ANALÚCIA SCHIAFFINO MORALES DE FRANCESCHI

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE AGENTES PARA GERÊNCIA DE REDES

FLORIANÓPOLIS
2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE AGENTES PARA GERÊNCIA DE REDES

Tese de Doutorado submetida à
Universidade Federal de Santa Catarina

ANALÚCIA SCHIAFFINO MORALES DE FRANCESCHI

Florianópolis, janeiro de 2003.

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL NO DESENVOLVIMENTO DE AGENTES PARA GERÊNCIA DE REDES

Analúcia Schiaffino Morales De Franceschi

Janeiro/2003

Orientador: Jorge Muniz Barreto.
Área de Concentração: Sistemas de Informação.
Palavras-chave: complexidade de RNAs, agentes inteligentes, redes recorrentes, aprendizado por exemplos, gerência de redes.
Número de Páginas: 145.

O presente trabalho apresenta uma metodologia para aplicação de técnicas de inteligência artificial na área de gerência de redes. O objetivo é buscar técnicas alternativas para auxiliar no trabalho de administradores de redes de computadores. A partir das pesquisas desenvolvidas aplicando técnicas de IA em gerência de redes será possível buscar alternativas para extrair as configurações para os sistemas de gerência de redes a partir da própria rede.

A metodologia foi aplicada no desenvolvimento de *baselines* para a gerência pró-ativa de redes. Para obter uma função *baseline* é necessário realizar uma atividade de amostragem durante vários períodos de tempo, identificando o desempenho normal da rede através de médias e cálculos estatísticos. Estabelecido o perfil da rede, utiliza-se a função para comparar o funcionamento atual com o estabelecido pelo perfil da rede. Com base nestas comparações é possível realizar um gerenciamento preventivo.

Foram testadas três redes neurais artificiais (RNAs) recorrentes com diferentes números de entradas e de camadas intermediárias para testar o aprendizado através de exemplos. Foram fornecidas amostras de dados de utilização da rede no sentido de verificar a utilização das RNAs recorrentes no desenvolvimento de *baselines*. Experimentos foram realizados e os resultados foram respectivamente, analisados e comparados com o desenvolvimento tradicional de *baselines.*

Abstract of Thesis presented to UFSC as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

EMPLOYING ARTIFICIAL INTELIGENCE TECHINIQUES TO THE DEVELOPMENT OF NETWORK MANAGEMENT AGENTS

Analúcia Schiaffino Morales De Franceschi

January/2003

Advisor: Jorge Muniz Barreto.
Area of Concentration: Information Systems.
Keywords: dynamic systems, autonomous agents, recurrent networks, learning from examples, computer network management
Number of Pages: 145.

The essence of this work is investigating the how distributed problem solving of the computer network management may employ the artificial intelligence issues. The AI techniques may be applied for automating the network management process. These techniques had been tested to extract the data configuration from the computer network environment.

Following methodology is defended in the present work. If the problem has been defined then it must have a heuristic solution. There are two techniques to implement this case, through production rules (Symbolic paradigm) or using feed forward neural networks (Connectionist paradigm). If the problem has no defined and the output is unknown it must be applied dynamic techniques. The dynamic problems must be "well" solved by dynamic tools. There are following forms presently to include dynamism in a neural network solution. To apply a sequential line of time delays between two inputs of a feed forward neural network, or using a network with cycles and dynamical neurons (ex: Hopfield network and recurrent neural networks). So, the example usage is proposed. If the input and output of neural network were known we may use a recurrent neural network that may be trained estimating the unknown state changes. If there are only input samples so the recurrent network must be able to provide the output function using the adaptative learning.

The methodology had been applied to the baselines development for proactive network management. The baseline must represent the normal behavior of the computer network. It may be construct employing statistics activities, such as averages and other calculus. This work presents some experiments using artificial neural networks (ANNs) to create baselines for proactive network management.

Three different recurrent ANNs had been examined. They had number of the inputs and the layers different and were adapted from examples. The data samples had the utilization rate and were used to feed the recurrent ANNs. The output of a recurrent ANN must represent the network behavior, which is the baselining. The results were analyzed and compared with the traditional method for baselining.

Publicações

[1] DE FRANCESCHI, A.S.M., MORAES, R., BARRETO, J.M., ROISENBERG, M. Employing Recurrent Artificial Neural Networks for Developing Baselines for Proactive Network Management. In: IASTED INTERNATIONAL CONFERENCE ON APPLIED INTELLIGENCE, 2003, Cancun. **Proceedings**… Mexico : IASTED Press, 2003.

[2] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Desenvolvendo Agentes de Software para Gerência de Redes Utilizando Técnicas de Inteligência Artificial. In: AGENT´S DAY – CBCOMP 2002 - CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 2002, Itajaí. **Anais...** Itajaí : Editora da Univali, 2002. (ISSN 1677-2822).

[3] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Aplicando Técnicas de Inteligência Artificial para Desenvolver Agentes para Gerência de Redes de Computadores. In: SBRN'2000 SIMPÓSIO BRASILEIRO DE REDES NEURAIS, 2000, Rio de Janeiro. **Anais**... Rio de Janeiro : SBRN Editora, 2000. Vol. II

[4] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Constructing Software Autonomous Agents to Computer Network Management. In: SBRN'2000 SIMPÓSIO BRASILEIRO DE REDES NEURAIS, 2000, Rio de Janeiro. **Resumo...** Rio de Janeiro : IEEE Press, 2000. Vol. I

[5] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Desenvolvimento de Agentes Autônomos em Gerência de Redes de Computadores. In: XVIII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 2000, Gramado. **Anais…** Gramado : 2000.

[6] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Intelligent, Dynamic and Distributed Solutions for Network Management. In: EXPO 2000 SHAPING-THE-FUTURE CONFERENCE - GLOBAL DIALOGUE, 2000, Hannover. **Proceedings**... Hannover : 2000.

[7] DE FRANCESCHI, A.S.M., BARRETO, J.M., ROISENBERG, M. Autonomous Software Agents for Computer Network Management. In: ICT'2000 IEEE INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 2000, Acapulco. **Proceedings…** Mexico : 2000.

[8] DE FRANCESCHI, A.S.M., ROISENBERG, M., BARRETO, J.M. Employing Intelligent Techniques to Develop Autonomous Agents for Network Management. In: AGENTS BASED SIMULATION WORKSHOP, 2000, Passau. **Proceedings…** Germany : SCS – International Society for Computer Simulation, 2000.

[9] DE FRANCESCHI, A.S.M., BARRETO, J.M. Distributed Problem Solving Based on Recurrent Neural Networks Applied to Computer Network Management. In: ICT'99 INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 1999, Cheju. Proceedings… Korea : IEEE Press, 1999.

Sumário

Publicações vi

Sumário vii

Lista de Figuras xi

Lista de Tabelas 13

Lista de Siglas 14

Capítulo 1 Introdução 15

1.1 Motivação 15

1.2 Objetivos 19

1.2.1 Objetivo Principal 19

1.3 Objetivos Secundários 20

1.4 Organização do Trabalho 20

Capítulo 2 Inteligência Artificial Conexionista 22

2.1 Introdução 22

2.2 Redes Neurais Artificiais (RNAs) 25

2.3 Modelos de Neurônio 25

2.4 Topologias das RNAs 27

2.5 Algoritmos de Treinamento 29

Supervisionado 29

Não Supervisionado 29

Por Reforço 29

Aprendizado em RNAs Recorrentes 30

2.6 RNAs Diretas versus RNAs Recorrentes 31

2.7 RNAs Recorrentes 32

2.8 Computabilidade e Complexidade de RNAS 33
2.9 Equivalência de Computabilidade 34
2.10 Complexidade de RNAS 34
2.11 Alguns Teoremas 35
2.12 Teoria da Complexidade Conexionista 36

Capítulo 3 IAC aplicada à Gerência de Redes 38

3.1 Introdução 38
3.2 Gerência de Sistemas 39
3.3 Áreas Funcionais 40
3.4 Gerência Internet 42
3.4.1 SNMPv2 e SNMPv3 42
3.4.2 Elementos da arquitetura SNMP 43
3.4.3 Gerência OSI X Gerência Internet 44
3.5 Comportamentos da Gerência 45
3.6 Aplicação da IAC 47
3.7 Utilização de RNAs Recorrentes 48
3.7.1 Redes diretas versus redes dinâmicas 48

Capítulo 4 Teoria de Sistemas 51

4.1 Introdução 51
4.2 Sistema Geral 52
4.3 Sistema Orientado 53
4.4 Sistema Temporal 54
Exemplos de Sistema Temporal 54
4.5 Sistema Funcional 56
4.6 Sistema Dinâmico 56
Exemplos de Sistema Dinâmico 58
Modelo Geral de Neurônio como Exemplo de Sistema Dinâmico 58
4.7 Sistema Complexo 59
Exemplos de Sistema Complexo 60

Capítulo 5 Desenvolvimento de Agentes Inteligentes e Distribuídos.....61

5.1 Introdução 61

5.2 Agentes na Solução de Problemas 62

5.3 Agentes sem Técnicas de IA.....64

5.4 Metodologia para aplicação de IA 65
5.4.1 Problemas bem definidos.....67
5.4.2 Problemas mal definidos.....67

5.5 Agentes com Técnicas de IA 68
5.5.1 Agentes Inteligentes Estáticos.....68
Outros exemplos.....69
5.5.2 Agentes Inteligentes Dinâmicos 70
Outro exemplo 71

5.6 Trabalhos relacionados.....72

5.7 Plataformas para Desenvolvimento de Agentes.....73
ABLE (Agent Building Learning Environment) 74
JDMK (Java Dynamic Management Kit).....74
Aglets.....75
JADE (Java Agent Development Environment).....75
JESS (Java Expert System Shell).....75
Linguagem KQML (Knowledge Query and Manipulation Language).....76

Capítulo 6 Experimentos Realizados 77

6.1 Introdução 77

6.2 Testes com a Rede Recorrente de Elmann 77

6.3 Definição do Problema 80
6.3.1 Desenvolvimento de Baselines.....80
Taxa de utilização 81

6.4 Ambiente de Testes.....81
6.4.1 Obtenção dos dados da rede.....82
6.4.2 MIB BROWSER.....83

6.5 Coleta dos Dados.....84

6.6 Modelagem da Rede Neural Recorrente.....85

6.6.1 Definição do problema: desenvolvimento de baselines 85
6.7 RNA recorrente com uma camada 87
6.7.1 Treinamento da RNA recorrente com uma camada 88
6.7.2 Resultados da RNA recorrente com uma camada 89
6.8 Rede recorrente com três camadas 91
6.8.1 Treinamento da RNA recorrente com três camadas 93
6.8.2 Resultados da RNA recorrente com três camadas 94
6.9 Rede recorrente com cinco camadas 96
6.9.1 Resultados da RNA recorrente com cinco camadas 99
6.10 Comparação com o método tradicional 100

Capítulo 7 Conclusões 102

7.1 Considerações Finais 102
7.2 Trabalhos Futuros 105

Anexo I – Script da rede de Elmann 107

Anexo II – Script das RNAs recorrentes 109

Anexo III – Exemplo de arquivo de *log* 124

Anexo IV – Script de leitura dos arquivos de log 134

Glossário 137

Referências Bibliográficas 142

Lista de Figuras

Figura 1. Várias abordagens de IA.....23
Figura 2. (a) Neurônios biológicos e (b) Neurônios artificia is.....26
Figura 3. O neurônio artificial.....27
Figura 4. Classificação para RNAs.....28
Figura 5. Implementação neural de autômato finito.....32
Figura 6. Problemas linear e não-linearmente separáveis.....35
Figura 7. Relacionamento gerente-agentes utilizando o protocolo SNMP.....44
Figura 8. Comportamento da gerência de redes.....45
Figura 9. Modelo para a gerência pró-ativa de redes [30].....46
Figura 10 - Exemplo de utilizaçao de RNA Direta x RNA Recorrente.....50
Figura 11. Teoria de sistemas.....52
Figura 12. O sistema geral visto como uma caixa preta.....52
Figura 13. Sistema orientado.....53
Figura 14. Sistema temporal.....54
Figura 15. Taxa de utilização de uma rede.....55
Figura 16. Sistema funcional.....56
Figura 17. Sistema dinâmico.....57
Figura 18. Sistema complexo.....59
Figura 19. Sistemas multi-agentes como sistemas complexos.....60
Figura 20. Taxonomia de agentes.....63
Figura 21. Metodologia híbrida para solução de problemas distribuídos.....66
Figura 22. Agentes inteligentes estáticos.....68
Figura 23. RNA baseada em regras.....69
Figura 24. Agentes inteligentes dinâmicos.....70
Figura 25. Rede recorrente que implementa um agente interpretador de eventos.....71
Figura 26. Rede recorrente de Elmann [37].....78
Figura 27. Treinamento para a rede de Elmann.....79
Figura 28. Generalização da rede de Elmann.....80
Figura 29. Estrutura da rede do CAV/UDESC.....82
Figura 30. Frame para coletar dados.....84
Figura 31. Função de transferência utilizada.....87
Figura 32. Rede neural recorrente personalizada.....88
Figura 33. Convergência do erro e número de épocas.....89
Figura 34. Resultado do treinamento da RNA 1 camada.....90
Figura 35. Simulação do dias 19 de dezembro.....90
Figura 36. Simulação do dias 20 de dezembro.....91
Figura 37. Simulação com amostras de dados de outra interface.....91
Figura 38. Rede recorrente com três camadas.....92
Figura 39. Convergência do erro e número de épocas.....94
Figura 40. Resultado do treinamento da RNA – 3 camadas.....95
Figura 41. Simulação da rede dia 19/dez.....95
Figura 42. Simulação da rede dia 20/dez.....95

Figura 43. Simulação com amostras de dados de outra interface 96
Figura 44. Rede recorrente com cinco camadas 97
Figura 45. Convergência do erro e número de épocas 98
Figura 46. Resultado do treinamento da RNA – 5 camadas 98
Figura 47. Simulação da rede dia 05/dez 99
Figura 48. Simulação da rede dia 05, 06, 07, 10 e 12/dez 99
Figura 49. Simulação com amostras de dados de outra interface 100

Lista de Tabelas

Tabela 1. Principais diferenças entre RNAs diretas e RNAs recorrentes.....31
Tabela 2. Computador baseado em instrução x neurocomputador.....33
Tabela 3. Principais funções das FCAPS.....41
Tabela 4. Exemplos de mensagens traps do SNMP.....64
Tabela 5. Sinais de ativação do agente – entradas e saídas.....72
Tabela 6. Objetos da MIB utilizados para cálculo da taxa de utilização.....85

Lista de Siglas

CCITT	*Consultative Committee on International Telephony and Telegraphy*
CMIP	*Common Management Information Protocol*
CMISE	*Common Management Information Services*
ComSoc	*Communication Society*
DCE	*Data Communications Equipment*
DME	*Distributed Management Environment*
DTE	*Data Terminal Equipment*
EMA	*Enterprise Management Architecture*
FCAPS	*Fault Configuration Account Performance and Security*
IAC	Inteligência Artificial Conexionista
IAE	Inteligência Artificial Evolucionária
IAH	Inteligência Artificial Híbrida
IAS	Inteligência Artificial Simbólica
IEC	*International Electrotechnical Committee*
IEE	*The Institution of Electrical Engineers*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*International Engineering Task Force*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
ITU-T	*International Telecommunications Union*
LAN	*Local Area Networks*
MIB	*Management Information Base*
NIST	*National Institute of Standards and Technology*
OMNI*Points*	*Open Management Interoperability Points*
OSF	*Open Software Foundation*
OSI	*Open Systems Interconnection*
PDU	*Protocol Data Unit*
SNMP	*Simple Network Management Protocol*
TCP	*Transport Control Protocol*
TMN	*Telecommunications Management Network*
UDP	*User Datagram Protocol*
UNMA	*Universal Network Management Architecture*
WAN	*Wide Area Networks*

Capítulo 1

Introdução

1.1 Motivação

"A primeira visão de Cris pela manhã foram pelo menos uns cem recados, no seu terminal uma mensagem piscava fora de controle e havia várias pessoas rondando o laboratório. Em coro, gritaram:" "A rede está fora do ar!". Estava começando mais um dia na vida de um administrador de rede. Rapidamente, começou a pressionar as teclas de seu terminal. Chicago, Singapura, Nova Iorque, São Francisco e Paris, não havia acesso nenhum. Os principais acessos da *MegaNet Company* pareciam ter desaparecido. Cris suspirou e por um instante gostaria de apertar um botão sobre a parede dizendo: "Não entrem em pânico!". No setor de atendimento a clientes, ouvia-se uma voz ao fundo: "Desculpe pode ligar mais tarde, a rede está fora do ar...". À direita, o setor de cobrança aguardava para entrar os dados da empresa. O pessoal do departamento de pesquisa e desenvolvimento foi dispensado e escalonado para trabalhar após as 17 horas. Apenas Cris e os gerentes ocupavam o escritório. Cris sabia exatamente como seu dia iria proceder: isolar o problema, solucionar o problema e imprimir relatórios e gráficos de gerência mostrando o que aconteceu e por quê, para não se repetir."

A história foi retirada de [63] ilustrando um cenário familiar àqueles que trabalham ou utilizam as redes de computadores. As redes vêm se tornando muito comuns no dia-a-dia. É o caso dos serviços dos bancos que podem ser acessados de casa ou de um terminal 24 horas. Compras com cartões de crédito podem ser aprovadas imediatamente, a qualquer hora do dia ou da noite. É possível sentar a frente de um computador pessoal e viajar através da Internet para qualquer lugar no mundo, acessando de forma rápida e eficiente uma vasta quantidade de

informação. No entanto, no caso de quedas de acesso destas redes, importantes negócios podem ser perdidos. Podem ocorrer atrasos no recebimento de dados importantes.

As redes de computadores são compostas por meios físicos e lógicos utilizados para compartilhar recursos entre máquinas. Existem ainda centenas de palavras que fazem parte do jargão relacionado às redes maiores, tais como, *gateways*, *bridges*, *hosts*, *hubs*, *switches* e alguns nomes de tecnologias de transmissão. Além de protocolos de gerência, como SNMP (*Simple Network Management Protocol*) e CMIP (*Common Management Interface Protocol*), protocolos de transmissão, como TCP (*Transport Control Protocol*) e UDP (*User Datagram Protocol*), protocolos de rede, como o IP (*Internet Protocol*), protocolos de enlace e milhares de aplicações.

As redes podem ser subdivididas pela abrangência geográfica em redes locais de computadores (LAN), redes metropolitanas (MAN) e redes de longa distância (WAN). As redes locais que podem ser interligadas umas com as outras formando uma grande rede local, desde que não ultrapassem poucos quilômetros. As redes metropolitanas são mais abrangentes do que as redes locais integrando, por exemplo, uma área ocupada por uma cidade inteira. As redes de longa distância envolvem grandes distâncias geográficas, como é o caso do *backbone* da Internet (linha dedicada que interliga uma rede local em uma rede de longa distância, neste caso a rede mundial de computadores – Internet) [3].

É comum nos dias atuais, a ocorrência de falhas durante uma operação em uma rede. As redes são equipamentos que operam de forma distribuída e estão sujeitos a diversos tipos de falhas: falhas nos equipamentos devido à ação do tempo, como umidade e calor excessivo; falhas operacionais ou de uso indevido dos equipamentos; falhas de sobrecarga; etc. Além de falhas existem outras preocupações para aqueles que utilizam ou administram seus serviços. A configuração, o desempenho, a contabilização e a segurança são outros aspectos funcionais sobre as redes que fazem parte de atividades de gerência de redes. Gerência de redes é o processo para monitoração e controle de uma rede de dados complexa (pode ser de computadores ou de telecomunicações) para aumentar a sua eficiência e garantir a sua produtividade [83].

A gerência de redes de computadores por um longo período foi caracterizada como proprietária, desenvolvida por cada fabricante. São exemplos de sistemas de gerência o *SunNet*

Manager (da SunConnect), o *NetView 6000 (da IBM)* e o *OpenView (da HP Hewllet-Packard).* A AT&T possui seu próprio pacote de produtos e protocolos chamado UNMA, a *Digital Equipment Corporation* possui uma arquitetura proprietária chamada EMA e seu sistema de gerência chama-se *DECmcc Director* [100].

Na verdade os protocolos de gerência disponíveis não satisfaziam todos os fabricantes. O modelo de referência OSI definiu uma série de normas para a gerência de redes. Entre outras coisas, definiu em modelo estrutural, modelo informacional e modelo funcional do protocolo de gerência CMIP. Quanto à estrutura foram especificados os objetos de gerência, o comportamento de agentes e o comportamento de gerentes. Quanto ao modelo informacional, determinou como o processo gerente deve invocar as operações de gerência e como o sistema gerenciado envia as notificações ao gerente, bem como de qual forma os dados permanecerão armazenados na Base de Informação de Gerência (MIB). Quanto à funcionalidade, a gerência foi dividida em cinco áreas funcionais: configuração, contabilização, desempenho, falhas e segurança.

No padrão Internet, existe um protocolo mais simples, chamado SNMP. O SNMP é baseado em pergunta/resposta (*request/reply*) e é muito simples. Foi introduzido no final dos anos 80 para controlar e monitorar redes TCP/IP. Devido a sua simplicidade para implementação e baixos custos, agentes baseados neste protocolo foram implementados por diversos fabricantes de redes. No sentido de melhorar a segurança dos endereçamentos e aperfeiçoar o protocolo SNMP foi lançada a versão dois. O SNMPv2, como é chamado, suporta entre outras novidades, comunicação de gerente-gerente e recuperação de um bloco de dados (na outra versão, era feito linha a linha). Atualmente, foi lançada a versão três (SNMPv3) com recursos de segurança e capacidade de configuração remota [63][83][85][91].

Conforme [91], as principais diferenças entre o CMIP e o SNMP destacam-se:

- CMIP define um conjunto de mensagens muito grandes, ao contrário do SNMP que é simplificado;
- o CMIP distribui a carga de tráfego gerada por gerentes e agentes;
- as mensagens do CMIP contêm informações sobre os parâmetros e identificação dos objetos gerenciados.

O protocolo CMIP possui mais recursos de gerência, porém seu custo é elevado, mais difícil de ser implementado. O SNMP, por outro lado, é mais simples e mais utilizado. Normalmente, os equipamentos de redes gerenciáveis possuem suporte ao SNMP.

Em meados dos anos 90 iniciou-se um processo para distribuir a gerência em ambientes heterogêneos. O objetivo de integrar estas redes é criar a imagem de uma rede virtual única, conhecida como arquitetura aberta, formada por componentes de diversos fabricantes, para facilitar o processo de gerência. Vários órgãos de regulamentação reuniram-se com o intuito de criar uma regulamentação para as redes interoperáveis, tais como: IETF, OSF, NIST, OMNI*Points*, etc. O DME (*Distributed Management Environment*) é um conjunto de especificações para produtos de gerência de redes distribuída. O DME está baseado no CMIP e no SNMP, baseado no paradigma orientado à objetos [85][100]. Em seguida surgiram os objetos distribuídos, destaque para a utilização de linguagens como JAVA e a plataforma CORBA para gerência de redes.

Atualmente existem plataformas para o desenvolvimento de agentes, como é o caso do JDMK (*Java Development Management Kit)* da Sun Microsystems e os *aglets* da IBM. Ressalta-se, que alguns fabricantes apontam seus produtos como inteligentes, no sentido da utilização dos recursos e desenvolvimento de aplicações e não, necessariamente, devido a utilização de técnicas de IA [12][58].

No Brasil, segundo artigo do professor da UFMG, José Marcos Nogueira [79] a pesquisa e desenvolvimento na área de gerência de redes de computadores e telecomunicações vêm sendo realizada, em sua grande maioria, por departamentos e instituições universitárias (geralmente públicas). Alguns grupos de instituições associam-se a empresas particulares ou estatais buscando solução de problemas práticos. Existem, também, grupos multi-institucionais financiados por agências públicas que trabalham em projetos temáticos. Um destes projetos, ainda citado em [79], foi o Projeto PlaGeRe (Plataformas para Gerência de Redes), financiado pelo CNPq no Programa ProTeM-CC.

