_Nome_Arquivo, "DADO_TES.TXT");
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo,
            "Dado Teste=%s\n\n", L03_Nome_Arquivo);
      }
      p070_Seleciona_Massa(L03_Nome_Arquivo, 0.3);
      //
      strcpy (L03_Nome_Arquivo, L05_Nome_Arquivo_Aux);
      strcat (L03_Nome_Arquivo, "DADO_TRE.TXT");
      strcpy (L04_Nome_Arquivo, L05_Nome_Arquivo_Aux);
      strcat (L04_Nome_Arquivo, "SAIDA_TRE.TXT");
      if (G99_04_Tem_Log == 1)
      {
         fprintf (G99_01_Arquivo,
            "Saida Treino=%s\n\n", L04_Nome_Arquivo);
      }
      p200_Treina_Rede(L03_Nome_Arquivo, L04_Nome_Arquivo);
//    p210_Avalia_Rede();
//      p200_Treina_Rede("RNL-TS.TXT", "RNL-SS.TXT");
//      printf ("%f ", p080_Funcao_Ativacao(L01_Entrada,
G10_Funcao_Ativacao));
//      printf ("%f\n",
//         p090_Derivada_Funcao_Ativacao(L01_Entrada,
G10_Funcao_Ativacao));
   }

   fclose(G99_01_Arquivo);

   system("PAUSE");
   return;
}
```