1.2 Objetivos

O processo de gerência de redes não é automatizado por completo, isto é, necessita ainda da intervenção humana na tomada de decisões. Os sistemas de gerência disponíveis no mercado, citados no item anterior, são ferramentas que auxiliam o processo fornecendo relatórios e emitindo alarmes, após alguma falha ser detectada. São sistemas passivos que não executam nenhum tipo de tomada de decisões aguardam que o administrador da rede solucione o problema. O objetivo da pesquisa é definir como as técnicas de inteligência artificial (IA) poderão auxiliar no desenvolvimento de agentes para a gerência de redes de computadores.

1.2.1 Objetivo Principal

O principal objetivo deste trabalho é explorar como as técnicas de inteligência artificial podem auxiliar no processo de automação de gerência de redes. Tais técnicas podem ser utilizadas para acrescentar um comportamento inteligente aos agentes de gerência de redes. E de certa forma auxiliar na configuração dos sistemas de gerência extraindo os dados necessários do próprio ambiente.

Na primeira etapa deste trabalho foram determinados alguns tipos de problemas na área de gerência de redes [29]. Assim os problemas foram classificados como estáticos ou dinâmicos. A segunda etapa consistiu em determinar quais os comportamentos para a gerência de redes: reativo (quando a tomada de decisões ocorre após o problema ter acontecido) ou pró-ativo (quando existem ações preventivas).

Definem-se duas formas distintas de aplicar as técnicas de IA aos agentes de gerência. É possível, baseado em heurísticas de administradores de redes, criar-se regras de produção ou utilizando-se RNAs diretas para desenvolver sistemas estáticos [33][34]. Ou utilizando exemplos, em que as entradas são fornecidas a uma RNA recorrente responsável por estimar as trocas de estados do sistema (abordagem dinâmica). A função de saída da RNA recorrente representará uma solução aceitável para o conjunto de entradas [31][32].

A metodologia foi aplicada no desenvolvimento de *baselines* para a gerência pró-ativa de redes. Para obter uma função *baseline* é necessário realizar uma atividade de amostragem durante vários períodos de tempo, identificando o desempenho normal da rede através de

médias e cálculos estatísticos. Estabelecido o perfil da rede, utiliza-se uma função para comparar o funcionamento atual com o estabelecido pelo perfil da rede. Com base nestas comparações é possível realizar um gerenciamento preventivo.

1.3 Objetivos Secundários

Foram testadas três RNAs recorrentes com diferentes números de entradas e de camadas intermediárias para testar aprendizado através de exemplos. Foram fornecidas amostras de dados de utilização da rede no sentido de verificar-se a utilização das RNAs recorrentes no desenvolvimento de *baselines*. Experimentos foram realizados e os resultados foram analisados e comparados com o desenvolvimento tradicional de *baselines*.

As seguintes etapas foram necessárias para que o objetivo geral do trabalho fosse atingido:

- Estudo sobre redes neurais artificiais;
- Definição do ambiente de testes para a coleta dos dados;
- Escolha de ferramentas para auxiliar nos testes: MatLab e *MIB Browser*;
- Aplicação da metodologia;
- Escolha das RNAs utilizadas;
- Realização dos experimentos;
- Análise dos resultados;
- Definição de conceitos sobre complexidade de RNAs.

1.4 Organização do Trabalho

O presente trabalho está organizado em seis capítulos. Segue esta introdução, o segundo capítulo com aspectos importantes de Inteligência Artificial e um estudo sobre as redes neurais artificiais (RNAs), em que são apresentados modelos de neurônios, as topologias mais conhecidas de RNAs e alguns algoritmos de aprendizado. Com o objetivo de diferenciar-se redes diretas e redes recorrentes é apresentado um quadro comparativo entre essas duas

topologias. No final do capítulo são abordados estudos de complexidade e computabilidade das RNAs.

O terceiro capítulo apresenta a inteligência artificial conexionista (IAC) aplicada à área de gerência de redes. Consta uma breve explicação de gerência de sistemas e das principais funções da área de gerência. Em seguida, apresenta-se o comportamento esperado para o gerenciamento de redes, que poderá ser reativo ou pró-ativo. E por fim, apresenta-se justificativas sobre porque aplicar técnicas de IAC e porque utilizar RNAs recorrentes.

O quarto capítulo apresenta os aspectos mais relevantes da teoria geral de sistemas. Apresentam-se diferença entre os sistemas estáticos e os dinâmicos. Estas definições são importantes porque são utilizadas no desenvolvimento da metodologia.

A essência e a originalidade do trabalho estão delineadas através dos itens do quinto capítulo. Apresenta-se a metodologia para aplicação de técnicas de IA considerando se os problemas são bem definidos ou se são problemas mal definidos, bem como, se são problemas estáticos ou se são problemas dinâmicos.

No sexto capítulo estão os aspectos da implementação e os experimentos realizados durante o desenvolvimento do trabalho. Foram propostas três topologias de RNAs recorrentes com números de camadas diferentes. Tais RNAs foram capazes de aprender os conjuntos de exemplos fornecidos como entrada. Os resultados dos treinamentos são apresentados neste capítulo.

Seguem as Conclusões, os Anexos, o Glossário e as Referências Bibliográficas.

Capítulo 2
Inteligência Artificial Conexionista

2.1 Introdução

Existem várias abordagens na área de Inteligência Artificial (IA), nas quais variam a manipulação do conhecimento, no sentido de como adquirí-lo, armazená-lo e empregá-lo. É proposta uma classificação para as diferentes áreas de IA, quanto ao método de solução de problemas e quanto à localização espacial. No primeiro caso, tem-se a IA Simbólica (IAS), a IA Conexionista (IAC), a IA Evolucionária (IAE) e a IA Híbrida (IAH). E quanto à localização espacial tem-se a IA Monolítica (IAM) e a IA Distribuída (IAD).

A IAS possui como ferramenta básica para manipular o conhecimento, a lógica, com suas regras de inferência inspirada nos silogismos enunciados há mais de 2.000 anos por Aristóteles. A IAC usando redes neurais artificiais, aplica-se a problemas mal definidos, mas que são conhecidos através de exemplos. Entre os campos de aplicações das técnicas conexionistas estão: reconhecimento de padrões; controle de processos industriais; robótica; e também, como opção às técnicas de raciocínio baseado em casos para a resolução de problemas. Na IAE, os mecanismos utilizados são os mesmos encontrados na evolução biológica. Pode ser considerado um método de otimização com restrições variáveis e muitas vezes desconhecidos [8].

A IAH reúne vantagens de mais de um tipo de método de abordagem para a resolução de problemas. A IAM envolve sistemas simples sem modularidade, como é o caso de sistemas especialistas. Enquanto que, o funcionamento da IAD depende de um determinado conjunto de partes (ou módulos) para resolver de modo cooperativo um determinado tipo de problema. Essa modularidade e a distribuição para a solução de problemas são solucionadas através da

implementação de sistemas multi-agentes [86]. Na Figura 1 é apresentado um diagrama correspondente à classificação das técnicas de IA quanto ao método de solução de problemas e quanto à lo calização espacial.

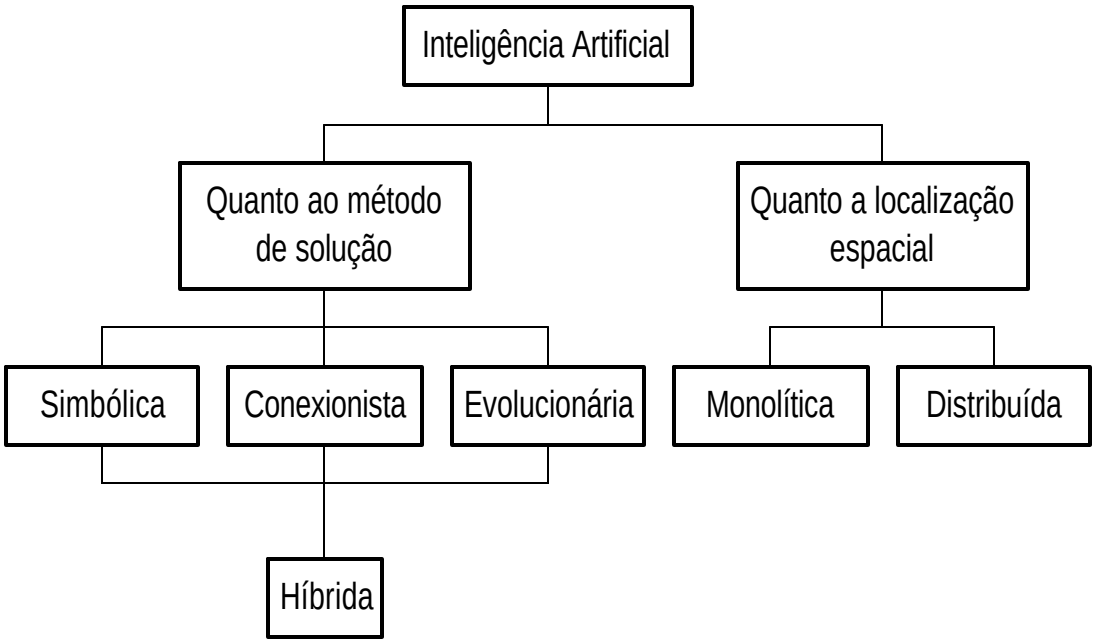


Figura 1. Várias abordagens de IA

Tanto a IAS quanto a IAC podem ser utilizadas em vários tipos de aplicações, porém em alguns casos existem particularidades que podem tornar uma mais atrativa do que a outra. Em Barreto [8], é apresentado uma série de aplicações na área de IA. Selecionaram-se algumas que poderiam ser relacionadas com a área de gerência de redes, para interesse do presente trabalho.

- Base de dados inteligentes: Uma base de dados inclui fatos organizados de diversas maneiras para facilitar o seu armazenamento e seu acesso. Os paradigmas envolvidos em bases de dados vão desde os mais antigos, em que o armazenamento era hierárquico, até os mais recentes que envolvem bancos de dados distribuídos. Recentemente, surgiram pesquisas em banco de dados inteligentes. Este paradigma inclui algum tipo de ferramenta de IA para o seu funcionamento, no armazenamento ou na recuperação das informações.

- Problemas de decisão: um problema é caracterizado como problema de decisão quando existirem diversas ações que poderão ser tomadas para resolvê-lo, cabendo a quem deve resolvê-lo optar pela mais adequada.
- Diagnóstico: é o processo de encontrar um defeito em um sistema. Este tipo de aplicação foi popularizado pelo sistema especialista do MYCIN [86].
- Monitoração: é o processo de diagnosticar em tempo real. Normalmente, interpretam-se sinais vindos do mundo exterior e produz-se um alarme quando uma intervenção se faz necessária. No caso de gerência de redes, estas monitorações podem ser realizadas por *polling*, verificações periódicas de um determinado recurso que está sendo monitorado. Ou através de emissões de sinais, quando um evento ocorre. Este evento normalmente é configurado previamente pelo administrador da rede.
- Reparação: é o processo de corrigir um problema após ter sido detectado por um sistema de diagnóstico. Pode ser uma tarefa difícil porque implica em uma seqüência de ações, cada uma produzindo um certo efeito sobre o mundo exterior.
- Projeto e ou planejamento: projeto é a geração de especificações satisfazendo requisitos particulares. Planejamento é a descrição das ações que devem ser executadas para implementar um projeto. Na área de redes de computadores são comuns o projeto de implantação e o planejamento da rede de computadores. Várias questões são abordadas neste tipo de problema, uma das principais é a questão da escalabilidade devido à expansão dos sistemas e do número de usuários.

Durante mais de uma década as pesquisas na área de RNAs permaneceram nos laboratórios. Nesta época desconhecia-se a potencialidade das aplicações das RNAS, o que pode ser ilustrado por um texto de um livro de IA [17] em 1984, que afirmava que a modelagem neural e abordagem teórica de tomada de decisões possuíam sucesso limitado. Estas afirmações eram baseadas em diversas expectativas criadas em trabalhos anteriores que haviam falhado e havia um certo declínio nas pesquisas envolvendo estes paradigmas.

Das características que colaboraram com o sucesso da IAC, a que merece maior destaque é a capacidade de aprender. No entanto, pode-se ainda citar: a generalização, o comportamento emergente, capacidade de adaptação, a evolução de excitação em redes com ciclos, etc.

2.2 Redes Neurais Artificiais (RNAs)

As redes neurais artificiais (RNAs), ou sistemas conexionistas, foram inspiradas em sistemas biológicos onde uns grandes números de células nervosos funcionam individualmente de forma lenta e imperfeita. No entanto, coletivamente são capazes de realizar tarefas que muitos computadores não tem capacidade de fazê-las. São normalmente formadas por diversos processadores simples, interconectados com vários elementos de memória, cujos pesos das conexões são ajustados por experiência. Este ajuste de pesos caracteriza a capacidade de aprendizado das redes neurais. Existem atualmente, diversos tipos de redes neurais, tais como Percéptrons, Adalines, Redes de Hopfield, Mapas de Kohonen, BAM (*Birectional Associative Memories*), entre outras.

As RNAs, como mencionado anteriormente, possuem inspiração biológica muito forte. A maioria delas tentam simular o sistema biológico das células nervosas. Da mesma forma que no organismo humano, os neurônios artificiais realizam tarefas mais complexas atuando coletivamente.

Entre as características mais importantes destacam-se:

- Processamento e memória distribuída ao longo da estrutura da rede,
- Treinamento: normalmente, as redes são treinadas para realizar uma determinada tarefa, e não programadas;
- Interligação: os neurônios são amplamente interligados de forma que o estado de um neurônio afeta o potencial de um grande número de neurônios,
- Conexões: os pesos das conexões são adaptativos, se ajustam à medida que a rede aprende,
- Funções: as unidades de processamento contêm funções de ativação do tipo não linear;
- Generalização.

2.3 Modelos de Neurônio

É comum encontrar-se na literatura diversos tipos de modelos de neurônios artificiais. A Figura 2 ilustra algumas semelhanças entre os neurônios biológicos e os neurônios artificiais. Os dendritos

correspondem às entradas, a soma corresponde ao somatório e os axônios às saídas do neurônio artificial.

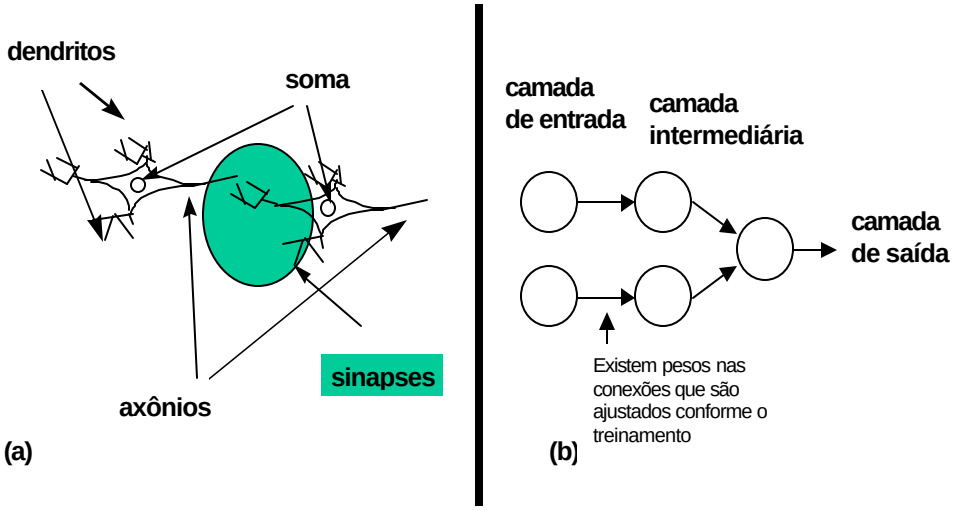


Figura 2. (a) Neurônios biológicos e (b) Neurônios artificiais

A maior diferença concentra-se nas sinapses, pois nos neurônios biológicos não existe uma ligação física (as trocas de informações são realizadas através de neurotransmissores), enquanto que nos neurônios artificiais existe uma ligação física e o armazenamento das informações depende dos pesos sinápticos. Assim como nos sistemas biológicos não é possível determinar exatamente em quais neurônios as informações estão armazenadas, elas encontram-se distribuídas ao longo da rede neural artificial.

O primeiro modelo de neurônio artificial foi apresentado por McCulloch e Pitts (1943) [73], e em 1959 foram apresentados simultaneamente, os Perceptrons por Rosemblatt e os Adalines por Widrow e Hoff [5], tanto os Perceptrons quanto os Adalines possuíam regras de aprendizado. Existe prova da convergência do algoritmo de aprendizado associado aos perceptrons. No Adaline é possível calcular pesos pelo método dos mínimos quadrados, apesar de ter sido apresentado com aprendizado iterativo.

A Figura 3 ilustra um modelo de neurônio artificial simples, composto de três entradas, denominadas x0, x1 e x2, em que x0 pode ser considerada uma polarização de valor 1. Aos

pesos sinápticos w0, w1 e w2, inicialmente são atribuídos valores aleatórios entre -0.1 e 0.1. Estes valores, normalmente são ajustados ao longo do treinamento. O somatório Ó calculado através da soma das entradas multiplicadas pelos pesos de cada uma irá ativar a saída.

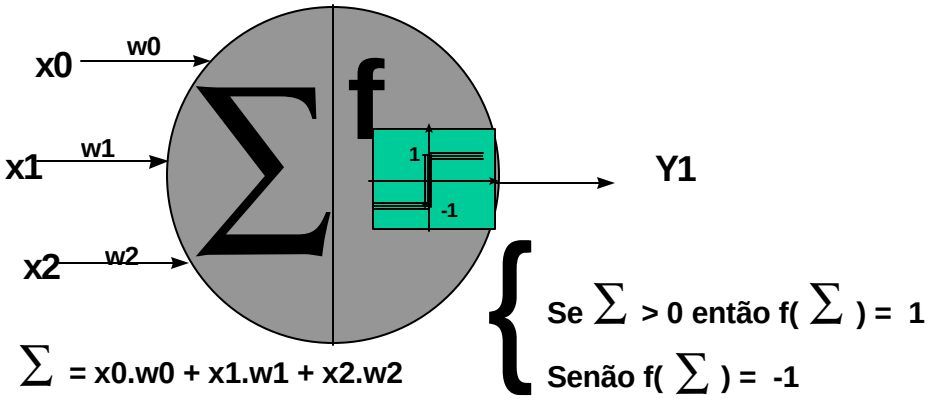


Figura 3. O neurônio artificial

A saída, denominada Y1, irá depender da função de ativação f, que no exemplo é uma função bipolar. Estas funções são determinadas no início de um projeto de redes neurais, e na maioria dos casos são utilizadas funções sigmoidais. Em uma definição formal, descrita por De Azevedo em [28], o autor apresenta o modelo geral de neurônio com base na Teoria de Sistemas e as principais funções de transferência utilizadas em RNAs.

2.4 Topologias das RNAs

As redes neurais artificiais podem ser subdivididas em redes diretas e redes recorrentes. As redes diretas são aquelas nas quais o fluxo das informações segue em uma única direção, enquanto que, as redes recorrentes (ou de realimentação) podem se realimentar de uma saída demonstrando troca de estado.

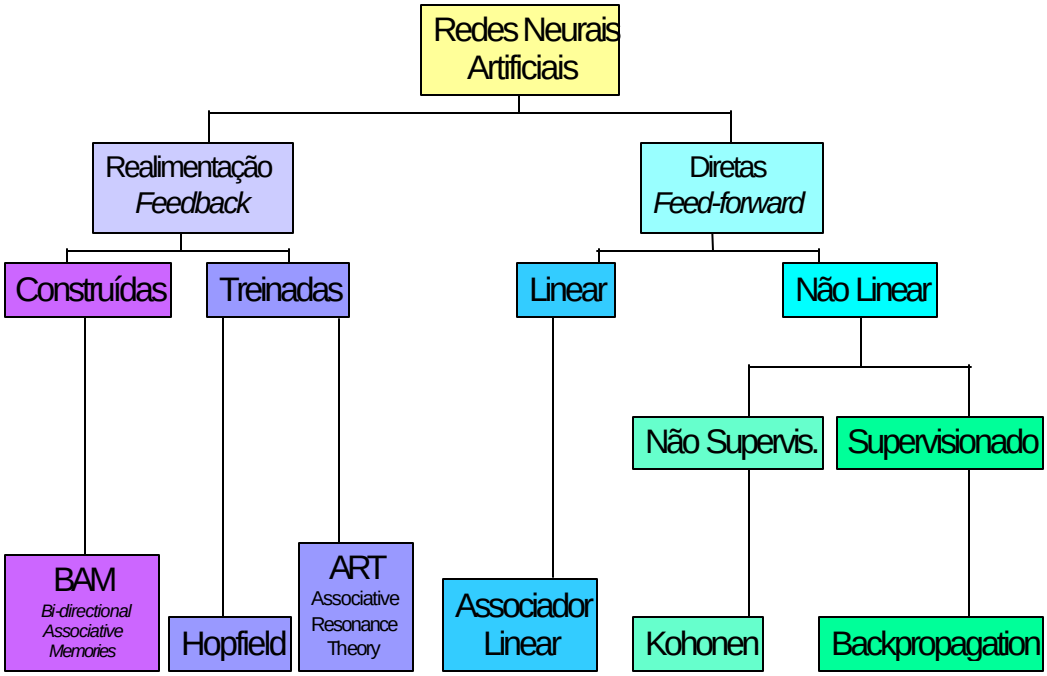


Figura 4. Classificação para RNAs

A Figura 4 [78] apresenta um diagrama que classifica as redes neurais conforme a sua topologia: realimentação (recorrentes) ou diretas. No caso de redes com realimentação (ou *feedback*) podem ainda ser subdivididas em construídas (por exemplo, redes do tipo BAM - *Bidirectional Associative Memories*) ou treinadas (Redes do tipo Hopfield ou do tipo ART - *Associative Resonance Theory* - por exemplo). As redes diretas podem ser subdivididas em redes lineares e não lineares. As redes não lineares podem ser subdivididas conforme o algoritmo de aprendizado: supervisionado e não supervisionado. O autor não mencionou em sua classificação, redes recorrentes adaptadas através de exemplos de entrada. Existem outras classificações que consideram o algoritmo de treinamento e o tipo de aprendizado, no entanto, possuem o inconveniente de que existem algoritmos de aprendizado que podem ser utilizados por várias topologias. Além disso, esta classificação foi aplicada à metodologia para o desenvolvimento de agentes. Na qual destaca-se o uso distinto de RNAs diretas e RNAs recorrentes na solução de problemas na área de gerência de redes.

2.5 Algoritmos de Treinamento

Os algoritmos de treinamento ou regras de aprendizado ditam como são feitos os ajustes dos pesos sinápticos para que a rede adquira experiência ao longo do treinamento.

Supervisionado

As regras ou algoritmos de aprendizado do tipo supervisionado atuam com o auxílio de um professor. Neste caso, a rede precisa conhecer o conjunto de entrada e o conjunto de saída. A saída obtida pela rede é calculada conforme a saída desejada, calcula-se um erro, que é utilizado para corrigir os pesos sinápticos.

Existem numerosas regras deste tipo, no entanto, a maioria são variações da regra de Hebb e da Regra Delta. Há mais de 30 anos atrás, Donald O. Hebb teorizou que a memória associativa biológica concentra-se nas conexões sinápticas das células nervosas, e que o processo de aprendizado e armazenamento de memória envolve trocas nas forças com as quais os sinais nervosos são transmitidos através das sinapses individuais. A primeira regra de Hebb diz que se um par de neurônios estão ativos simultaneamente existe uma troca dos pesos sinápticos, reforçando esta conexão [6].

Não Supervisionado

Regras de aprendizado do tipo não supervisionado atuam de forma competitiva ou auto-supervisionada. Para este tipo de regras são fornecidas as redes apenas o conjunto de entrada. Os neurônios da rede competem entre si fornecendo uma classificação como saída. Existe, além da competição, uma cooperação entre os neurônios da rede. Entre os algoritmos mais conhecidos, destacam-se: o vencedor leva tudo (*the winner takes all*) e a inibição lateral [6].

Por Reforço

O aprendizado por reforço (*reinforcement learning,* em inglês) é um método por tentativa e erro. Existe um índice de desempenho, chamado de sinal de reforço, que é utilizado para otimização. Este paradigma de aprendizado tem profunda inspiração biológica em que os

comportamentos satisfatórios são reforçados e os insatisfatórios geram alterações nos valores correspondentes às conexões [6].

Aprendizado em RNAs Recorrentes

Em alguns casos, os valores de entrada e saída dos sistemas dinâmicos são conhecidos, no entanto, o conjunto de estados é desconhecido. Neste caso, a RNA recorrente tenta estimar o estado com base no conjunto de valores de entrada e saída apresentados durante o treinamento. O objetivo do aprendizado é encontrar uma regra para ajuste dos pesos durante o treinamento [6][87].

Roisenberg [87] descreveu exemplos de aprendizado em RNAs recorrentes e apresentou um algoritmo baseado em retropropagação quando o estado da rede é conhecido. A rede é vista como uma Máquina de Estados Finitos (MEF). Utilizando a equivalência, ele transformou a rede neural recorrente em uma rede direta correspondente, posteriormente o algoritmo aplica a retropropagação. Durante o treinamento, a linha de atraso entre os neurônios de saída dos estados da MEF e os neurônios da camada intermediária foram desligadas, obtendo uma rede direta com neurônios de entrada que correspondem aos sensores no tempo k e neurônios de saída que correspondem a estados da MEF no tempo k. Também foram utilizados neurônios de entrada que correspondem a entradas de sensores, a seqüência temporal de todas as possíveis entradas da MEF, de tal forma que passe por todos os estados possíveis da MEF. Então, apresenta-se a rede os dados de entrada u(k) e o estado x(k) da MEF, e propagou-se na rede, obtendo o próximo estado calculado x(k+1). Depois desta etapa é possível aplicar o algoritmo de retropropagação tradicional, calculando o erro a partir do estado desejado da MEF x'(k+1).

Outra possibilidade testada neste trabalho é o aprendizado através de um conjunto de exemplos. É fornecido a RNA recorrente um conjunto de exemplos de entrada. Através de um algoritmo de adaptação, a rede neural vai ajustando as saídas, bias e pesos conforme cada entrada apresentada a RNA recorrente.

2.6 RNAs Diretas versus RNAs Recorrentes

O objetivo desta comparação é apresentar as vantagens do uso de redes dinâmicas. Talvez por falta de informação ou porque as redes diretas são mais divulgadas, o número de trabalhos que utilizam redes estáticas para solução de problemas dinâmicos é muito grande. A verdade é que as redes diretas podem funcionar para um caso particular do problema, mas não para todas as instâncias.

A Tabela 1, apresenta um quadro comparativo entre as redes neurais diretas e as redes recorrentes.

Tabela 1. Principais diferenças entre RNAs diretas e RNAs recorrentes

Características	RNAs Diretas	RNAs Recorrentes
Quanto ao estado	Estáticas – não existe troca de estados.	Dinâmicas – existe troca de estados conforme o tempo.
Quanto à topologia	Não possui ciclos. O fluxo dos dados possui apenas uma direção, da entrada para a saída.	Possui ciclos com realimentação. A saída de uma das camadas pode realimentar a entrada de dados (o fluxo pode ser da entrada para a saída, da camada intermediária para a entrada,da camada intermediária para intermediária, da saída para camada intermediária ou da camada de saída para a entrada).
Quanto ao número de camadas	Para solucionar um problema **LINEAR**: duas camadas, uma de entrada e uma de saída. **NÃO LINEAR**: tem que ter no mínimo uma camada intermediária.	Como existem ciclos com realimentação, pode haver ou não camada intermediária, depende da ordem do predicado.
Quanto à sua construção	**Treinadas** ou de forma **supervisionada** (como é o caso do algoritmo de treinamento Backpropagation) ou **não-**supervisionada (como é o caso dos mapas de Kohonen).	**Construídas** : exemplos de redes neurais construídas são BAM (*Bidirectional Associative Memory* - Kosko) e Hopfield . **Treinadas** : o ajuste dos pesos deve ser feito nos dois sentidos para frente e para trás. Exemplo deste tipo de rede ART (*Adaptative Resonance Theory* – Grossberg & Carpenter). **Adaptadas através de exemplos** : redes com retardos que aprendem através de exemplos de entrada (exemplo no Capítulo 6)

2.7 RNAs Recorrentes

É característica das RNAs recorrentes possuirem o grafo de sua topologia com ciclos e trabalharem com o tempo discreto e síncrono. Existem, no entanto, redes com ciclos que não devem ser chamadas de recorrentes, como é o caso de dinâmica em que o neurônio é representado por uma equação diferencial [6][87]. É possível encontrar uma rede recursiva a partir de um autômato finito definido por sua equação matemática (Figura 5):

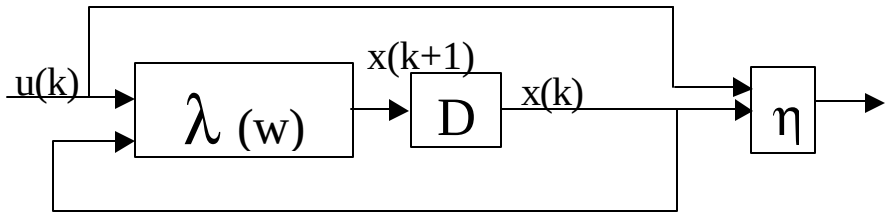


Figura 5. Implementação neural de autômato finito

$S=\{U, Y, X, x_0, \lambda, \eta\}$

Onde:

U é um conjunto finito de entradas;

Y é um conjunto finito de saídas;

X é um conjunto de estados ou espaço de estado;

$x_0 \in X$ é o estado inicial;

$\lambda : U \times X \rightarrow X$ é a função de transição de estado;

$\eta : U \times X \rightarrow Y$ é a função saída.$x(k+1) = \lambda(x(k), u(k), W)$

$y(k) = \eta(x(k), u(k))$

A Figura 5 apresenta a implementação das equações acima com uma camada de neurônios munidos de retardo (transformando k+1 em k) implementando (ou aproximando) a função λ que tem por entradas x(k) e u(k). Um segundo conjunto de neurônios implementa a função η (podendo ser neurônios estáticos, portanto sem retardos).

2.8 Computabilidade e Complexidade de RNAS

Existem duas abordagens para o estudo de complexidade. Uma clássica e, portanto, bem explorada, que estuda a complexidade dos algoritmos de treinamento de RNAs (colocar referências da Teresa). A outra abordagem estuda a complexidade sobre as RNAs na solução de problemas complexos. Esta segunda foi apresentada por Barreto [6] e estuda a capacidade das RNAS em solucionar problemas complexos.

Para Barreto [6], o sucesso das RNAs faz crer que um computador utilizando estas redes como bloco básico, possa resolver problemas que computadores que não usem esta tecnologia são incapazes, ou ao menos, teriam muita dificuldade para resolver. Considerando o computador como uma máquina de resolver problemas. Ao fato que um problema possa ser resolvido por recursos finitos chama-se *computabilidade* e a quantidade de recursos envolvidos chama-se *complexidade*. Goldschlager apresenta estes conceitos de forma detalhada em [47].

Para verificar se um problema é computável e qual a complexidade da solução em um computador baseado em instrução e um neurocomputador, é necessário esclarecer como cada uma destas máquinas deverá soluciona o problema abordado.

Tabela 2. Computador baseado em instrução x neurocomputador

COMPUTADOR BASEADO EM INSTRUÇÃO (CBI)	NEUROCOMPUTADOR (CBR)
O computador virtual (circuitos e programas)	A rede de neurônios com entradas e saídas (simulado ou materialmente implementado)
O ato de fazer o computador apto a resolver um problema específico (carregar o programa no computador)	Um meio de fixar os pesos das conexões muitas vezes, usando um algoritmo de aprendizagem (equivalente a carregar o programa).
O programa será compilado, interpretado e então executado para solucionar o problema	Usar a rede já treinada para resolver o problema com os dados a serem usados na entrada da rede (equivalente a executar um programa)

A computabilidade de um problema dependerá dos dois primeiros pontos. No caso de um CBI, a possibilidade de resolver um problema depende do material que se dispõe, ou seja, se existe um programa munido de instruções que satisfaçam a solução daquele determinado problema. No caso de um neurocomputador seria necessário o conjunto de pesos sinápticos capazes de solucionar o problema em questão. A complexidade do problema irá depender

exclusivamente do terceiro ponto, executar o programa ou excitar a rede neural com os dados a serem utilizados.

2.9 Equivalência de Computabilidade

Quando a solução de um problema é abordada por um CBI, isto corresponde à existência de um algoritmo, composto de uma seqüência finita de instruções, capaz de solucionar tal problema. No entanto, um computador construído com redes neurais é equivalente a uma Máquina de Turing, e em conseqüência, tem capacidade para resolver qualquer problema computável e somente eles, tem despertado grande interesse científico. Esta equivalência foi inicialmente provada por McCulloch e Pitts [73] usando manipulações de lógica e por Arbib [5] com argumento bastante intuitivo.

Mas se as redes neurais não resolvem problemas que antes não se sabia resolver, qual seu interesse? A resposta vem da facilidade de resolver problemas e dos recursos necessários. A isto se chama complexidade. Quando se usa um computador digital esta complexidade é geralmente expressa em quantidade de memória e tempo. Quando se usa a abordagem conexionista deve-se se falar em topologia da rede (número de neurônios e como são ligados) e como são os neurônios.

2.10 Complexidade de RNAS

Os aspectos mais relevantes ao estudo da complexidade de redes neurais são:

1. Dado um problema, definir a topologia da rede necessária para a sua solução. Por exemplo, se a rede deve agir como um modelo de um sistema de controle adaptativo, e a entrada da rede é a saída do processo, será possível utilizar uma rede neural direta?

2. Dado um problema que pode ser resolvido por uma determinada topologia de RNA, e uma precisão desejada, qual o tamanho mínimo da rede que deve ser utilizado?

Existe ainda o conceito de RNA equivalentes em termos de complexidade. Duas redes são ditas equivalentes, em termos de complexidade, quando ambas são capazes de resolver o mesmo problema.

Apesar dos estudos de Minsky e Papert, relatados em [74], prejudicarem o andamento das pesquisas sobre IAC. O livro constitui a primeira contribuição ao estudo da complexidade de problemas resolvidos por RNAs. A classificação dos problemas em: linearmente separáveis; e, não linearmente separáveis. Se um problema é linearmente separável ele pode ser resolvido com uma camada de entrada e saída de neurônios, como é o caso dos circuitos lógicos E, OU e Não. Se o problema é não linearmente separável ele precisa de uma camada intermediária de neurônios entre a entrada e a saída da rede, como é o caso do OU-Exclusivo e da paridade de uma seqüência de 0 e 1.

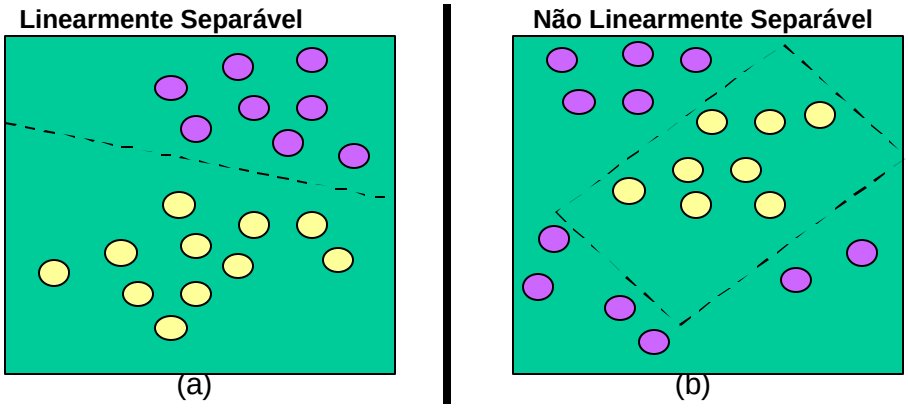


Figura 6. Problemas linear e não-linearmente separáveis

2.11 Alguns Teoremas

Os seguintes teoremas foram utilizados no desenvolvimento do trabalho. As comprovações destes teoremas foram apresentadas por Barreto em [8], estão relacionados a complexidade de redes neurais artificiais.

- **Teorema I**: Toda RNA constituída apenas de neurônios estáticos, incluindo ciclos (ou seja, com retroação), é equivalente a uma outra rede estática sem ciclos.

- **Corolário I**: todo problema que possa ser resolvido por uma rede com neurônios estáticos contendo retroações, pode ser resolvido por uma rede, sem retroações.
- **Teorema II**: Todo autômato finito pode ser realizado por uma RNA com neurônios estáticos e um conjunto de retardos.
- **Teorema III**: Toda rede direta, com topologia por camadas, com neurônios lineares é equivalente a uma rede linear contendo apenas duas camadas de neurônios separadas por uma camada de conexões.
- **Teorema IV**: Uma rede estática, direta, usada isoladamente, é incapaz de aproximar um sistema dinâmico.
- **Teorema V**: uma rede direta munida de um conjunto de retardos é capaz de aproximar um sistema dinâmico. Segundo Roisenberg [87], não basta somente uma rede neural com retardo mas também precisa de uma camada intermediária de neurônios.

2.12 Teoria da Complexidade Conexionista

Quando um problema é tratado por um CBI, é costume considerar-se problemas de complexidade linear, polinomial e NP.

- Problemas de complexidade linear: são problemas em que a quantidade de recursos cresce linearmente com o número de dados. Como por exemplo, o tempo gasto para o planejamento de uma rede. Este tempo será proporcional a quantidade de recursos que se deseja acrescentar nesta rede. Quanto maior for o número de elementos da rede maior será o tempo gasto no seu planejamento.
- Problema polinomial: é um problema semelhante à inversão de matrizes em que o número de multiplicações mínimas é perto de n^3 onde n é a ordem da matriz.
- Problemas NP: são problemas que não se conhece até o momento algoritmo que exija esforço polinomial, a quantidade de recursos cresce exponencialmente.

A classificação para problemas tratados por neurocomputadores segue uma outra linha de avaliação:

- Problemas estáticos linearmente separáveis: problemas envolvendo a implementação de uma função (por ser um problema estático) e que podem ser resolvidos por um percéptron de uma camada de conexões.
- Problemas estáticos não-linearmente separáveis: problemas envolvendo a implementação de uma função (por ser um problema estático) e que podem ser resolvidos por uma rede direta, com neurônios estáticos, exigindo ao menos uma camada de neurônios internos.
- Problemas dinâmicos com dinâmica finita: os problemas com dinâmica finita são aqueles que a duração da resposta do sistema após a entrada dura um tempo finito. Um exemplo é os filtros FIR (*Finite Impulse Response*). Estes problemas podem ser resolvidos por redes diretas com neurônios dinâmicos.
- Problemas dinâmicos com dinâmica infinita: os problemas com dinâmica infinita são aqueles que a duração da resposta do sistema após uma entrada pode durar um tempo infinito. Como por exemplo, os filtros IIR (*Infinite Impulse Response*). Estes problemas devem ser abordados com redes com retroação e com neurônios ou redes estática e conjunto de retardos. Neste caso, o problema da estabilidade da rede, se a rede encontrará ou não solução, e quanto tempo será necessário são questões em aberto.

Capítulo 3

IAC aplicada à Gerência de Redes

3.1 Introdução

Gerência de redes é o processo de controla-se uma rede de dados complexa com o objetivo de aumentar sua eficiência e produtividade. Rede de dados pode ser definida como uma coleção de circuitos e dispositivos utilizados para a transferência de informações de um computador a outro que permite a usuários de diferentes localidades compartilhar recursos de outros computadores, sem necessitar conhecer o espaço físico onde eles estão situados [63]. O processo para gerenciar uma rede de dados ainda depende muito de um administrador de redes. Este trabalho aumenta conforme a expansão da rede, devido ao escopo e a complexidade. Características como heterogeneidade, complexidade, dinamismo e a natureza distribuída, dificultam o gerenciamento das redes de dados, principalmente, a implantação de um gerenciamento automatizado. A linguagem JAVA tem sido usada para contornar problemas de heterogeneidade e de distribuição de objetos porque é independente de plataforma. No entanto, o uso destas novas ferramentas não é o suficiente para alcançar a automatização do processo. Existe ainda a complexidade e o dinamismo dos sistemas, que não são solucionados através de ferramentas comuns.

No capítulo anterior foram apresentadas algumas das aplicações da área de IA que podem ser relacionadas com a gerência de redes. No caso de bases de dados inteligentes, existe a possibilidade de aplicação de técnicas de IA para a recuperação e armazenamento dos objetos de gerência armazenados numa MIB. Em gerência de redes, é característica do administrador da rede tomar as decisões com relação às ações para solucionar problemas que

tenham acontecido. Para automatizar este processo, poderiam ser aplicadas técnicas de IA para auxiliar na tomada de decisões.

Aplicações de diagnóstico em gerência de redes podem ser comprovadas por experimentos anteriores a este trabalho. Em [32][33], foi testado o desenvolvimento de módulos para o diagnóstico de falhas utilizando regras de produção. Uma novidade seria a utilização de técnicas conexionistas para sistemas de diagnósticos em gerência de redes. Monitorações são atividades freqüentes nesta área, normalmente são realizadas por *polling*, que são verificações periódicas de um determinado recurso ou objeto da rede. Outra possibilidade é utilizar emissões de sinais quando um evento ocorre (alarmes e mensagens de *trap*). Este evento normalmente é configurado previamente pelo administrador da rede. A reparação seria a tomada de decisões indicando uma série de ações a serem realizadas para a recuperação de um determinado sistema. E finalmente, as aplicações de projeto e planejamento na área de redes de computadores, em que a escalabilidade devido à expansão dos sistemas de redes e do número de usuários seja considerado.

Porém o que caracteriza o estado-da-arte da aplicação de técnicas de IA a gerência de redes é apossibilidade de se extrair o conhecimento do próprio ambiente de rede. Desta forma, é possível projetar sistemas que se tornem independentes da configuração do administrador do sistema. Desta forma a atividade de gerência não dependeria exclusivamente da mão de obra altamente especializada do administrador da rede. A idéia de extrair o conhecimento visando a automação dos sistemas de gerência de redes pretende minimizar o trabalho dos administradores de redes.

Atualmente as atividades de tomada de decisões, recuperação de falhas da rede, definição de limites para a monitoração do sistema, projeto e planejamento dependem da intervenção do administrador do sistema.

3.2 Gerência de Sistemas

Como gerência é um processo complexo e distribuído, alguns autores definem divisões no sentido de viabilizar o desenvolvimento de atividades de gerência de uma forma organizada. Para Nogueira [79], por exemplo, a área de gerência de redes pode ser dividida em gerência de

sistemas e quanto à funcionalidade do que é gerenciável, que será apresentada na Seção 3.4. A gerência de sistemas é definida em níveis como: gerência de elementos, em que são focalizados os elementos constituintes de uma rede, tais como: estações de trabalho, centrais de comutação, multiplexadores, etc. O gerenciamento em nível de redes, onde são enfocados conjuntos de elementos e sua interconexão, tais como uma rede local, uma rede de longa distância, uma sub-rede de comunicação. O gerenciamento em nível de serviços, em que são focalizados aspectos do serviço prestados aos usuários ou clientes, tais como, a qualidade do serviço e o relacionamento com o cliente. E finalmente, o gerenciamento no nível de negócios, em que a são focalizados aspectos do negócio, do ponto de vista do proprietário dos sistemas. Inclui-se aí o planejamento dos serviços, estratégias de mercado, diretrizes do crescimento, etc.

Quanto à funcionalidade do que é gerenciável, a ISO subdividiu a gerência de redes em cinco áreas – modelo OSI FCAPS: gerência de falhas, gerência de configuração, gerência de desempenho, gerência de contabilização e gerência de segurança [43][75].

3.3 Áreas Funcionais

A gerência de falhas é o processo de localizar problemas, ou falhas, em uma rede de dados. Envolve as tarefas de descobrir o problema, isolá-lo e solucioná-lo quando possível. Entre as causas mais prováveis para falhas em uma rede estão: erros de projeto e implementação da rede, erros de sobrecarga, distúrbios externos, tempo de vida útil de equipamentos expirado e má implementação de softwares (famosos *bugs*). Uma gerência de falhas bem projetada aumenta a confiabilidade na rede, fornecendo ferramentas ao administrador da rede que auxiliem a detectar os problemas e iniciar os procedimentos de recuperação [25][63][83].

Existem dispositivos que controlam o comportamento dos dados que trafegam pela rede. A gerência de configuração é o processo que determina e configura estes dispositivos críticos. Roteadores, pontes (*bridges*, em inglês), terminais e servidores são exemplos destes dispositivos. A gerência de configuração auxilia a localizar quais os softwares e versões estão dispostos em cada equipamento [25][63][83].

A gerência de desempenho deve assegurar que a rede tenha capacidade para suportar e acomodar uma certa quantidade de usuários. Ou seja, ela é extremamente necessária para

otimizar a Qualidade do Serviço. Este processo mede o desempenho dos equipamentos e softwares disponíveis através de registros de algumas taxas de medidas. Exemplos destas taxas são vazão (*throughput*), taxas de erros, taxas de utilização e tempo de resposta [25][63][83].

Determinar quais os recursos e a forma que estão sendo utilizados pelos usuários é tarefa da gerência de contabilização. Além disso, este processo auxilia a assegurar que os usuários tenham acesso à quantidade suficiente dos recursos disponíveis. Envolve também, garantir ou remover permissões de acesso à rede [63][83]. A gerência de segurança é o processo que controla o acesso às informações disponíveis na rede. Existem informações armazenadas em computadores ligados à rede que são impróprias a todos os usuários. O grupo de informações mais conhecido em que não fica disponível, para evitar ações impróprias que prejudiquem os usuários, é o conjunto de senhas que permitem o acesso à rede. A gerência de segurança permite que o administrador monitore as tentativas de invasão na rede [63][83].

O *International Engineering Consortium WebPro Forum Tutorial* apresenta as principais características de gerência que devem ser suportadas por cada uma das áreas do OSI FCAPS (*Fault Configuration Account Performance Security*). Um resumo das principais funções de cada uma das áreas das FCAPS, estão dispostas na Tabela 3 e foram obtidas em [43].

Tabela 3. Principais funções das FCAPS

Falhas	**Configuração**	**Contabilização**	**Desempenho**	**Segurança**
Detecção de falha	Inicialização de recursos	Utilização de recursos e serviços	Taxas de utilização e erros	Acesso seletivo de recursos
Correção da falhas	Atualização da rede	Custo dos serviços	Desempenho consistente	Habilitar funções dos nós
Isolamento da falha	Detecção automática de equipamentos	Limite de contabilização	Coleta de dados de desempenho	Logs de acesso
Recuperação da rede	Tratamento de Base de Dados	Combinar custos para múltiplos recursos	Geração de relatórios de desempenho	Relatórios de eventos e alarmes de segurança
Tratamento de alarmes	Shut down de recursos	Determinação de cotas para uso	Análise de dados de desempenho	Privacidade dos dados
Filtragem de alarmes	Gerência de mudanças	Auditoria	Relatórios de problemas	Verificação de direitos de acesso dos usuários
Geração de alarmes	Auxílio para pré-atualização	Relatórios de fraudes de contas	Planejamento de capacidade	Precaução contra ataques e tentativas de invasão

Correlação de alarmes	Gerência de inventário	Auxílio para modos diferentes de contabilização	Coleção de dados/ estatísticas de desempenho	Log de auditoria e segurança
Teste de diagnóstico e tratamento de erros, logs de erros e estatísticas	Configuração de cópias e configuração remota de recursos		Manutenção e verificação de logs sobre históricos de desempenho	Distribuição de informações relacionadas a segurança

3.4 Gerência Internet

Com o crescimento da Internet em meio à década passada surgiu a necessidade de estruturar e padronizar um gerenciamento apropriado. Em 1987, surgiram três propostas, das quais duas sobreviveram. Uma delas foi o *Simple Network Management Protocol*, ou SNMP como é mais conhecido. E a outra é o *Common Management Over TCP/IP* ou CMOT. O CMOT foi uma tentativa de usar padrões de gerência OSI em ambientes Internet. Problemas como demora de implementações e experiências operacionais com este protocolo não foram relatadas [91].

No SNMP, um gerente pode controlar muitos agentes. O protocolo é construído sobre o UDP, que é o protocolo de transporte não orientado à conexão do ambiente Internet. A informação é armazenada em PDUs (*Protocol Data Units*) do SNMP codificadas de acordo com a linguagem ASN.1 diretamente sobre o protocolo de transporte. Os cinco tipos de PDUs são: *GetRequest, GetNextRequest, SetRequest, Response* e *Trap* [22][23][91].

Como o SNMP utiliza um serviço de transporte não confiável, as operações entre gerente e agente são realizadas sem confirmação. Como o serviço é não confiável deve-se ter certeza que a mensagem chegou ao destinatário através de um reconhecimento. Se o reconhecimento não chega durante um certo período (*time-out*) a mensagem deve ser retransmitida [63][83].

3.4.1 SNMPv2 e SNMPv3

Desde a publicação original do SNMP surgiram outras propostas para melhorá-lo. Em 1992, um grupo de pesquisa resolveu acatar algumas dessas propostas e produzir um novo padrão. Entre estas melhorias destacam-se: maior segurança, a possibilidade de construir uma hierarquia de gerentes e uma nova primitiva que permite o resgate de um grupo de informações.

Recentemente, foi lançada a versão três (SNMPv3) com recursos de segurança e a capacidade de configuração remota [63][83][85][91].

3.4.2 Elementos da arquitetura SNMP

Os sistemas utilizados para gerência procuram realizar os serviços sem sobrecarregar as entidades gerenciadas ou os canais de comunicação. Os elementos comuns de se encontrar em um sistema de gerência são [3][91]:

- **Agentes:** são os elementos de rede gerenciáveis, isto é, aqueles computadores, pontes, roteadores, etc., que possuem o protocolo SNMP embutido. Quando um elemento de rede não possui tal protocolo, utiliza-se agente procurador ("proxy", em inglês), que fornece funções de conversão de protocolo. A Seção 5.3 apresentará mais características dos agentes de gerência considerando que não utilizam técnicas de IA.
- **Gerentes:** são as estações de gerência, responsáveis por executar as funções de gerenciamento. São responsáveis por relatórios sobre operações e informações dos objetos gerenciáveis. Estas informações são repassadas ao gerente humano através de algum tipo de interface.
- **Base de Dados (MIB)**: é uma coleção de objetos gerenciáveis mantidos pelos agentes. Os objetos definem as diferentes características dos dispositivos de rede que estão sendo gerenciados.
- **Protocolo:** para comunicação entre as entidades é utilizado um protocolo do tipo pergunta/resposta que é utilizado para trocar informações entre os agentes e gerentes.
- **Autenticação:** é o meio de segurança pelo qual o agente SNMP valida as requisições de uma estação de gerência antes de respondê-las.

A Figura 7 ilustra como seria o relacionamento entre os elementos de gerência de redes utilizando o protocolo de comunicação SNMP.

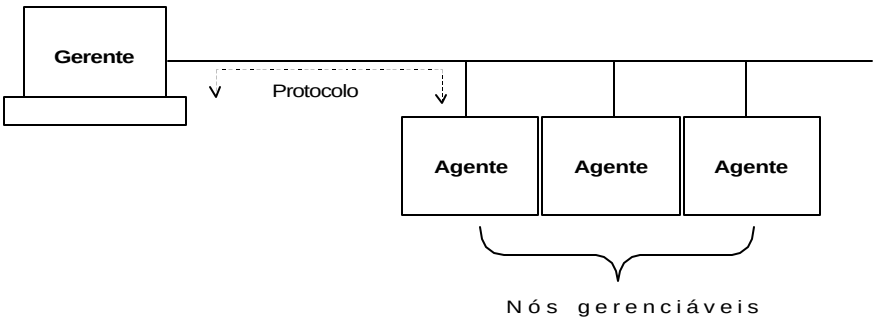


Figura 7. Relacionamento gerente-agentes utilizando o protocolo SNMP

3.4.3 Gerência OSI X Gerência Internet

Para Oates [80], os protocolos CMIP e SNMP simplesmente fornecem mecanismos, independentes de fabricantes, para especificar e manipular o conteúdo de bases de informação de gerência. Existem questões difíceis e importantes que não são abordadas por estes protocolos. Quais informações são mais relevantes na tarefa de identificação de falhas, por exemplo. Como decidir a partir das informações se existe uma falha, quantas falhas existem e se o problema é temporário ou persistente. Qual nível de síntese e abstração dos dados disponíveis é mais apropriado. As decisões de gerência deveriam ser feitas localmente com informações locais por componentes específicos da rede, ou deveriam ser feitos globalmente com informações no nível de rede. Com sua abordagem em identificação de falhas o autor tenta responder estas questões. Talvez estas dúvidas tenham motivado fabricantes como a Sun e a IBM a desenvolver pacotes baseados na linguagem JAVA para auxiliar a criação de agentes inteligentes para gerência de redes [12][58]. Estes pacotes serão devidamente discutidos no Capítulo 5. Leiwand [63] complementa que os protocolos simplesmente oferecem os métodos para monitorar e configurar dispositivos de redes. O desafio de analisar as informações contidas nos protocolos é tarefa das aplicações.

3.5 Comportamentos da Gerência

Existem dois comportamentos que podem ser assumidos no gerencimento de uma rede de computadores: a gerência reativa e a gerência pró-ativa. A gerência reativa é aquela tradicional, o problema ocorre primeiro e depois será tratado. Como desvantagem tem-se a queda do sistema por um determinado período de tempo até o que problema seja encontrado e resolvido. O estado da rede passa a ser inativo até que a falha seja identificada, isolada e solucionada, como ilustra o esquema da Figura 8.

O outro comportamento existente é o pró-ativo. A gerência pró-ativa é aquela capaz de localizar os problemas antes que eles aconteçam, sejam eles de desempenho, de falhas, de configuração, de contabilização e de segurança [57]. O esquema apresentado na Figura 8 ilustra que a atividades de gerência pró-ativa devem ser realizadas enquanto o estado da rede ainda está ativo. Este comportamento é uma forma de prevenção de possíveis problemas que possam degradar a rede de computadores.

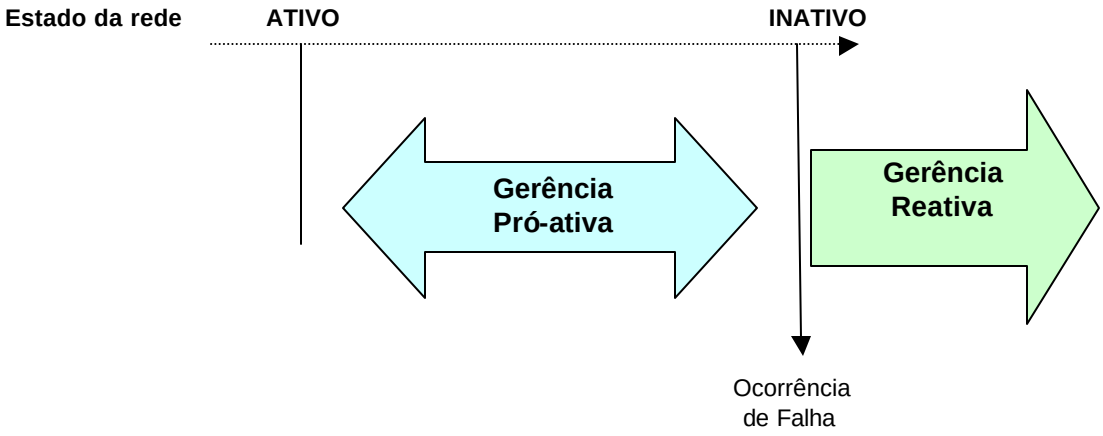


Figura 8. Comportamento da gerência de redes

Em [30], foi apresentado um modelo para o desenvolvimento da gerência pró-ativa de redes. Neste trabalho, baseado em atividades práticas e teóricas, o autor descreveu um modelo para a gerência pró-ativa contendo as seguintes entidades com o objetivo de realizar um gerenciameno preventivo:

- o **Módulo de Gerência Pró-ativa** é responsável por monitorar e analisar a rede, com o objetivo de verificar as tendências de degradação da rede, conforme a aplicação especificada. Se constatada alguma tendência, deve enviar notificações para a Plataforma de Gerência. Os componentes são a *baseline*, um monitor de rede e um serviço de verificação.
- a ***Baseline*** é um registro do comportamento normal da rede, freqüentemente consultada pelo serviço de verificação;
- o **Monitor de Rede** ou **Agente Remoto** coleta as amostras de objetos gerenciados das MIBs em questão;
- o **Serviço de Verificação** é responsável por verificar as tendências de degradação da rede, por meio de comparações entre os valores armazenados na *Baseline* e os valores coletados, constantemente, pelo Agente ou Monitor Remoto;
- a **Plataforma de Gerência** é responsável por receber as notificações e realizar as ações corretivas para impedir a degradação da rede. Atualmente, as ações corretivas estão sendo determinadas pelo gerente (ou administrador da rede).
- o **Serviço de Comunicação** é responsável por interconectar o Módulo de Gerência Pró-ativa e a Plataforma de Gerência. Este serviço pode utilizar um protocolo de transporte ou o protocolo de gerência SNMP.

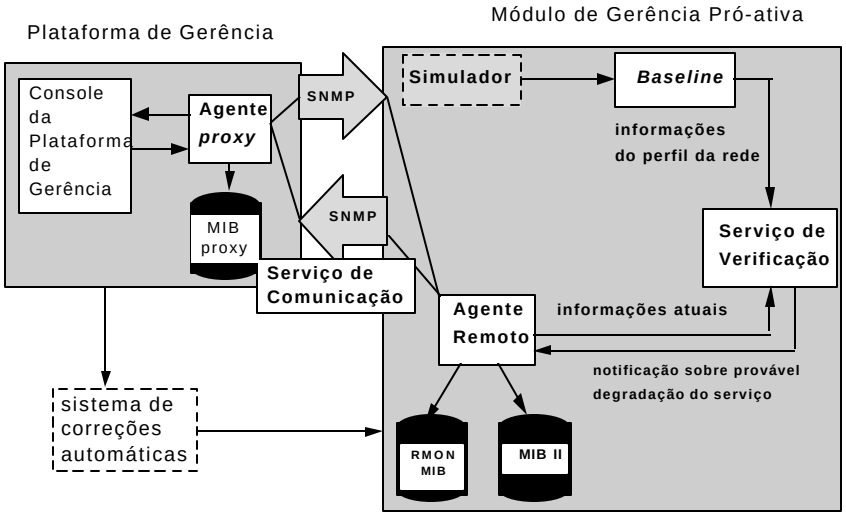


Figura 9. Modelo para a gerência pró-ativa de redes [30]

Quando este modelo foi descrito, a gerência de redes ainda possuía caráter

proprietário. Ou seja, utilizava-se uma plataforma de um determinado fabricante para fazer o controle e monitoração das informações de gerência. Estas plataformas foram sendo substituídas pela utilização de aplicações de objetos distribuídos utilizando a plataforma CORBA (ou outras plataformas, tais como: OLE, ActiveX, COM, DCOM, etc) para desempenhar as atividades de gerência.

Atualmente, existem sistemas multi-agentes que são compostos por uma coleção de entidades de software que realizam ações de forma autônoma. Os agentes diferem dos objetos por possuir estados internos mais complexos do que os objetos distribuídos. Os agentes possuem a capacidade para decidir o que fazer e quando fazer. Eles podem dizer "não" quando forem solicitados para realizar uma determinada tarefa. Ao contrário, os objetos possuem regras fixas. Agentes podem ser programados para alterar as regras dinamicamente, conforme a aplicação que estão executando. Em uma aplicação de objeto distribuído, cada objeto distribuído contribui com uma pequena parte da função para alcançar o objetivo único da aplicação. Em um sistema de agentes, existe uma coleção de entidades com objetivos distintos que colaboram para a realização do objetivo da aplicação. Objetos invocam métodos enquanto agentes trocam mensagens. Objetos são definidos como pacotes de dados esperando ser ativados por algum processo enquanto agentes decidem quando devem ser ativados [12].

3.6 Aplicação da IAC

As primeiras aplicações de IA na área de gerência de redes foram os sistemas especialistas (SE), abordagem simbólica [63]. Um SE recebia como entrada uma situação atual, avaliava os dados segundo fontes fornecidas ao sistema e por fim sugeria uma solução ao problema. Isto era realizado através de um conjunto de regras *if-then* consideradas seqüencialmente. A principal desvantagem era o tempo gasto para executar um simples teste e delinear as soluções. Além disso, segundo a complexidade e dinamismo das redes de dados, o conhecimento adquirido pelos SE torna-se frágil à medida que o estado da rede evolui. Portanto, sistemas especialistas são limitados pelo gargalo da aquisição de conhecimento e fragilidade das informações.

Outro aspecto que não pode deixar de ser mencionado é a quantidade de informações que circulam em uma rede de dados, neste caso sistemas de raciocínio baseados em IAC são capazes de manipular estas informações com maior rapidez do que pensadores humanos.

3.7 Utilização de RNAs Recorrentes

No comportamento pró-ativo da gerência de redes devem ser considerados o dinamismo com que as redes de dados operam e a quantidade de informações manipuladas. Na seção anterior foram mencionados os componentes para realizar um gerenciamento pró-ativo, definidos em [30], Figura 9. Naquela época foi definida a utilização de um simulador para gerar informações relevantes sobre os aspectos operacionais e gerenciais da rede. Através do uso da simulação era possível determinar as ações de controle a serem realizadas e delinear uma *baseline* ou perfil da rede, estatisticamente correta.

No presente trabalho, foram realizados experimentos com redes recorrentes com aprendizado por exemplos para simular o perfil de utilização de uma rede de computadores. 'E poss'ivel substituir a utilização da simulação pela aplicação de técnicas de IA no modelo proposto anteriormente. A aplicação de desenvolvimento de *baseline* foi escolhida porque agrega características dinâmicas e envolve grandes quantidades de dados. Tenta-se extrair o conhecimento sobre a rede a partir da própria rede.

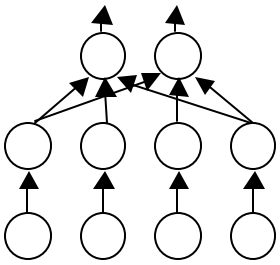
3.7.1 Redes diretas versus redes dinâmicas

A gerência de falhas é a área de gerência de redes em que têm sido mais freqüente o uso de técnicas de IA, na correlação de alarmes. Um dos grandes problemas no gerenciamento de falhas de redes de computadores e de telecomunicações é o número de alarmes gerados por uma determinada falha. Como por exemplo, a necessidade de correlação de alarmes em redes de telefonia celular é justificada pelo grande número de alarmes que podem ser emitidos. Segundo o trabalho apresentado em [103], cada elemento de uma rede de telefonia celular gera vários alarmes quando um enlace falha. Para a modelagem da correlação de alarmes os autores utilizaram uma rede direta (tipo *feedforward*), onde a entrada da rede é um vetor dos alarmes de

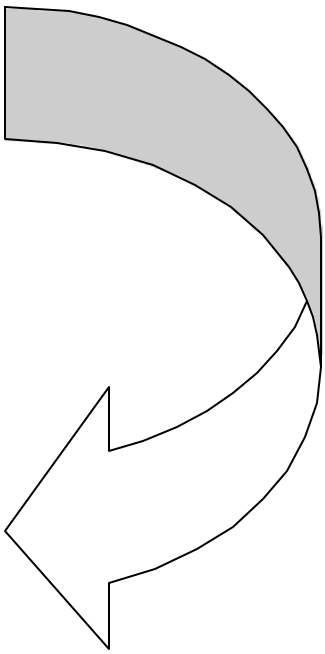
um conjunto de elementos da rede. As saídas são classificadas como falha1, se for entre a estação central e a base 1. Falha 2 se o alarme tiver origem entre a estação central e a Base 2. Desta maneira, a rede direta satisfaz a solução do problema, que é apontar a origem do alarme entre duas bases operacionais e a estação central. No caso desta rede ser expandida, ou seja, se forem colocadas mais bases operacionais para atender a demanda de usuários a rede neural deverá ser reconstruída e deverá ser treinada para a nova situação. No momento em que a rede de telefonia celular tiver um aumento significativo, a rede neural direta deverá possuir tantas entradas quantas forem necessárias para modelar o número de bases operacionais. Pode-se pensar em 100 bases operacionais cada uma, com 4 neurônios de entrada (que é o número de alarmes).

Este exemplo ilustra que através da utilização de uma rede recorrente não seriam necessários tantos neurônios de entrada. Como uma rede recorrente tem capacidade de aprender transições de estado, as bases operacionais poderiam ser aumentadas, porém, a topologia da rede não seria alterada. Neste caso, seria apenas necessário o treinamento para um novo conjunto de informações (Figura 10).

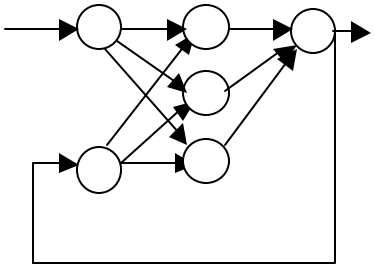
Modelo Utilizando RNAs Diretas



Cada entrada é um tipo de alarme
Problema: se o número de estações crescerem não haverá neurônios suficientes para representá-los.



Modelo Utilizando RNAs Recorrentes



Neste tipo de RNA as entradas são seqüenciais não limitam a estrutura da rede

Figura 10 - Exemplo de utilizaçao de RNA Direta x RNA Recorrente

Capítulo 4 Teoria de Sistemas

4.1 Introdução

A teoria de sistemas foi criada com a finalidade de ser independente do domínio estudado, tal como uma teoria matemática [6][7][8]. Conceitos como: o que constitui um sistema; qual a orientação do sistema (entradas e saídas); quais as funções do sistema; topologia (no caso de um sistema complexo – são necessários para a compreensão de sistemas estáticos e sistemas dinâmicos). A metodologia para o desenvolvimento de agentes inteligentes em gerência de redes utilizará estes conceitos, por isso este capítulo se faz necessário. A Figura 11 ilustra um diagrama com os tipos de sistemas.

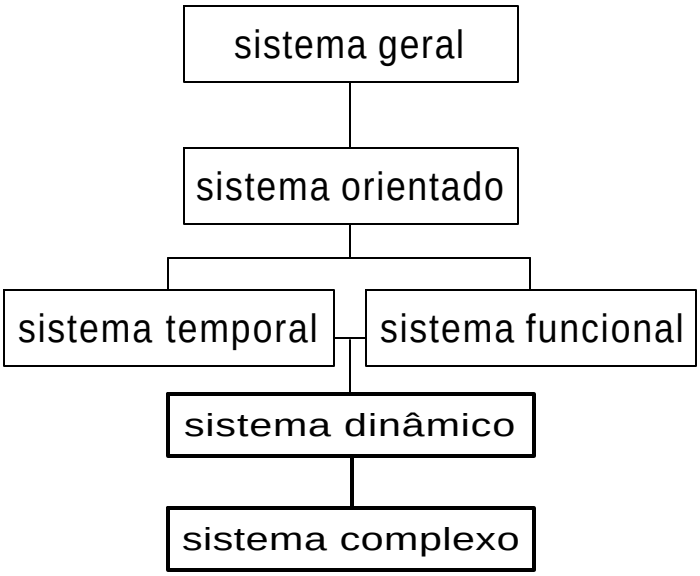


Figura 11. Teoria de sistemas

O sistema geral possui a mais alta abstração de um sistema, pela sua definição é possível afirmar que um sistema geral é constituído por um conjunto de relações entre suas entidades e o mundo exterior. O sistema orientado possui conjunto de entrada e saídas. O sistema temporal apresenta seus conjuntos em função do tempo, enquanto que o sistema funcional em função de estados. Estes dois últimos sistemas unidos formam sistemas dinâmicos. Finalmente, sistemas complexos são caracterizados por um agrupamento de sistemas dinâmic os. Estes conceitos serão discutidos em maiores detalhes porque serão necessários para a compreensão das próximas seções deste trabalho.

4.2 Sistema Geral

Em uma primeira definição um sistema geral pode ser considerado como o menor nível de descrição possível, em que somente as interações externas são conhecidas. Ou seja, o conceito de sistema geral é o mais alto nível de abstração sobre o conceito de um sistema, que pode ser visto como uma caixa preta com indicações de interação com o mundo exterior.

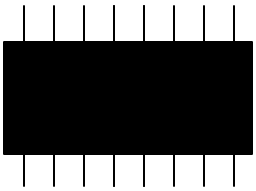


Figura 12. O sistema geral visto como uma caixa preta

Definição 1.1: um sistema geral Σg é definido por um conjunto de relações entre as entidades que caracterizam as interações com o mundo exterior. Então $\Sigma g \in$ I, onde I são todas as interações.

4.3 Sistema Orientado

Caso deseje-se definir quais das variáveis serão usadas como entrada e quais serão utilizadas como saída, tem-se um sistema orientado. Seguindo a analogia da caixa preta, o sistema orientado poderia ser visto como a caixa preta com indicações das entradas e saídas (Figura 13).

Figura 13. Sistema orientado

Definição 1.2: um sistema orientado pode ser caracterizado por uma relação entre o conjunto de entradas e saídas.

$$\Sigma o \subset \Omega \times \Gamma$$

onde:

Σo é o sistema de transferência de arquivos;

Ω é o conjunto de entradas admissíveis, ou seja é o conjunto de operações que podem ser solicitadas ao sistema. Onde cada uma das operações são representadas por ω, de forma que $\omega \in \Omega$. Ω = {append, bye, delete, dir, cd, lcd, open, quit, user}.

Γ é o conjunto de saídas admissíveis, ou seja, a resposta para cada operação fornecida na entrada do sistema. Se for colocado na entrada uma operação `ftp <endereço>`, tem-se:

```
C:\>ftp ftp.ufsc.br
Connected to ftp.ufsc.br.
220 ftp.ufsc.br FTP server (Version wu-2.4.2-academ[BETA-
15](1) Tue Jun 9 15:44:
20 GRNLNDDT 1998) ready.
User (ftp.ufsc.br:(none)): anonymous
331 Guest login ok, send your complete e-mail address as
password.
Password:
230 Guest login ok, access restrictions apply.
```

Outra operação: se fornecer como entrada a operação *dir* tem-se a seguinte saída*:*

```
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 48
d--x--s--x   2 0   0          512 May 10 1998  bin
dr--r-Sr--   3 0   0          512 May 26 1998  etc
drwxr-s--x  30 0   0          2048 Jun  2 11:07 incoming
dr-xr-xr-x   2 0   0          512 May 26 1998  lib
dr-xr-xr-x  33 0   0          1024 Jun  1 10:12 pub
dr-xr-xr-x   3 0   0          512 May 26 1998  usr
226 Transfer complete.
369 bytes received in 0.16 seconds (2.31 Kbytes/sec)
```

4.4 Sistema Temporal

Os dois conceitos apresentados, sistemas geral e orientado, não incluem o tempo como intrínseco ao sistema (Figura 14). Entretanto, freqüentemente quando lidando com sistemas que são realizadas abstrações do mundo real, o tempo é um parâmetro relevante. O sistema temporal é o sistema orientado com o acréscimo dos tempos. Pode-se imaginar uma entrada na caixa preta no tempo inicial (t_0), e após uma quantia de tempo denotado por t_{i+1} é gerada uma saída.

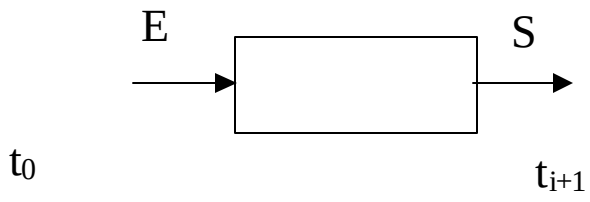


Figura 14. Sistema temporal

Exemplos de Sistema Temporal

A utilização de um enlace de uma rede Ethernet pode suportar altas rajadas de tráfego, no entanto se a carga normal exceder a 40% o segmento de rede terá seu desempenho comprometido. Para determinar a utilização de uma linha é necessário examinar a quantidade de bits transmitidos e de bits recebidos sobre a largura de banda disponível (Figura 15).

Tem-se então:

Utilização = (bits_recebidos + bits_ transmitidos) / largura_de_ banda

Definição 1.3: um sistema orientado onde Ω e Γ são funções do tipo: $\Omega: T \rightarrow U$ e $\Gamma: T \rightarrow Y$, onde:

U é o conjunto dos valores de entrada, valores transmitidos e recebidos no tempo inicial, *t0*.

Y é o conjunto dos valores de saída, passado o tempo *t+1*, ou seja, serão os valores transmitidos e recebidos no tempo *t* e o valor da utilização neste período.

T é um conjunto ordenado com um primeiro elemento, normalmente denotado por t_0 (às vezes é usual considerar t_0 como $-\infty$) que é chamado de conjunto $T = \{t_0, t_1, t_2, ...t_n\}$.

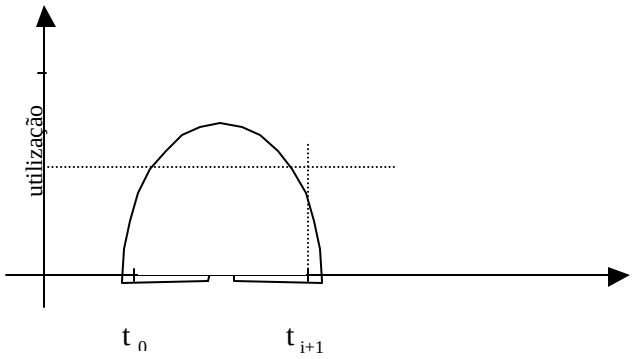


Figura 15. Taxa de utilização de uma rede

Note que um sistema temporal cujas funções Ω e Γ são constantes, ou seja, a imagem será os mesmos elementos de U e Y, é dito ser um Sistema Estático. No exemplo do servidor de uma rede, se não houver alteração na taxa de utilização em um determinado período, ou seja, se este for constante, ou é porque houve alguma falha ou o sistema permaneceu estável naquele período.

4.5 Sistema Funcional

Conceito de Estado: o conceito de estado é freqüentemente usado em várias disciplinas, por exemplo, na teoria de autômatos finitos.

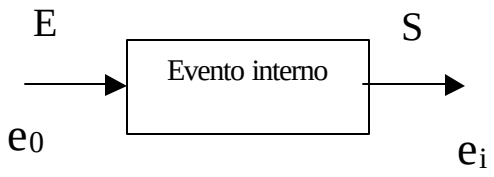


Figura 16. Sistema funcional

Através de um exemplo bem simples, considere uma agenda telefônica, em que a entrada é o nome da pessoa e a saída é o telefone. E se houver mais de um telefone: o de casa e o celular, por exemplo? Qual o telefone aparecerá? É possível definir um conjunto de estados X = {telefone_casa, telefone_celular, telefone_escritório} e aí definir a função:

Definição 1.4 : Dada a seguinte função $f: \Omega \times X \rightarrow \Gamma$, onde:

f é a função busca de número de telefone conforme o número;

Ω é o conjunto das entradas admissíveis, ou seja, nome da pessoa a ser procurada na agenda, ω, de forma que $\omega \in \Omega$;

X é o conjunto dos estados, X = {telefone_casa, telefone_celular, telefone_escritório}, que poderá mudar automaticamente, e esta mudança poderá ocorrer de diversas formas. Se h = 8 –12 e 14 – 18 então telefone_trabalho; se h = 18 – 8 e 12 – 14 então telefone_casa. Se telefone_casa não responde depois de 20 chamadas então telefone_celular.

Γ é a saída, conjunto de telefones da lista, em que γ é um telefone, $\gamma \in \Gamma$.

4.6 Sistema Dinâmico

Quando um sistema possui características de um sistema funcional e de um sistema temporal, ao mesmo tempo tem-se um Sistema Dinâmico (Figura 17), com o estado do sistema variando conforme o tempo. Em um sistema dinâmico, descreve-se um sistema como se estivesse

descrevendo o mecanismo de como ele trabalha (internamente), especificando como o conjunto de estados varia conforme o tempo. Na analogia com a caixa preta seria, como é o comportamento interno desta caixa conforme o tempo. Para gerar uma descrição comportamental basta fazer o mecanismo funcionar para cada entrada desejada, gerando o sistema orientado correspondente.

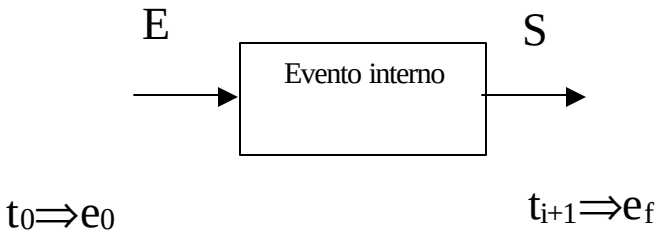


Figura 17. Sistema dinâmico

Formalmente, tem-se:

Definição 1.5: um sistema dinâmico é o objeto matemático.

$S=\{T,U,\Omega,Y,\Gamma,X,\Phi;\eta\}$

Onde:

T é o conjunto dos tempos,

Ω é o conjunto das funções de entrada $\Omega=\{\omega:T\rightarrow U\}$,

U é o conjunto dos valores de entrada,

Y é o conjunto dos valores de saída,

Γ é o conjunto das funções de saída $\gamma \in \Gamma = \{\gamma:T \rightarrow Y\}$,

X é o conjunto dos estados,

Φ é a função de transição dos estados: $\Phi : T \times T \times X \times \Omega \rightarrow X$,

η é a função de saída: $\eta: T \times X \times U \rightarrow Y$.

Observação: quando o conjunto de estados é vazio, o sistema é considerado estático. Ou seja, ele não varia o estado conforme o tempo. O estado é sempre o mesmo.

Exemplos de Sistema Dinâmico

Um modem "half-duplex" pode encontrar-se em um dos três estados: Inativo, Transmitindo ou Recebendo. A troca de estado dependerá se existe ou não mensagens na fila de transmissão ou se existe alguma indicação para recepção de mensagens, caso contrário permanece inativo.

Formalizando o exemplo tem-se:

$S=\{T,U,\Omega,Y,\Gamma,X,\Phi;\eta\}$

Onde:

T é o conjunto dos tempos T, em que $T \in \Re$;

Ω é o conjunto das funções de entrada $\Omega = \{\omega:T\rightarrow U\}$, se existem mensagens na fila de transmissão ou se existe sinal para recebimento de mensagens;

U é o conjunto dos valores de entrada, as mensagens a serem transmitidas e recebidas;

Y é o conjunto dos valores de saída, Y = {sinal de reconhecimento, sinal de erro).

Γ é o conjunto das funções de saída $\gamma \in \Gamma = \{\gamma:T\rightarrow Y\}$, emite o sinal de reconhecimento se a mensagem foi recebida corretamente, caso contrário envia um sinal de erro;

X é o conjunto dos estados do sistema, X={Inativo, Transmitindo, Recebendo};

Φ é a função de transição dos estados: $\Phi : T \times T \times X \times \Omega \rightarrow X$;

η é a função de saída: $\eta: T \times X \times U \rightarrow Y$.

Modelo Geral de Neurônio como Exemplo de Sistema Dinâmico

Utilizando teoria de sistemas, o modelo formal de um neurônio pode ser apresentado como um sistema dinâmico, conforme segue. Com efeito, as variáveis de definição de um sistema dinâmico são identificáveis em um neurônio formal. Observe o seguinte objeto matemático:

$S=\{T,U,\Omega,Y,\Gamma,X,\Phi;\eta\}$

Onde:

T é o conjunto dos tempos, conjunto munido de uma ordem completa. Exemplos de conjuntos de tempo usuais são os números naturais e os reais.

Ω é o conjunto das funções de entrada $\Omega=\{\omega:T\rightarrow U\}$, provenientes dos sensores dos neurônios ou da saída de outros neurônios,

U é o conjunto dos valores de entrada, que geralmente são vetores de entrada cujos componentes são os valores dos sinais nas entradas dos dendritos,

Y é o conjunto dos valores de saída, corresponde ao valor no axônio no neurônio biológico, que é a freqüência de pulsos resultantes dos potenciais de ação,

Γ é o conjunto das funções de saída $\gamma \in \Gamma = \{\gamma : T \rightarrow Y\}$

X é o conjunto dos estados, conhecidos como estados de ativação dos neurônios,

Φ é a função de transição dos estados: $\Phi : T \times T \times X \times \Omega \rightarrow X$,

η é a função de saída: $\eta : T \times X \times U \rightarrow Y$.

Quando o conjunto dos estados for igual a um, o modelo de neurônio é dito *estático*, que é um caso particular do neurônio dinâmico.

4.7 Sistema Complexo

Um conjunto de sistemas dinâmicos formam um sistema complexo. Utilizando o exemplo das caixas, seriam várias caixas de sistema dinâmico trabalhando como módulos paralelos para atingir um objetivo comum (denotado na pela seta Saída na Figura 18).

Definição 1.6: Um sistema complexo $\sum_c$ é definido como um conjunto de sistemas denotados por $\sum_d$ interconectados. $\sum_d \times \sum_d \rightarrow \sum_c$

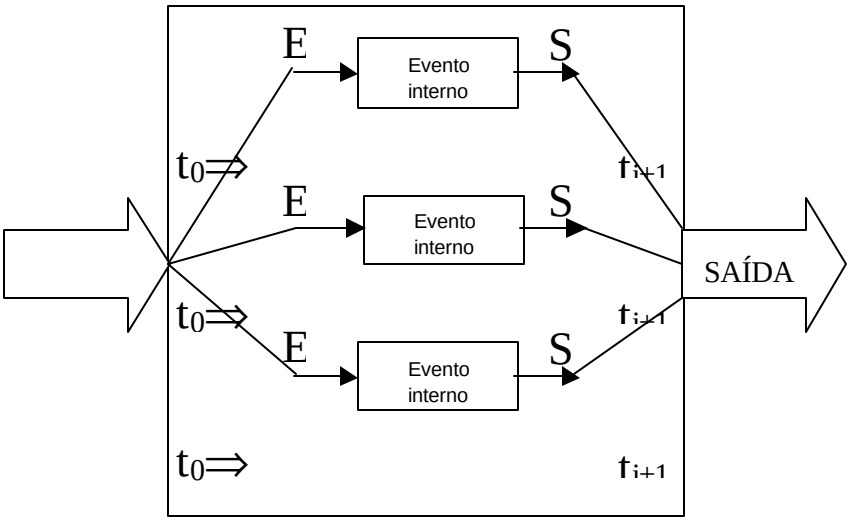


Figura 18. Sistema complexo

Exemplos de Sistema Complexo

As plataformas de gerência de redes, tanto de computadores quanto de telecomunicações, são exemplos de sistemas complexos. É possível identificar as áreas funcionais de gerência como sistemas dinâmicos que compõem tais plataformas. Por exemplo, um sistema para gerenciamento de uma rede deverá ter controle de acesso dos usuários (Gerência de Segurança), contabilizar o uso dos equipamentos (Gerência de Contabilização), incluir equipamentos ou refazer a topologia da rede (Gerência de Configuração), controlar o desempenho (Gerência de Performance), prevenir e solucionar falhas na rede (Gerência de Falhas). A maioria destas, entre outras funções não citadas, devem ser feitas em paralelo sobre um sistema também dinâmico (nesse caso, a rede de computadores).

Os sistemas multi-agentes também podem ser considerados sistemas complexos. Em um sistema multi-agentes cada entidade assume um papel dentro da solução do problema. Existem outras abordagens sobre sistemas multi-agentes que podem ser encontrados em [10][11][55][69][90][92][101][104]. Outro exemplo é o organismo humano que possui características cognitivas, como visão, audição, tato, paladar e olfato.

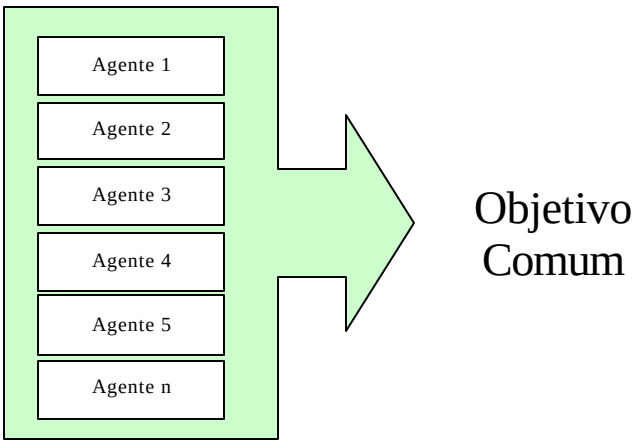


Figura 19. Sistemas multi-agentes como sistemas complexos

Capítulo 5

Desenvolvimento de Agentes Inteligentes e Distribuídos

5.1 Introdução

A inspiração biológica na área da IA sempre se faz presente, e não existe maneira melhor de explicar o que é IA Distribuída (IAD). Imagine uma colônia de formigas que tem sobrevivido aos rigores da evolução por cerca de bilhões de anos. Com a visão de apenas uma formiga não é possível compreender a imensidão de uma colônia. Por outro lado, do ponto de vista da colônia podemos ver a distribuição do trabalho como uma linguagem muito mais poderosa. Estas colônias só sobrevivem porque a distribuição tem um significado, que provavelmente é invisível aos níveis inferiores [51][52].

Inspiração biológica, além do exemplo das formigas citado anteriormente, pode-se pensar em outros seres vivos que para sobreviver vivem em bandos, para proteger os seus filhotes, por exemplo.

- O próprio ser humano, agrupa-se em nacionalidades, sociedades, comunidades para solucionar seus problemas maiores, formam uma estrutura tão complexa que existe uma área destinada ao seu estudo que é a *sociologia*;
- Decompor problemas complexos em problemas de menor escala, cada um com suas características distintas (porém esta não é a única abordagem para implementação de IAD);

- Utilizar as vantagens de ambientes distribuídos para determinar vários agentes paralelos que possam trabalhar de forma cooperativa.

Em [86], Rich & Knight, apresentam duas contribuições importantes relacionadas aos conceitos de arquiteturas paralelas e distribuídas que influenciam no desenvolvimento de sistemas inteligentes. Primeiro, a modelagem psicológica, visto que o ser humano trabalha em paralelo. Segundo, a modularidade, é mais fácil operar sobre módulos independentes do que em um módulo único, desta forma existe um aumento da eficiência porque nem todos os conhecimentos são necessários a todos os módulos.

Uma das primeiras implementações em IAD foi o modelo do quadro negro, apresentado no projeto de compreensão da fala HEARSAY-II [86]. No entanto existem vários outros métodos.

Bem mais recentes, são os agentes inteligentes, ou agentes autônomos de software, que serão analisados na próxima seção. Em Barreto [8], é feito um breve comentário sobre a aplicação de agentes autônomos de software em gerência de redes. Transcrevendo suas palavras: *"Uma das mais promissoras aplicações é na gerência de redes de computadores. Neste caso cada nó estaria munido de um agente funcionando de modo autônomo, e contendo inteligência suficiente para desempenhar sua função de gerir o funcionamento do nó do melhor modo possível"*.

5.2 Agentes na Solução de Problemas

Agentes têm sido utilizados para solucionar os mais variados tipos de problemas. Entenda-se "paradigma" como um termo formal utilizado para modelos que explicam ou mostram como determinados sistemas podem ser produzidos. É possível considerar então, agentes como um paradigma para o desenvolvimento de sistemas, sejam simples ou complexos. No entanto, existem diversos tipos de agentes, de forma que não é possível considerar com o mesmo nível de classificação a função de um robô (que é um agente autônomo de hardware) e de um programa coletor de dados de gerência, por exemplo.

Na Figura 20 o presente trabalho propõe uma taxonomia para o uso de agentes no sentido de ilustrar diferentes comportamentos dos agentes considerando a aplicação de técnicas de IA.

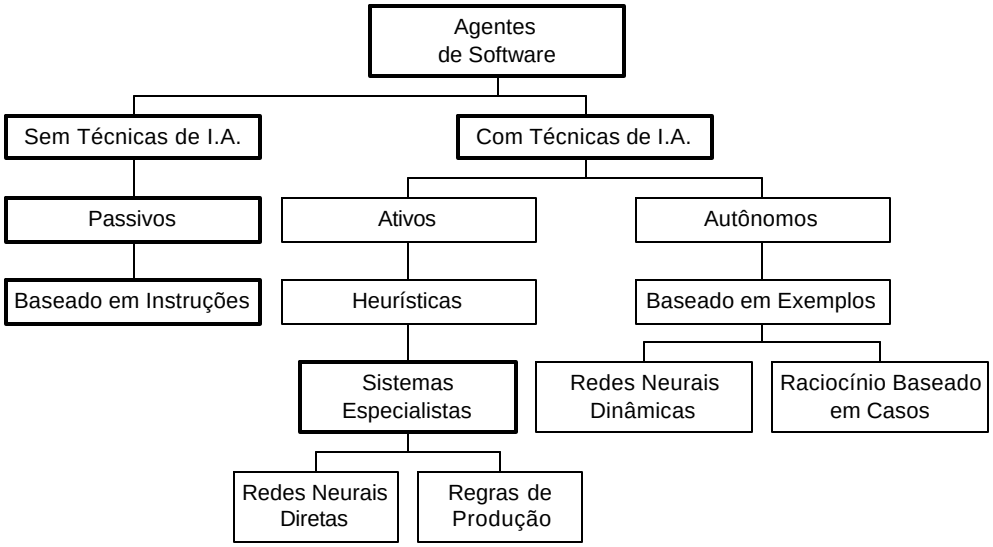


Figura 20. Taxonomia de agentes

Agentes de hardware normalmente são autônomos e desenvolvidos com o auxílio de técnicas de inteligência artificial (IA). Segundo Roisenberg [87] "agentes autônomos são definidos como sistemas computacionais que operam em ambientes dinâmicos e imprevisíveis". Eles interpretam dados obtidos pelos sensores que refletem eventos ocorridos no ambiente e executam comandos em motores que produzem efeitos no ambiente. O grau de autonomia de um agente está relacionado à capacidade de decidir por si só como relacionar os dados dos sensores com os comandos dos seus atuadores em seus esforços para atingir objetivos, satisfazer motivações, etc. As principais aplicações são encontradas na área da robótica, nas quais destacam-se: limpeza de material nuclear, escavações de ambientes perigosos e exploração de outros planetas [49]. No entanto, como estes agentes não estão viculados diretamente com a área de redes de computadores não foram incluídos na classificação representada pela figura acima.

Já os agentes de software são encontrados em um número muito maior de aplicações, eles podem ou não ser desenvolvidos com o auxílio de técnicas de IA. Agentes passivos foram classificados como agentes que não possuem autonomia e não utilizam técnicas de IA. Assemelham-se a simples programas e são construídos através de instruções. Os agentes de software desenvolvidos com o auxílio de técnicas de IA podem ser ativos ou autônomos. Os agentes ativos são desenvolvidos com o auxílio de heurísticas. Na área de gerência de redes, estas heurísticas são fornecidas pelo administrador da rede. Servem para definir regras de produção ou redes neurais diretas para auxiliar na solução dos problemas. São semelhantes aos sistemas especialistas e não possuem autonomia. Agentes autônomos possuem características dinâmicas. Devem ser desenvolvidos com o auxílio de técnicas que satisfaçam esta característica, e podem ser desenvolvidos com o auxílio de RNAs dinâmicas ou através de técnicas de raciocínio baseado em casos.

5.3 Agentes sem Técnicas de IA

Existem agentes que não necessitam da utilização de técnicas de IA. Os agentes de gerência de redes são baseados em instruções. Normalmente, segundo o protocolo SNMP, as informações entre gerentes e agentes são trocadas através de mensagens tipo GET-REQUEST, GET-RESPONSE, GET-NEXTREQUEST, SET-REQUEST e TRAP. As mensagens de GET-RESPONSE são utilizadas para responder as consultas dos gerentes. Enquanto, que as mensagens de SET-REQUEST são utilizadas para alterar os valores das entidades gerenciadas.

As mensagens *traps* são assíncronas enviadas pelos agentes para os gerentes quando algum evento ocorre. As mensagens *traps* não são garantidas, o agente faz uma tentativa de melhor esforço para notificar a estação de gerência [3][85]. A Tabela 4 descreve alguns exemplos de mensagens de *traps* definidas no protocolo SNMP. O agente poderá gerar estas mensagens se o evento pré-definido for detectado através da monitoração do objeto que está sendo gerenciado.

Tabela 4. Exemplos de mensagens *traps* do SNMP

TIPO DE TRAP	DESCRIÇÃO

ColdStart Trap	Inicialização com modificação na configuração ou na implementação do protocolo.
WarmStart Trap	Inicialização sem modificação na configuração ou na implementação do protocolo.
LinkDown Trap	Canal de comunicação inativo.
LinkUp Trap	Canal de comunicação ativo.
AuthenticationFailure Trap	Erro na autenticação de uma mensagem de solicitação.
EGPNeighborLoss Trap	Perda do roteador EGP vizinho.

Além das mensagens de *traps*, os eventos são monitorados através de um mecanismo chamado *polling* que verifica periodicamente os objetos que estão sendo gerenciados [3]. Este mecanismo permite a identificação e o diagnóstico de falhas nos objetos. Ressalta-se que estes mecanismos de verificação são configurados conforme a heurística do administrador da rede. Desde o intervalo de *polling* até os valores para determinar os possíveis diagnósticos de falhas.

5.4 Metodologia para aplicação de IA

Com base na teoria geral de sistemas, apresentada no capítulo anterior, propõe-se uma metodologia para a aplicação de técnicas de IA no desenvolvimento de agentes na área de gerência de redes. A metodologia parte da definição do problema. O que é um problema, é uma interrogação que preocupa a mente humana desde os antigos filósofos gregos. Destacam-se, Descartes com o seu método analítico baseado na filosofia *dividir para conquistar,* e deste século, Polya [82] sugere que se responda as seguintes perguntas antes de solucionar um problema:

- Quais são os dados?
- Quais são as possíveis soluções?
- O que caracteriza uma solução satisfatória?

O problema pode ser formalizado como um objeto matemático do tipo: P = {D, R, q}, consistindo de dois conjuntos não vazios, D os dados e R os resultados possíveis e de uma relação binária $q \subset D \times R$, a condição que caracteriza uma solução satisfatória, associando a cada elemento a solução única desejada. Desta forma o problema é representado como uma função.

Segundo Barreto [8] existem diversas formas de definir uma função, dentre elas destacam-se enumeração exaustiva, declarativamente, por um programa ou através de exemplos. Caso um problema não seja completamente definido para todos os valores de seus dados, conhecendo-se apenas a definição do problema para um subconjunto dos dados possíveis, a solução não será única: todas as funções que forem iguais dentro da região em que o problema é definido são válidas. Neste caso, é melhor ter uma solução aproximada do que os dados para definir a função.

Este trabalho propõe que resolver o problema será então encontrar um modo de implementar a função ou aproximá-la com as ferramentas disponíveis. Ou seja, através de exemplos treinar a rede e obter-se valores estimados da solução para os outros valores, utilizando a propriedade de generalização das RNAs.

Na Figura 20 é apresentada uma classificação para o desenvolvimento de agentes de acordo com o problema a ser resolvido.

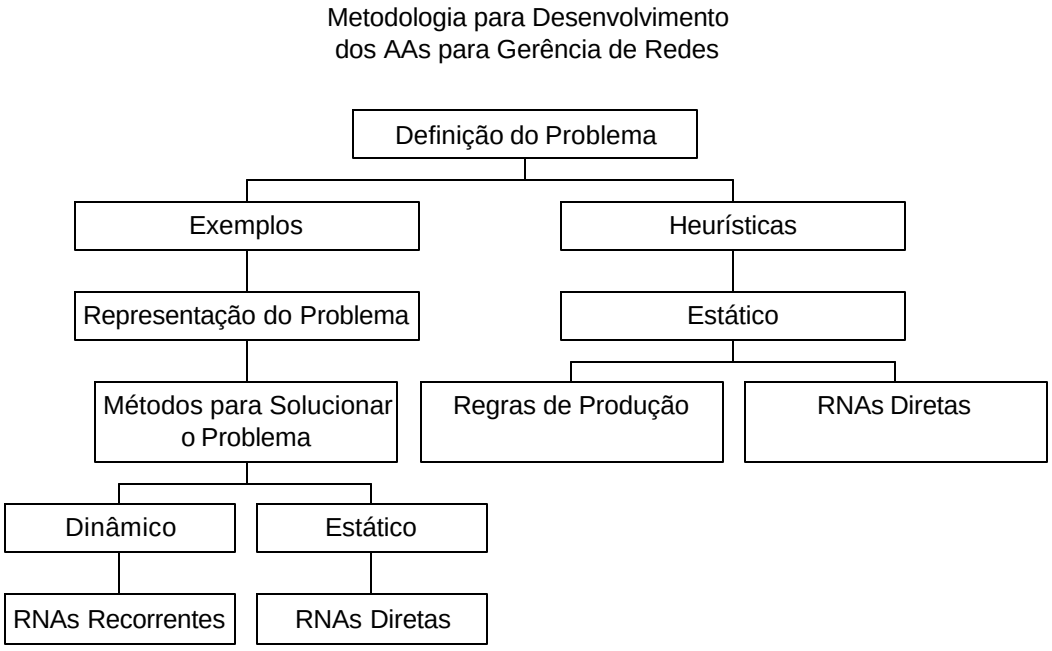


Figura 21. Metodologia híbrida para solução de problemas distribuídos

5.4.1 Problemas bem definidos

Se o problema é bem definido então a solução é conhecida: tem-se um conjunto de entradas e um conjunto de saídas bem definido. Ou seja, existem heurísticas que podem ser utilizadas para determinar a solução do problema. Ou seja, uma forma declarativa de resolver uma função, fornecendo-se as propriedades que devem ser satisfeitas para solucionar o problema,demanda o auxílio de um especialista. O especialista que será um administrador de redes deverá indicar quais os sintomas, para diagnosticar o problema e sugerir as possíveis soluções. Com essas informações pode-se construir uma base de dados (tipo "troubletickets") para um sistema de diagnóstico de falhas. Este sistema poderá utilizar técnicas de IA, tais como, regras de produção (IA Simbólica) ou RNAs diretas (IA Conexionista).

5.4.2 Problemas mal definidos

Ao contrário, o problema pode não ser bem definido. Neste caso, são necessários exemplos que poderão ser utilizados para alimentar uma RNA recorrente para aproximar uma solução. É dito que é uma solução aproximada, por entender-se que para um problema mal definido não existe uma saída e sim uma saída mais próxima à solução. Isto é definido pelo valor do erro de aproximação da rede neural recorrente. Os exemplos formam uma representação do problema, ou seja, a definição do problema é conhecida para um subconjunto de dados possíveis. A partir daí, deve-se analisar este subconjunto e verificar se é um problema estático ou dinâmico. Cada tipo de problema possui um método de solução diferente. Um problema estático pode ser resolvido por RNAs diretas. Porém o dinâmico, só é "bem resolvido" se forem utilizadas RNAs com características dinâmicas. Deseja-se então, conhecer os elementos do conjunto de respostas admissíveis para todos os elementos do conjunto de dados, mesmo aqueles que não estão incluídos na definição da função. Os exemplos, portanto, são utilizados para treinar uma rede neural com algoritmo adaptativo e obter os valores estimados da solução para outros valores, utilizando a propriedade de generalização.

As duas abordagens possuem vantagens e desvantagens. A ação dispensável do especialista é uma das principais vantagens da abordagem dos exemplos. A extração do conhecimento de um especialista é uma tarefa muito complicada. Muitas vezes, a solução de um problema é realizada com a intuição, e fica impossível indicar porque determinada ação foi

realizada. Como desvantagem tem-se a não garantia da convergência de uma rede neural. Não existem métodos para garantir que uma rede neural chegue a uma solução. No entanto, podem ser fornecidos bons exemplos para que a probabilidade de que a rede neural aprenda um determinado padrão seja boa.

5.5 Agentes com Técnicas de IA

Dois tipos distintos de agentes poderão ser desenvolvidos: Agentes Inteligentes Estáticos e Agentes Inteligentes Dinâmicos.

5.5.1 Agentes Inteligentes Estáticos

Para desenvolver-se Agentes Inteligentes Estáticos (AIE) busca-se normalmente heurísticas para a solução do problema. Normalmente, são indicados quais os sintomas para diagnosticar determinados problemas, e sugerir as possíveis soluções. A Figura 22 ilustra as técnicas de IA que podem ser aplicadas na construção de AIE.

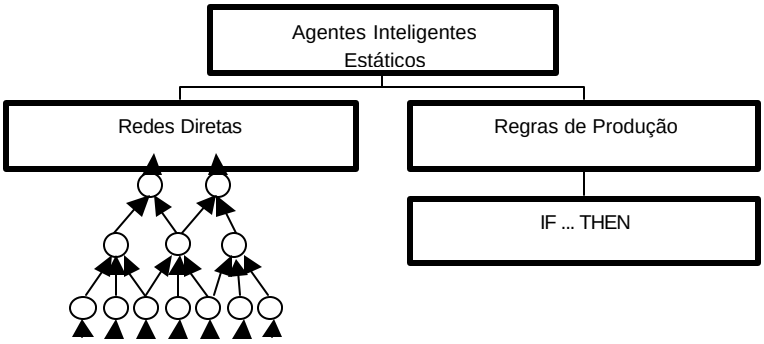


Figura 22. Agentes inteligentes estáticos

Assim, tem-se o conjunto dos dados, o conjunto das possíveis soluções e a relação entre eles de forma bem definida. Estas informações serão utilizadas para treinar uma rede neural direta

(paradigma conexionista) ou poderão ser traduzidas para regras de produção (paradigma simbólico).

Koch [60] apresenta agentes autônomos para gerência de redes utilizando regras de produção e uma pequena biblioteca para programação de uma rede neural direta.

Outros exemplos

Um exemplo de sistema que pode ser implementado com agentes inteligentes estáticos é o de diagnóstico de falhas, ou até mesmo uma base de dados com fichas de falhas da rede e suas possíveis soluções ("troubletickets"). A seguir segue um exemplo de regras de produção que identificam falhas em pacotes recebidos pela rede. Estas regras foram utilizadas em um trabalho anterior que implementava um sistema especialista [33][34].

```
SE Δ pacotes de entrada > baseline        (a)
   E pacotes de saída < baseline          (b)
   E erros de saída > baseline            (c)
ENTÃO  "pacotes errados"                  (X)

SE Δ pacotes de entrada > baseline        (a)
   E pacotes de saída < baseline          (b)
   E erros de saída > baseline            (c)
   E fila de pacotes de saída > baseline  (d)
ENTÃO "rajada de pacotes errados"         (Y)
```

Existem vários métodos para converter regras de produção em redes neurais diretas. Em [13][28], os autores apresentam uma taxonomia de arquiteturas híbridas e como converter regras em RNAs diretas. É necessário que as regras iniciais estejam no formato de Cláusulas de Horn, porque se deseja apenas uma saída, ou seja, uma conclusão.

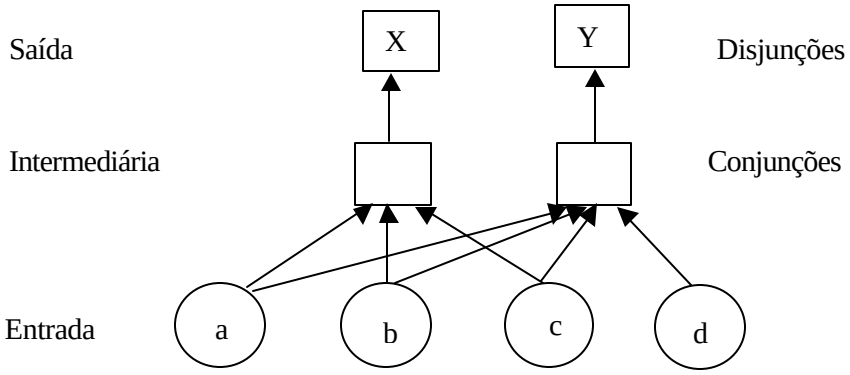


Figura 23. RNA baseada em regras

A Figura 23 apresenta a rede conexionista baseada nas regras acima apresentadas. A arquitetura da rede é caracterizada por alternar camadas conjuntivas (E) e disjuntivas (OU), porque a RNA direta equivalente às regras é construída utilizando grafos do tipo E/OU.

5.5.2 Agentes Inteligentes Dinâmicos

Agentes Inteligentes Dinâmicos (AID) podem ser desenvolvidos com o auxílio de exemplos. Através de exemplos tem-se uma representação indireta do problema, ou seja, apenas a definição do problema é conhecida para um subconjunto de dados possíveis.

Deseja-se então, conhecer os elementos do conjunto de respostas admissíveis para todos os elementos do conjunto de dados, mesmo aqueles que não estão incluídos na definição da função. Para entender-se o funcionamento destes agentes pode-se fazer uma analogia com o traçado de uma reta a partir de um conjunto de pontos de entrada. Basta pensar que é possível que alguns pontos tenham sido fornecidos com erros e a reta será traçada a partir da correção do erro e finalmente a plotagem da reta. A Figura 24 ilustra as tecns de IA aplicadas no desenvolvimento de AID.

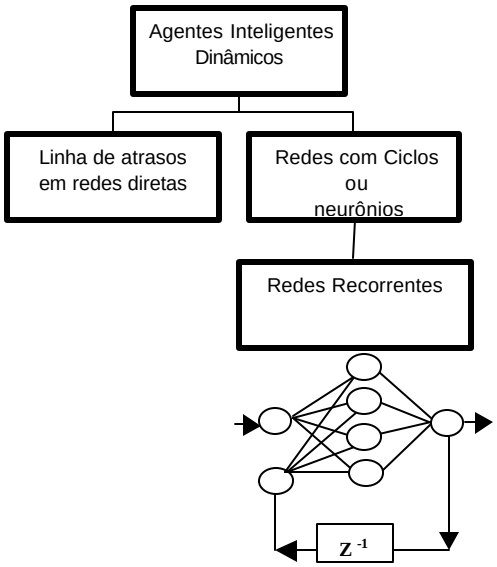


Figura 24. Agentes inteligentes dinâmicos

Da mesma forma irá funcionar o comportamento destes agentes, fornece-se um conjunto de exemplos de entrada à uma rede neural recorrente e ela aproxima uma determinada função a partir da entrada fornecida.

É o mesmo caso de técnicas de raciocínio baseado em casos, só que neste caso existe o problema de encontrar os pesos das diferenças entre os casos. Por exemplo, tempo de resposta longo em uma rede de computadores pode significar vários tipos de falhas. Fazendo uma analogia a sistemas médicos, vários tipos de doenças têm como sintoma febre alta. A maior limitação desta técnica é como diferenciar o peso de cada sintoma para cada doença.

Outro exemplo

Supondo-se um agente interpretador de eventos que poderia ser ativado por sinais de *polling* e alarme de eventos.

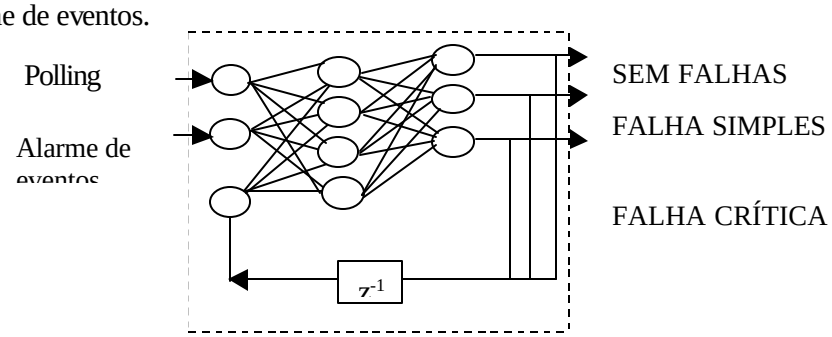


Figura 25. Rede recorrente que implementa um agente interpretador de eventos

Em primeiro lugar, define-se o problema. Tem-se um conjunto de entrada e saída bem definidos, e um problema dinâmico. As características dinâmicas são determinadas pelo número de equipamentos que podem enviar os sinais para o agente. Se fosse utilizada uma RNA direta com uma entrada para cada máquina, o agente estaria limitando as suas entradas ao número de máquinas na rede. Portanto, propõe-se uma rede recorrente com entradas seqüênciais.

Como entrada tem-se uma seqüência de *polling* e uma seqüência de alarmes de eventos. Atividades de *polling* são utilizadas em plataformas de gerência para verificar a ocorrência de falhas, enquanto que os alarmes de eventos são limites definidos previamente no

caso da ocorrência de falhas em uma rede. Supondo-se os valores da Tabela 5 como ativação dos sensores para classificar se existe ou não falha na rede. Se pelo menos um sinal 1 aparecer na seqüência é porque ocorreu um evento. Se nas duas entradas aparecer sinal 1 é porque existe a ocorrência de um evento grave.

Tabela 5. Sinais de ativação do agente – entradas e saídas

Entrada		Saída		
P	E	SF	FS	FC
0	0	1	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

Após a definição das entradas e saídas utiliza-se um algoritmo para treinamento da rede. Nos experimentos preliminares deste trabalho foi aplicado o algoritmo proposto por Roisenberg [87] para treinamento de redes recorrentes. Considerando o retardo da rede como uma terceira entrada e em seguida treina-se com o algoritmo de retropropagação.

É necessário esclarecer que deve existir uma hierarquia de agentes, onde cada um possui funções específicas. As informações de entrada deste agente podem ser recebidas de outros agentes considerados coletores de informações. Semelhantes aos agentes de gerência de redes, que não possuem autonomia apenas coletam informações da base de dados.

5.6 Trabalhos relacionados

Desde o início desta pesquisa foram lidos diversos tipos de artigos envolvendo as áreas de inteligência artificial, redes de computadores, sistemas distribuídos e gerência de redes. No entanto, alguns trabalhos merecem destaque por seguirem uma linha de pesquisa semelhante, ou por aplicar outros tipos de técnicas de IA na área de gerência de redes.

Em [80], o autor desenvolveu uma máquina de aprendizado para encontrar falhas na rede, analisando uma seqüência de caracteres sobre o tempo. Entre as vantagens sobre o sistema tradicional de encontrar falhas na rede, o autor aponta a sua técnica como pró-ativa. Outra

vantagem, as regras são aprendidas a partir do histórico do estado da rede, as regras podem então ser utilizadas para encontrar como a dinâmica da rede evolui sobre o tempo, ou ainda diferentes conjuntos podem ser aprendidos para diferentes condições de operação da rede. As regras podem ainda ser utilizadas por sistemas especialistas.

Em [53], o autor apresenta um agente baseado em redes *bayesianas* para identificação de falhas para gerência pró-ativa. Neste trabalho, são definidos objetos que formam uma rede probabilística, do acontecimento de falhas. Não são descritos quais os objetos foram utilizados e o comportamento dos objetos é considerado normal ou anormal.

Em [1], autor utiliza lógica temporal para encontrar uma relação entre eventos com propósito de correlação. Uma base de regras de inferência é usada para integrar funções que determinam qual a melhor relação entre os eventos. A correlação de eventos é uma tentativa de minimizar o número de disparos de alarmes emitidos por uma mesma falha. Em muitos casos, diversos sintomas que indicam a mesma falha disparam alarmes, muitas vezes confundindo o administrador de redes que tem a obrigação de correlacionar os eventos heuristicamente.

Na tentativa de automatizar o processo de gerência de falhas, em [38], o autor apresenta como desenvolver agentes móveis utilizando um conjunto de ferramentas para descrição de agentes móveis. Este trabalho utiliza regras de aprendizado fornecidas ao JESS, que é utilizado como máquina de inferência, porém não apresenta as regras no decorrer do trabalho.

5.7 Plataformas para Desenvolvimento de Agentes

A maioria dos trabalhos envolvendo agentes inteligentes baseia-se em plataformas de implementação de sistemas multi-agentes. Na área de gerência destacam-se duas: o JDMK da Sun e os *aglets* da IBM. As demais são utilizadas para desenvolver multi-agentes em ambientes distribuídos. Normalmente, quando utilizam regras de produção, os agentes inteligentes utilizam o JESS como mecanismo de inferência das regras. Estas plataformas estão dispostas neste trabalho para efeito de conhecimento do que é utilizado na área de desenvolvimento de agentes inteligentes.

Bigus em [12] descreve como desenvolver agentes inteligentes utilizando a linguagem JAVA e a estrutura ABLE. Entre as aplicações utilizando o paradigma conexionista destacam-se

as redes neurais diretas com retropropagação e mapas de Kohonen. Existem outras plataformas para o desenvolvimento de agentes que permitem a construção de sistemas de inferência baseados em regras de produção que serão descritas ao longo desta seção.

Existe um órgão chamado FIPA (*Foundation for Intelligent Physical Agents*) que regulamenta especificações para plataformas de desenvolvimento de sistemas de agentes. A relação das plataformas que seguem estas regulamentações pode ser obtida pela Internet (http://www.fipa.org/resources/livesystems.html).

Além das plataformas apresentadas, Roisenberg [88] propõe um modelo formal baseado na teoria de sistemas para o desenvolvimento de agentes autônomos. Este modelo poderia ser utilizado para criar agentes em um ambiente distribuído.

ABLE (Agent Building Learning Environment)

O ABLE é uma estrutura para desenvolvimento e organização de agentes híbridos e aplicações de agentes. Esta estrutura fornece um conjunto de componentes Java Beans reusáveis, chamados Able Beans, que através de vários métodos de interconexão combinados com estes componentes permitem a criação de agentes de softwares. Os Able Beans implementam acesso de dados, filtragem e transformação, capacidades de aprendizado e raciocínio.

JDMK (Java Dynamic Management Kit)

O JDMK é utilizado na construção e distribuição de gerência de redes dinâmica e inteligentes através de aplicações, redes e dispositivos. Utiliza os componentes *Java Beans* que permitem criar rapidamente, soluções embutidas para gerenciar todo o sistema de redes. Os componentes de *Java Beans* são utilizados para modelar os componentes das redes que são gerenciadas. Podem ser dispositivos, tempo de computação, controles da rede, aplicações do usuário, enfim qualquer objeto que possa ser manipulado através de uma aplicação de gerência. No entanto, os agentes são como um elo de ligação entre gerentes e objetos gerenciados, da mesma forma que é realizado em plataformas de gerência tradicionais. A estrutura envia solicitações de gerência e retorna eventos de volta aos gerentes. Os *Mbeans* (como são chamadas as estruturas *Java*

Beans gerenciáveis no JDMK) podem ser escalonados ou não da estrutura, como se fossem hardwares empilhados. Esta API de gerência habilita aplicações para acessar agentes de forma transparente, utilizando os protocolos SNMP, HTTP, RMI, etc). No entanto, na modelagem para o comportamento inteligente dos agentes utilizam regras de produção.

Aglets

Os *aglets* são agentes autônomos desenvolvidos pela IBM. Eles fornecem capacidades básicas necessárias para a mobilidade. Cada *aglet* tem um nome global único. Um intinerário e as ações que serão realizadas durante a viagem devem ser especificados. No caso de um agente ter de ser executado em um ambiente particular, o sistema deve ser capaz de executar uma aplicação de. O *aglet* comunica-se utilizando um quadro branco para habilitar agentes para colaborar e compartilhar informações assincronamente. Suporta informações síncronas e assíncronas para realizar a *aglet* comunicação entre os agentes. No entanto, não são considerados inteligentes. Podem utilizar o código JAVA para extensões inteligentes através do JESS.

JADE (Java Agent Development Environment)

Este ambiente de desenvolvimento foi criado pelo CSELT S.p.a, em Torino, Itália. É um conjunto de ferramentas para a criação de aplicações multi-agentes baseadas nas descrições da FIPA. Fornece um conjunto de ferramentas para depuração e desenvolvimento de agentes distribuídos. Este ambiente fornece um conjunto de serviços, incluindo serviços de nome, protocolos de transporte, e protocolos de interação conforme a FIPA. Este ambiente precisa de JVM (*Java Virtual Machine*) em cada máquina para o correto funcionamento dos agentes. Suporta a integração com a máquina de inferência JESS.

JESS (Java Expert System Shell)

É uma implementação do sistema especialista CLIPS (*C Language Integration Production System*) escrito em JAVA. Ernest Friedman-Hill desenvolveu JESS nos Laboratórios Sandia. Permite a inclusão de regras de inferência aos agentes desenvolvidos em JAVA. A sintaxe é muito semelhante ao código LISP. Fornece um ambiente para entrada interativa e avaliação de

regras, criando uma base de conhecimento com fatos para representar uma instância específica de um componente *Java Bean*.

Linguagem KQML (Knowledge Query and Manipulation Language)

Apesar de não ser uma plataforma, KQML é a linguagem mais utilizada para realizar a comunicação entre sistemas de agentes. A linguagem foi projetada para suportar interações entre agentes inteligentes. Ela foi desenvolvida pelo ARPA (DARPA - *Defense Advanced Research Projects Agency*), no programa de compartilhamento de informações e implementada de forma independente por diversos grupos de pesquisa. A comunicação existe em vários níveis. O conteúdo da mensagem é apenas uma parte da comunicação [44][60].

Capítulo 6

Experimentos Realizados

6.1 Introdução

Inicialmente, selecionou-se um problema dinâmico com saída indefinida para testar o aprendizado de uma RNA através de exemplos. Uma rede recorrente de Elmann foi utilizada, porém, o resultado não foi satisfatório. Optou-se então, por construir uma rede neural personalizada para solucionar o problema. A resolução consiste em encontrar uma função que represente o conjunto de entrada fornecido à RNA.

Neste capítulo são apresentados experimentos realizados considerando a metodologia para desenvolvimento de agentes para gerência de redes. Foram testadas três topologias de RNAs recorrentes diferentes, cada RNA proposta possui número de entradas distintas. Os dados utilizados para o treinamento e simulações foram obtidos no ambiente de teste descrito na seção 6.4. Foi utilizado o conjunto de funções de redes neurais, o NNET Toolbox, disponível no MatLab 6 para a construção dos modelos.

6.2 Testes com a Rede Recorrente de Elmann

Alterações foram feitas em uma rede neural recorrente de Elmann para que esta atingisse o objetivo de aprendizado por exemplos. A rede de Elmann é uma rede de duas camadas, com retroação da saída da camada intermediária para a sua entrada, conforme pode ser visualizado pela Figura 26, retirada de [37]. No *toolbox* do Matlab, os neurônios da rede de Elmann

possuem como função de ativação, na camada intermediária: a tangente hiperbólica; e na camada de saída: uma função linear. Na tentativa de aprendizado por exemplos, alterou-se a função de ativação tanto da camada intermediária, quanto da camada de saída para função sigmoidal. Esta alteração deve-se aos seguintes fatos: os primeiros testes não tinham sido satisfatórios e o conjunto de exemplos a ser aprendido é representado por valores entre 0 e 1 com distribuição não-linear. A linha abaixo identifica a criação de uma rede de Elmann com duas entradas, uma variando entre 0 e 1 (seria a taxa de utilização) e a outra o intervalo de tempo das amostras de 10 em 10 segundos (length(p1)*10). O parâmetro do meio identifica o número de neurônios na camada intermediária (400) e na camada de saída (1). A camada de saída deverá ter um neurônio porque o objetivo é o aprendizado através de exemplos para gerar uma função de saída.

```
net = newelm([0 1; 1 (length(p1)*10)],[400 1],{'logsig','logsig'});
```

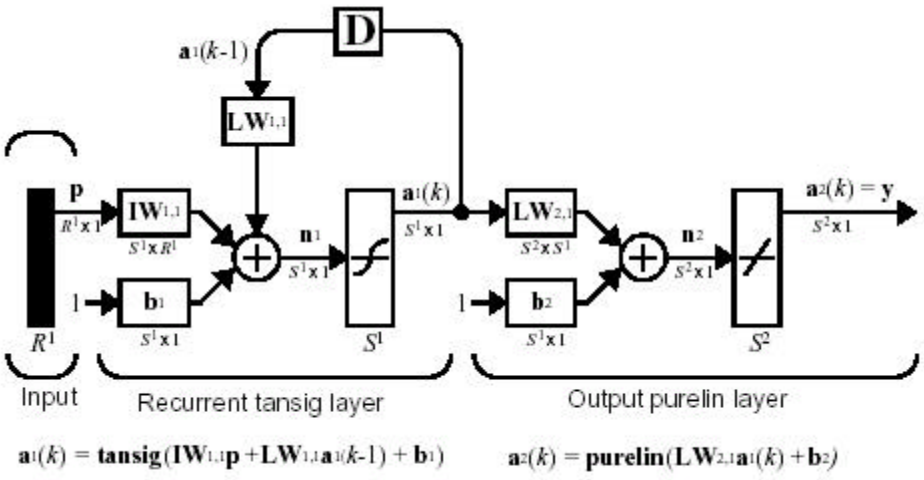


Figura 26. Rede recorrente de Elmann [37]

Os dados foram apresentados a rede de Elmann de forma seqüencial, considerando-se que a ordem dos exemplos é importante. Aplicou-se então, a função *adapt* para o treinamento da rede de Elmann, porque esta função não necessita de um conjunto de saída para o treinamento.

Informa-se a entrada, a função calcula um valor para o erro. A cada entrada é gerado um novo erro, que vai sendo minimizado até alcançar a taxa de erro solicitado, ou que o número de épocas definido seja alcançado. Uma cópia do programa em linguagem *script* do Matlab é fornecida no Anexo I.

No entanto, a rede de Elmann, apesar de ser uma rede neural recorrente não apresentou um resultado satisfatório para o aprendizado com exemplos. A Figura 27 ilustra os valores fornecidos como exemplo ao fundo, e os valores simulados pela rede de Elmann após o treinamento. Acredita-se que tal fato ocorreu porque a realimentação da rede é feita a partir da camada intermediária para ela mesma, de forma a não estabelecer uma saída adequada para o conjunto de entrada fornecido no treinamento. Com isso, partiu-se para o desenvolvimento de uma rede recorrente personalizada.

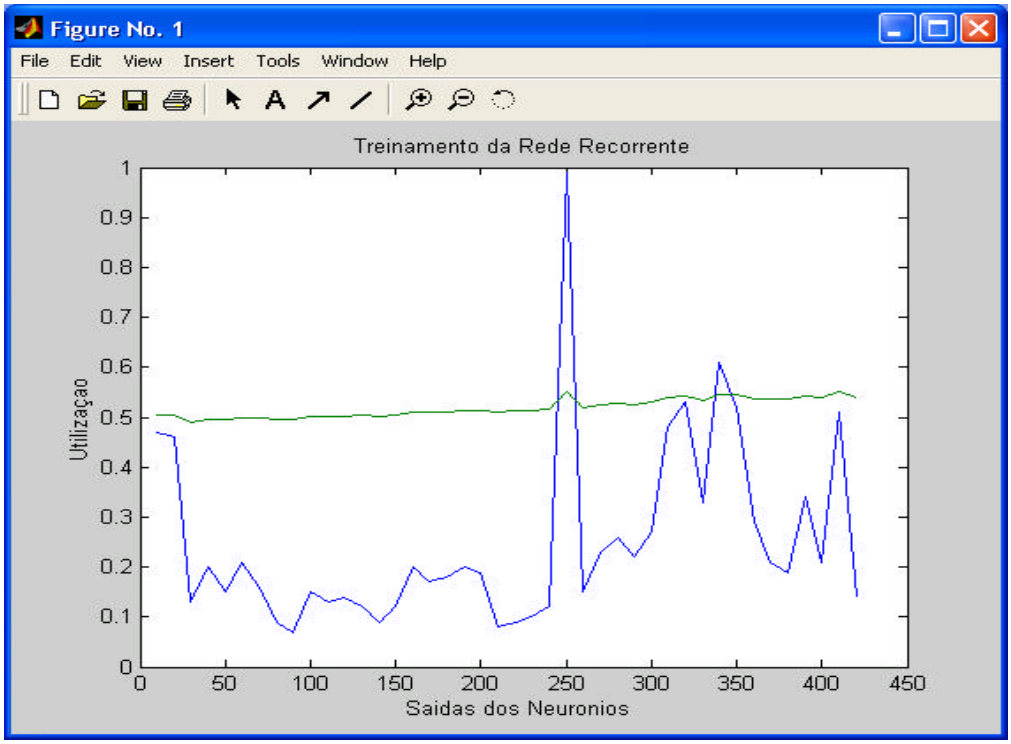


Figura 27. Treinamento para a rede de Elmann

Depois de treinada foram fornecidos novos valores e o resultado da simulação da rede neural pode ser visualizado pela Figura 28. Percebe-se não ser o comportamento apresentado através de exemplos no início do treinamento.

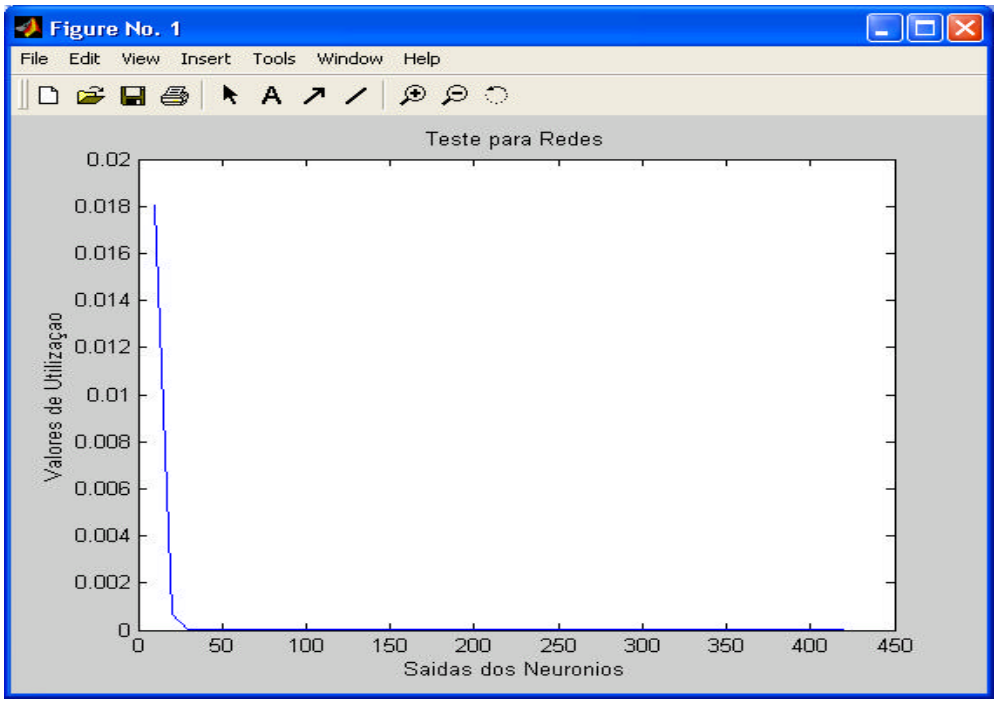


Figura 28. Generalização da rede de Elmann

6.3 Definição do Problema

Seguindo a metodologia, em primeiro lugar é preciso definir o problema. Deseja-se desenvolver uma *baseline* para a gerência pró-ativa de redes de computadores, que descreva o perfil da utilização de uma rede. Considerando que a utilização de uma interface de rede varia conforme o tempo e a carga da rede, tem-se uma saída indefinida. A rede deve então ser treinada com um conjunto de exemplos que represente um perfil da rede. Estes exemplos são coletados do ambiente de rede de forma a extrair os dados da própria rede.

6.3.1 Desenvolvimento de Baselines

A função *baseline* ou perfil da rede é uma das partes mais importantes para realizar uma gerência pró-ativa. Para obtê-la é necessário realizar uma atividade de amostragem durante períodos de tempo, identificando o perfil normal através de médias e cálculos estatísticos. Desta forma, é possível estabelecer um perfil para qualquer tipo de rede. No caso da gerência de desempenho,

uma *baseline* pode ser associada a outras funções, tais como, planejamento de capacidade e estimativa de níveis de tráfego com o objetivo de avaliar opções de conectividade [57].

O desenvolvimento da *baseline* utilizando técnicas de simulação foi apresentado por Schweitzer [93]. Através da modelagem da rede, definiu-se um conjunto de experimentos sob os diversos cenários operacionais e gerenciais. A partir dos resultados desta modelagem é possível definir o conteúdo da *baseline* e as ações corretivas a serem realizadas durante o gerenciamento. Uma *baseline*, portanto, visa produzir uma caracterização estatisticamente válida do comportamento normal da rede sobre um longo período de tempo. Além disso, este perfil da rede deve ser verificado e revisado regularmente para manter-se atualizado mediante trocas de padrões de tráfegos que possam acontecer.

A necessidade de utilização de uma *baseline* consistente para o desenvolvimento de um gerência pró-ativa de redes foi comprovada em [30]. O presente trabalho apresenta uma rede neural recorrente que aprende o perfil da rede através de exemplos de utilização.

Taxa de utilização

A utilização de um recurso é medida através da razão entre o tempo em que o recurso é ocupado, realizando o serviço, e o tempo total decorrido em um determinado período. O período em que o recurso não está sendo utilizado é chamado de "período ocioso". Balancear a carga do sistema de forma que nenhum recurso seja super ou sub-utilizado é uma das características mais importantes desta métrica, esta atividade é conhecida como "planejamento de capacidade". A taxa de utilização é fornecida em porcentagem.

$$\text{Taxa de Utilização}_{j-i} = \frac{(\text{BitsEnviados}_j - \text{BitsEnviados}_i) + (\text{BitsRecebidos}_j - \text{BitsRecebidos}_i)}{\text{Capacidade do canal}}$$

Onde i é o tempo inicial e j é o tempo final da amostra.

6.4 Ambiente de Testes

O ambiente para a realização do testes foi a estrutura de rede da Universidade do Estado de Santa Catarina (UDESC), especificamente, no Centro de Ciências Agroveterinárias (CAV), em

Lages. O campus possui aproximadamente duzentos computadores, os quais estão distribuídos em diversos prédios e laboratórios. Na Figura 29 pode-se observar os principais enlaces que compõe a rede.

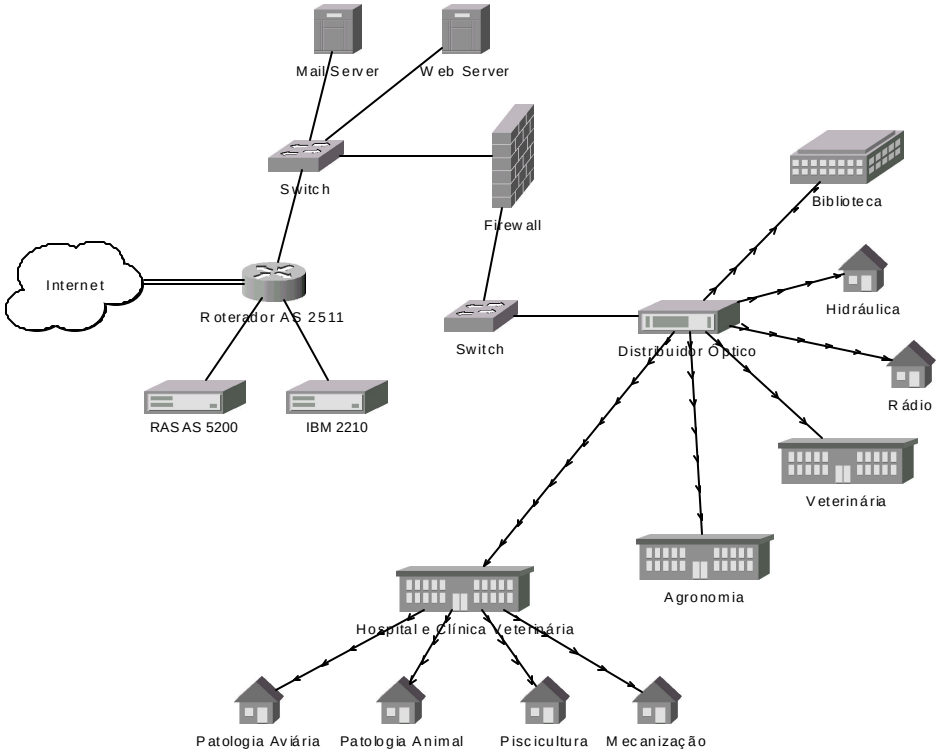


Figura 29. Estrutura da rede do CAV/UDESC

6.4.1 Obtenção dos dados da rede

Inicialmente, para o fornecimento de exemplos de utilização da rede, foi utilizado o aplicativo *netstat –e* para coletar as informações necessárias para o cálculo das taxas de utilização dos recursos de rede do CAV/UDESC.

Foram realizados testes para verificar o aprendizado a partir dos exemplos. Como os testes satisfatórios de aprendizado da rede neural proposta, partiu-se então para a coleta de dados através de um *MIB BROWSER*.

6.4.2 MIB BROWSER

O *MIB Browser* destina-se a fazer a leitura das variáveis da árvore MIB II através do protocolo SNMP. As variáveis são coletadas em agentes gerenciáveis públicos, através do endereço IP do agente, armazenando-as em arquivos de *log*. Estes dados foram utilizados para

O *Mib Browser* foi desenvolvido na linguagem de programação JAVA (J2SDK 1.4.1), o desenvolvimento do programa foi dividido em quatro classes, sendo elas: MibBrowserUn, FrameAjudaSobre, FrameColetarDados, FrameAnalisarLog. A classe MibBrowserUn, é a classe principal do programa, a qual apresenta o acesso para as outras classes. No menu Principal, é possível acessar as classes FrameColetarDados, FrameAnalisarLog, além de outras opções, no menu Ajuda tem a opção para abrir a classe FrameAjudaSobre. A Classe FrameColetarDados é responsável por fazer as leituras dos OIDs na rede, e gerar o arquivo de log. Para seu desenvolvimento além dos pacotes padrões do Java foi utilizado o pacote SNMP, este pacote contém os métodos para a comunicação entre o aplicativo e o agente gerenciáveis. Na coleta dos dados é necessário a entrada dos seguinte dados:

- **IP do Agente:** Endereço IP do agente gerenciável;
- **Community:** Senha de acesso ao agente (ou public);
- **Versão:** Versão SNMP;
- **Time Out:** Tempo limite, em segundos, entre a requisição e o retorno de uma resposta do agente;
- **OID:** Identificação do objeto que se deseja ler o valor;
- **Tempo:** Tempo, em minutos, que o programa vai ficar fazendo a leitura de variáveis. Caso não se queira indicar o tempo, este valor deve ficar em 0. Desta forma o programa só vai parar de executar quando for interrompido;
- **Intervalo:** Intervalo, em minutos, entre uma leitura e outra;
- **Path:** Local onde se deseja gravar o arquivo de log, contendo os resultados das leituras.

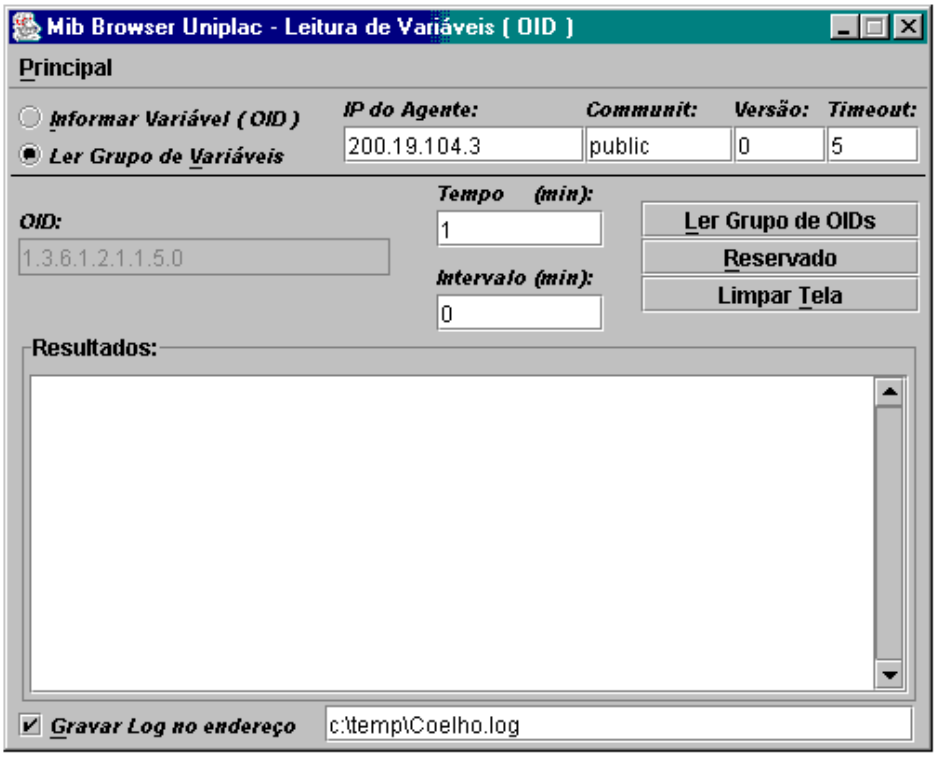


Figura 30. Frame para coletar dados

6.5 Coleta dos Dados

Durante os testes foram feitos estudos de caso envolvendo o MIB Browser para coletar dados e determinar a taxa de utilização, a partir de duas interfaces de rede do roteador CISCO AS-2511 da rede do CAV (Figura 29): a primeira, uma interface *Ethernet* de 10 Mbps e a segunda, uma interface WAN de 2 Mbps, a qual faz a ligação da rede interna com a Internet.

O MIB Browser foi executado em um PC AMD K6-II 500 MHz com 256 MB de memória e sistema operacional GNU/*Linux*, distribuição Conectiva 8.0. O teste descrito a seguir faz uma consulta a MIB, para obtenção de informações de tráfego. Neste teste são consultadas as entradas da MIB, ifInOctets, contendo o valor do contador de tráfego de entrada e ifOutOctets, contendo o valor do contador de tráfego de saída.

Foram feitas 120 amostras a cada dia, intercaladas por cinco minutos, entre os dias 06/12 e 21/12, armazenando-se os valores em arquivos de *log* e posteriormente, obtendo os valores médios (Anexo III).

As entradas da MIB que foram consultadas são apresentadas na Tabela 6.

Tabela 6. Objetos da MIB utilizados para cálculo da taxa de utilização

Identificador (OID)	Descrição	Tipo de Dado (ASN.1)
1.3.6.1.2.1.2.2.1.10	IfInOctets	Counter 32
1.3.6.1.2.1.2.2.1.16	IfOutOctets	Counter 32

6.6 Modelagem da Rede Neural Recorrente

Conforme descrito anteriormente, a Rede de Elmann não se mostrou adequada para o aprendizado através de exemplos. Optou-se por personalizar uma rede neural recorrente com a retroação, partindo da camada de saída para a camada de intermediária.

6.6.1 Definição do problema: desenvolvimento de baselines

Para construir a rede recorrente foi utilizada a classe *network* do MatLab. O trecho abaixo em linguagem *script* descreve a criação de uma rede, com uma entrada (*net.numInputs*), e mais duas camadas (*net.numLayers*), a intermediária e a de saída.

```
% Cria um objeto rede
    net=network;
% Definicao dos parametros de entrada e numero de camadas
    net.numInputs=1;
    net.numLayers=2;
```

O passo seguite é definir as conexões das entradas, as conexões entre as camadas e os bias. As conexões de entrada são definidas pela linha seguinte (*net.inputConnect*), que indica qual camada receberá os valores de entrada. Neste caso, a camada intermediária representada pelo primeiro vetor. A seguir, são definidas as conexões entre as camadas. O primeiro conjunto de valores representa as conexões da camada intermediária, os valores 0 e 1, indicam que haverá uma conexão vinda da segunda camada. Esta conexão será utilizada para a realimentação ou atraso. O segundo conjunto de valores 1 e 1, indica que a camada de saída receberá uma conexão da camada intermediária e outra dela mesma, que também será utilizada como realimentação ou atraso.

```
% Configuracao de  conexoes das entradas, das camadas, das saidas e
dos bias
     net.biasConnect=[0; 0];
     net.inputConnect=[1 0; 0 0];
     net.layerConnect=[0 1; 1 1];
```

As saídas da rede são definidas pelos parâmetros *net.outputConnect* e *net.targetConnect*. Os valores representam a posição da saída, que no caso do exemplo, será a camada de saída. O *target* é utilizado somente quando existe um conjunto de saídas definido.

```
     net.outputConnect=[0 1];
     net.targetConnect=[0 1];
```

A quantidade de valores no vetor de entrada da rede é definido pelo parâmetro net.inputs{1}.range. O número 1 significa que é da entrada 1, quando houver um número maior de entradas na rede. Os valores 0 e 1 indicam que os valores de entrada terão que variar entre 0 e 1, como é o caso da taxa de utilização.

```
      net.inputs{1}.range=[0  1];
```

Após diversos testes, percebeu-se que, com um número maior de neurônios o aprendizado se tornava mais rápido porém não atingia os melhores resultados. Testou-se a rede com 1.000, 500, 100 e 10 neurônios na camada intermediária. Após as análises dos resultados, considerando o tempo de aprendizado e os valores aprendidos pela rede, optou-se por utilizar 500 neurônios na camada intermediária.

```
       % inicializacao dos neuronios da primeira camada – a camada
intermediária
      net.layers{1}.size=500;
      net.layers{1}.transferFcn='logsig';
      net.layers{1}.initFcn='initnw';
%Segunda camada -  a camada de saida
      net.layers{3}.transferFcn='logsig';
      net.layers{3}.initFcn='initnw';
```

A função de transferência foi definida como uma função sigmoidal, que no MatLab é representada por logsig. Escolheu-se esta função porque os valores da taxa de utilização são positivos e variam entre 0 e 1. A função *initnw* é pré-definida pelo MatLab e utiliza o algoritmo de Nguyen-Widrow [37].

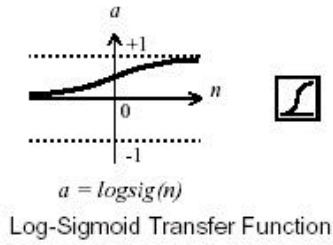


Figura 31. Função de transferência utilizada

Para especificar os atrasos que caracterizam a recorrência da RNA utiliza-se os paarâmetros abaixo. Os índices entre chaves indicam respectivamente quem receberá o atraso e quem envia o atraso. Ou seja, a camada de saída (2) envia um atraso para a camada intermediária (1). E no segunda linha a camada de saída envia um atraso para ela mesma. O valor 1 indica que a saída da função da camada será adicionada como atraso.

```
net.layerWeights{1,2}.delays=1;
net.layerWeights{2,2}.delays=1;
```

Além disso, devem ser configurados os parâmetros para o treinamento por adaptação. Como a rede deverá aprender através de exemplos, deve-se optar pela função *traingdx* no parâmetro net.adaptFcn. A função *traingdx* é uma função de treinamento que atualiza pesos e bias conforme cada entrada apresentada a rede pelo algoritmo de adaptação.

Entre os parâmetros de treinamento que devem ser configurados estão, o valor do erro, o número de épocas de treinamento e o mínimo gradiente.

```
net.adaptParam.goal=1e-20; % Define o erro admitido
net.adaptParam.epochs=1000; %Define o numero de epocas
net.adaptParam.min_grad=1e-20;
```

6.7 RNA recorrente com uma camada

A Figura 32 ilustra a RNA recorrente com uma camada intermediária utilizada para o treinamento através de exemplos. E_1 representa a entrada da rede, IW a ligação de peso da entrada com a

camada intermediária. LW representa uma ligação com pesos entre camadas da RNA. TDL representa as linhas de atraso. Y representa a função de saída da rede.

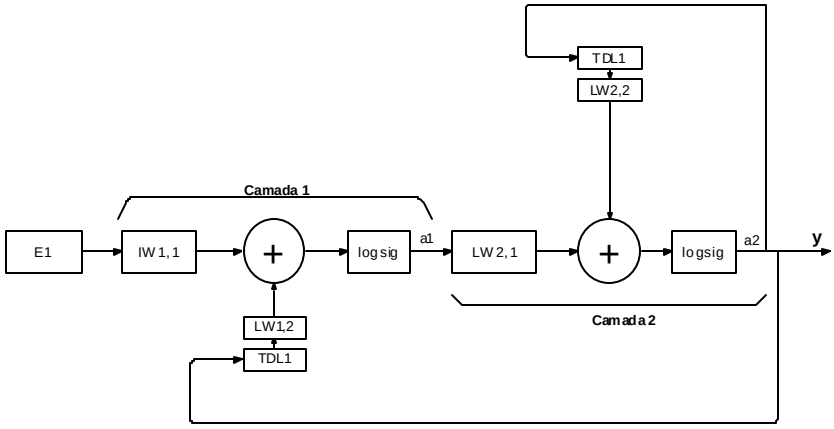


Figura 32. Rede neural recorrente personalizada

6.7.1 Treinamento da RNA recorrente com uma camada

Segue abaixo os valores dos erros do treinamento da rede de uma camada. É possível observar o nome da função utilizada pela adaptação por exemplos (*TRAINGDX*), o número de épocas (o máximo foi configurado para 1000 épocas de treinamento), o valor erro quadrático (MSE – *Mean Square Error*) e o valor do gradiente, configurado para ser da ordem de 1E-20. O algoritmo encerra o treinamento quando uma destas variáveis atingir o valor estipulado para estes critérios.

```
>> TRAINGDX, Epoch 0/1000, MSE 0.183968/1e-020, Gradient 2.33321/1e-020
TRAINGDX, Epoch 25/1000, MSE 0.000669553/1e-020, Gradient 0.0145962/1e-020
TRAINGDX, Epoch 50/1000, MSE 0.000293418/1e-020, Gradient 0.00645595/1e-020
TRAINGDX, Epoch 75/1000, MSE 0.000174766/1e-020, Gradient 0.00386114/1e-020
TRAINGDX, Epoch 100/1000, MSE 7.25779e-005/1e-020, Gradient 0.0016114/1e-020
TRAINGDX, Epoch 125/1000, MSE 2.25734e-005/1e-020, Gradient 0.000503157/1e-020
TRAINGDX, Epoch 150/1000, MSE 6.54082e-006/1e-020, Gradient 0.000146127/1e-020
TRAINGDX, Epoch 175/1000, MSE 1.91171e-006/1e-020, Gradient 4.2761e-005/1e-020
TRAINGDX, Epoch 200/1000, MSE 5.63279e-007/1e-020, Gradient 1.26076e-005/1e-020
TRAINGDX, Epoch 225/1000, MSE 1.66241e-007/1e-020, Gradient 3.72219e-006/1e-020
TRAINGDX, Epoch 250/1000, MSE 4.90699e-008/1e-020, Gradient 1.0989e-006/1e-020
TRAINGDX, Epoch 275/1000, MSE 1.44861e-008/1e-020, Gradient 3.24443e-007/1e-020
TRAINGDX, Epoch 300/1000, MSE 4.27712e-009/1e-020, Gradient 9.57983e-008/1e-020
TRAINGDX, Epoch 325/1000, MSE 1.26295e-009/1e-020, Gradient 2.8288e-008/1e-020
TRAINGDX, Epoch 350/1000, MSE 3.7294e-010/1e-020, Gradient 8.35334e-009/1e-020
TRAINGDX, Epoch 375/1000, MSE 1.10129e-010/1e-020, Gradient 2.46674e-009/1e-020
TRAINGDX, Epoch 400/1000, MSE 3.25214e-011/1e-020, Gradient 7.28433e-010/1e-020
TRAINGDX, Epoch 425/1000, MSE 9.6037e-012/1e-020, Gradient 2.15109e-010/1e-020
TRAINGDX, Epoch 450/1000, MSE 2.83603e-012/1e-020, Gradient 6.35225e-011/1e-020
TRAINGDX, Epoch 475/1000, MSE 8.37495e-013/1e-020, Gradient 1.87585e-011/1e-020
TRAINGDX, Epoch 500/1000, MSE 2.47317e-013/1e-020, Gradient 5.53946e-012/1e-020
TRAINGDX, Epoch 525/1000, MSE 7.30344e-014/1e-020, Gradient 1.63583e-012/1e-020
TRAINGDX, Epoch 550/1000, MSE 2.15675e-014/1e-020, Gradient 4.83067e-013/1e-020
TRAINGDX, Epoch 575/1000, MSE 6.36902e-015/1e-020, Gradient 1.42652e-013/1e-020
TRAINGDX, Epoch 600/1000, MSE 1.88081e-015/1e-020, Gradient 4.21257e-014/1e-020
TRAINGDX, Epoch 625/1000, MSE 5.55415e-016/1e-020, Gradient 1.24399e-014/1e-020
```

```
TRAINGDX, Epoch 650/1000, MSE 1.64017e-016/1e-020, Gradient 3.67356e-015/1e-020
TRAINGDX, Epoch 675/1000, MSE 4.84353e-017/1e-020, Gradient 1.08482e-015/1e-020
TRAINGDX, Epoch 700/1000, MSE 1.43032e-017/1e-020, Gradient 3.20352e-016/1e-020
TRAINGDX, Epoch 725/1000, MSE 4.22382e-018/1e-020, Gradient 9.46012e-017/1e-020
TRAINGDX, Epoch 750/1000, MSE 1.24732e-018/1e-020, Gradient 2.79361e-017/1e-020
TRAINGDX, Epoch 775/1000, MSE 3.6834e-019/1e-020, Gradient 8.24966e-018/1e-020
TRAINGDX, Epoch 800/1000, MSE 1.08773e-019/1e-020, Gradient 2.43616e-018/1e-020
TRAINGDX, Epoch 825/1000, MSE 3.21212e-020/1e-020, Gradient 7.19408e-019/1e-020
TRAINGDX, Epoch 849/1000, MSE 9.95984e-021/1e-020, Gradient 2.23066e-019/1e-020
TRAINGDX, Performance goal met.
elapsed_time =  39.8280
```

A variável *Elapsed-time* apresenta o período de treinamento da rede em segundos, de acordo com o conjunto de entradas apresentado.

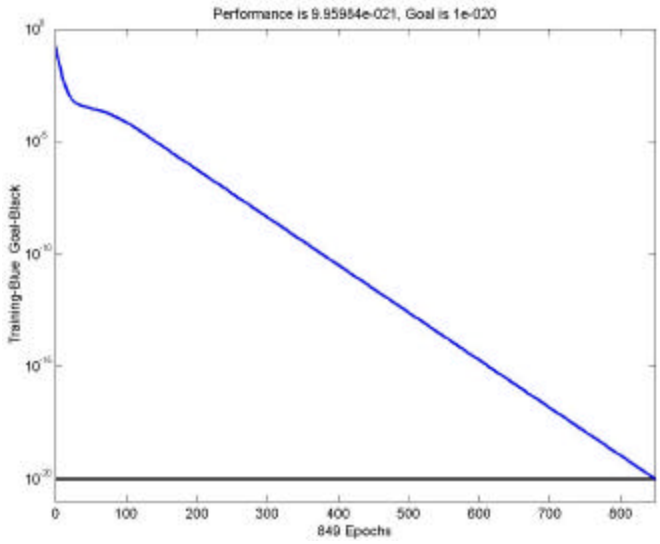


Figura 33. Convergência do erro e número de épocas

6.7.2 Resultados da RNA recorrente com uma camada

A RNA recorrente da Figura 32 foi treinada com amostras de dados coletadas no dia 18 de dezembro. Abaixo tem-se os valores treinados e os valores simulados. As linhas sobrepostas representam que a rede aprendeu os valores fornecidos na entrada.

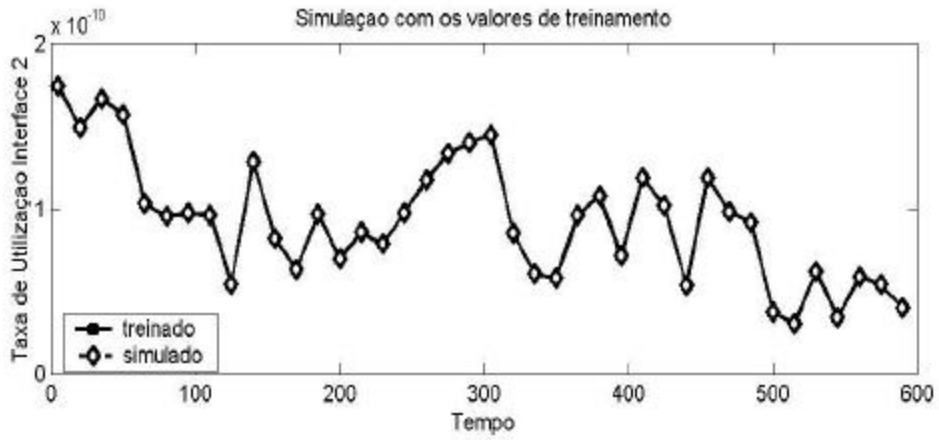


Figura 34. Resultado do treinamento da RNA 1 camada

Em seguida a rede foi simulada para os valores obtidos nos dias 19 e 20 de dezembro. Os valores obtidos pela saída da RNA foram então comparados como saída do programa. É possível perceber que as funções simuladas seguem um mesmo comportamento, observando a escala dos valores obtidos nas duas simulações, as Figuras 39 e 40 apresentam os resultados obtidos.

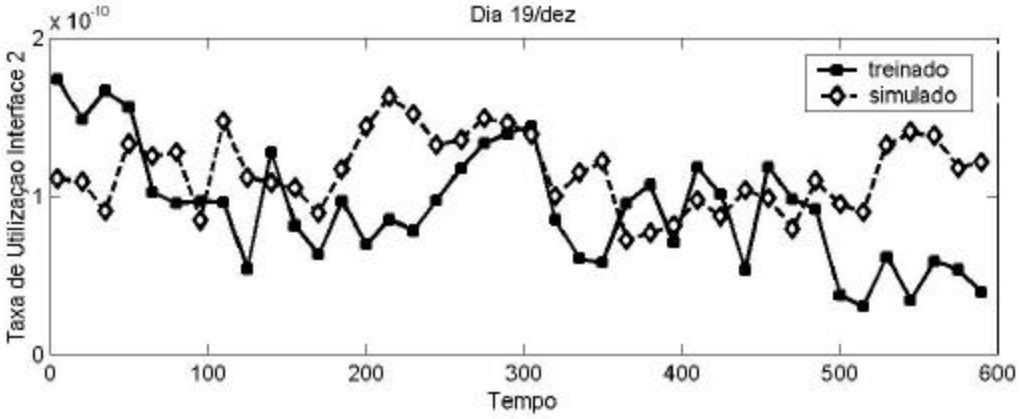


Figura 35. Simulação do dias 19 de dezembro

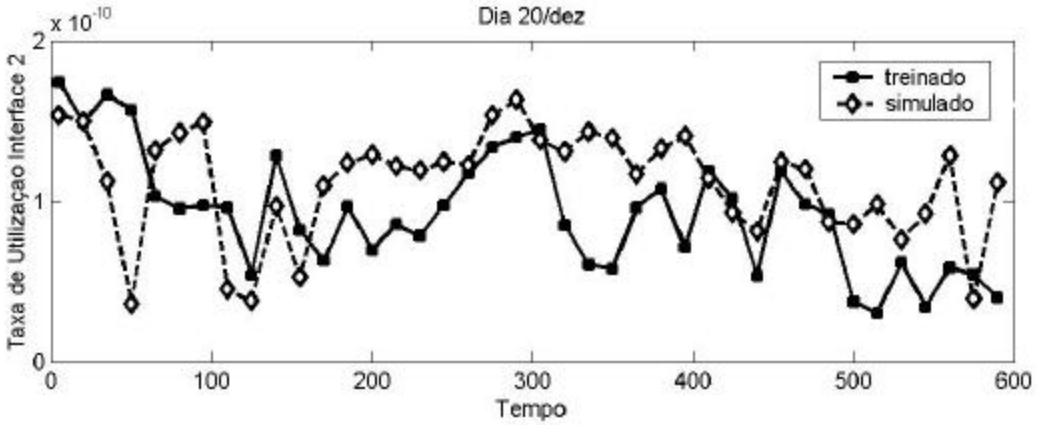


Figura 36. Simulação do dias 20 de dezembro

Em seguida a RNA foi simulada com dados de uma interface diferente da utilizada para o treinamento, com os dados coletados no mesmo período e com a mesma duração. A Figura 41 apresenta a função obtida nesta simulação. Neste caso, observa-se que a função apresenta um comportamento diferente da função obtida no treinamento, como era esperado.

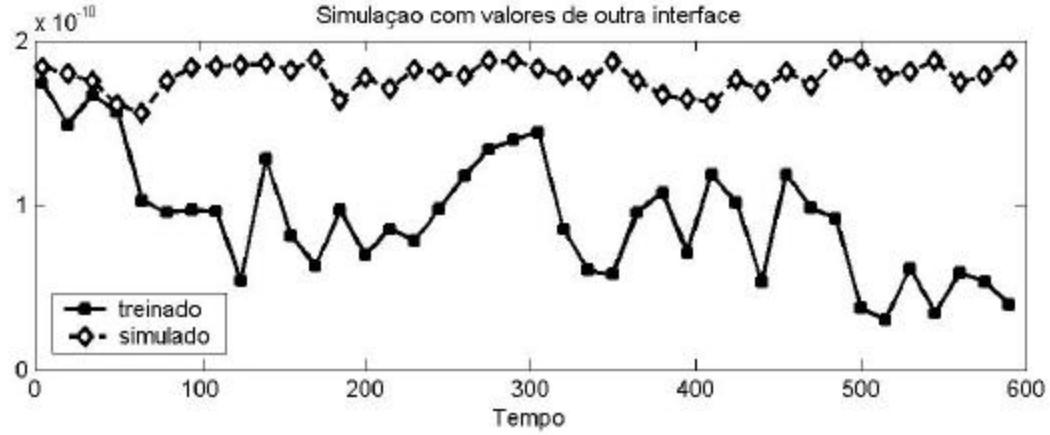


Figura 37. Simulação com amostras de dados de outra interface

6.8 Rede recorrente com três camadas

Como segundo experimento decidiu-se aumentar o número de camadas para verificar redes com desenvolvida uma segunda RNA com três camadas intermediárias, cada uma sendo alimentada por uma entrada diferente. Para o treinamento foram fornecidas amostras de dados coletadas durante os dias 16, 17 e 18 de dezembro. Para esta rede foram criadas quatro

realimentações, da saída para a primeira camada, da saída para a segunda camada, da saída para a terceira camada e da saída para ela mesma, como pode ser observado pela Figura 42.

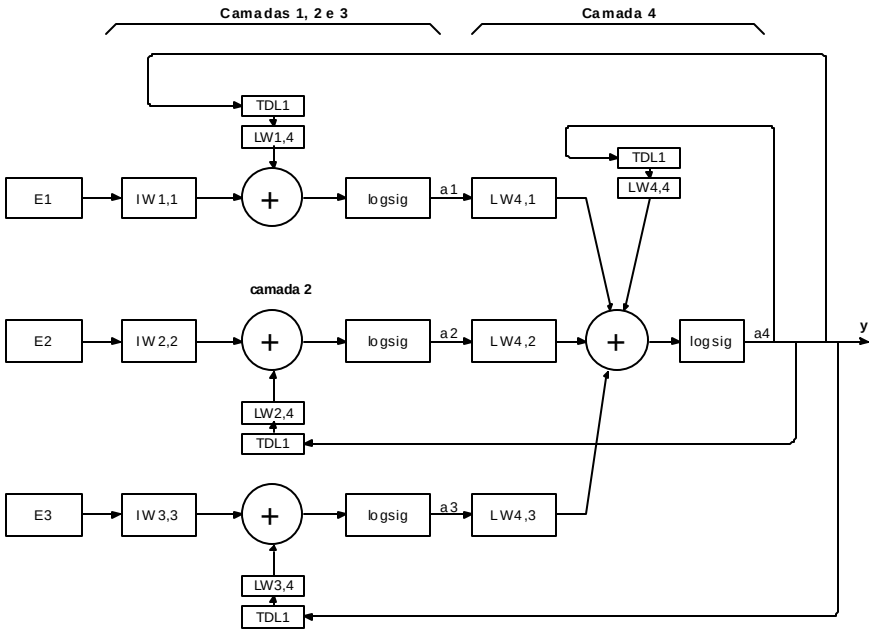


Figura 38. Rede recorrente com três camadas

Para criar a rede com três entradas e quatro camadas, os seguintes parâmetros devem ser alterados.

```
% Configuracao do numero de entradas e de camadas
      net.numInputs=3;
      net.numLayers=4;
```

Em seguida, configura-se as interconexões de entrada e das camadas da RNA recorrente.

```
      net.biasConnect=[0; 0; 0; 0];
      net.inputConnect=[1 0 0; 0 1 0; 0 0 1; 0 0 0];
      net.layerConnect=[0 0 0 1; 0 0 0 1; 0 0 0 1; 1 1 1 1];
      net.outputConnect=[0 0 0 1];
      net.targetConnect=[0 0 0 1];
```

Define-se o número de neurônios em cada entrada, neste caso, foi definido que cada entrada receberia dados de um dia diferente.

```
      net.inputs{1}.range=[0 1];
      net.inputs{2}.range=[0 1];
      net.inputs{3}.range=[0 1];
```

Define-se então, o número de neurônios para as camadas intermediárias. Conforme foi descrito na seção anterior, foi determinado utilizar 500 neurônios. Repete-se estes valores para

todas as camadas intermediárias, alterando-se apenas o índice entre chaves que indica o número da camada.

```
net.layers{1}.size=500;
net.layers{1}.transferFcn='logsig';
net.layers{1}.initFcn='initnw';
net.layers{2}.size=500;
net.layers{2}.transferFcn='logsig';
net.layers{2}.initFcn='initnw';
net.layers{3}.size=500;
net.layers{3}.transferFcn='logsig';
net.layers{3}.initFcn='initnw';
```

Para a camada de saída , define-se a função de ativação, que neste caso é logsig, conforme descrito na seção anterior.

```
net.layers{4}.transferFcn='logsig';
net.layers{4}.initFcn='initnw';
```

Por último definem-se os atrasos que caracterizarão o dinamismo da RNA recorrente.

```
net.layerWeights{1,4}.delays=1;
net.layerWeights{2,4}.delays=1;
net.layerWeights{3,4}.delays=1;
net.layerWeights{4,4}.delays=1;
```

6.8.1 Treinamento da RNA recorrente com três camadas

Da mesma forma que a RNA anterior foi utilizada a função de treinamento (*TRAINGDX*), o número de épocas (configurado para 1000 épocas de treinamento), o valor erro quadrático MSE e o valor do gradiente, configurado para ser da ordem de 1E-20. O algoritmo pode parar o treinamento quando uma destas condições atingir o valor estipula do.

```
>> TRAINGDX, Epoch 0/1000, MSE 1.97768e-009/1e-020, Gradient 7.7337e-008/1e-020
TRAINGDX, Epoch 25/1000, MSE 1.97768e-009/1e-020, Gradient 7.73369e-008/1e-020
TRAINGDX, Epoch 50/1000, MSE 1.97767e-009/1e-020, Gradient 7.73366e-008/1e-020
TRAINGDX, Epoch 75/1000, MSE 1.97765e-009/1e-020, Gradient 7.73357e-008/1e-020
TRAINGDX, Epoch 100/1000, MSE 1.97757e-009/1e-020, Gradient 7.73327e-008/1e-020
TRAINGDX, Epoch 125/1000, MSE 1.97731e-009/1e-020, Gradient 7.73224e-008/1e-020
TRAINGDX, Epoch 150/1000, MSE 1.97642e-009/1e-020, Gradient 7.72877e-008/1e-020
TRAINGDX, Epoch 175/1000, MSE 1.97342e-009/1e-020, Gradient 7.71701e-008/1e-020
TRAINGDX, Epoch 200/1000, MSE 1.9633e-009/1e-020, Gradient 7.67744e-008/1e-020
TRAINGDX, Epoch 225/1000, MSE 1.9297e-009/1e-020, Gradient 7.54605e-008/1e-020
TRAINGDX, Epoch 250/1000, MSE 1.8231e-009/1e-020, Gradient 7.1292e-008/1e-020
TRAINGDX, Epoch 275/1000, MSE 1.52911e-009/1e-020, Gradient 5.97954e-008/1e-020
TRAINGDX, Epoch 300/1000, MSE 9.6411e-010/1e-020, Gradient 3.77007e-008/1e-020
TRAINGDX, Epoch 325/1000, MSE 3.9973e-010/1e-020, Gradient 1.56306e-008/1e-020
TRAINGDX, Epoch 350/1000, MSE 1.24656e-010/1e-020, Gradient 4.87422e-009/1e-020
TRAINGDX, Epoch 375/1000, MSE 3.63541e-011/1e-020, Gradient 1.42142e-009/1e-020
TRAINGDX, Epoch 400/1000, MSE 1.06705e-011/1e-020, Gradient 4.17188e-010/1e-020
TRAINGDX, Epoch 425/1000, MSE 3.15066e-012/1e-020, Gradient 1.23175e-010/1e-020
TRAINGDX, Epoch 450/1000, MSE 9.30873e-013/1e-020, Gradient 3.63906e-011/1e-020
TRAINGDX, Epoch 475/1000, MSE 2.74944e-013/1e-020, Gradient 1.07478e-011/1e-020
TRAINGDX, Epoch 500/1000, MSE 8.11996e-014/1e-020, Gradient 3.17399e-012/1e-020
TRAINGDX, Epoch 525/1000, MSE 2.39808e-014/1e-020, Gradient 9.3733e-013/1e-020
TRAINGDX, Epoch 550/1000, MSE 7.08236e-015/1e-020, Gradient 2.76811e-013/1e-020
TRAINGDX, Epoch 575/1000, MSE 2.09166e-015/1e-020, Gradient 8.17474e-014/1e-020
TRAINGDX, Epoch 600/1000, MSE 6.17739e-016/1e-020, Gradient 2.41415e-014/1e-020
```

```
TRAINGDX, Epoch 625/1000, MSE 1.82439e-016/1e-020, Gradient 7.12943e-015/1e-020
TRAINGDX, Epoch 650/1000, MSE 5.38804e-017/1e-020, Gradient 2.10545e-015/1e-020
TRAINGDX, Epoch 675/1000, MSE 1.59127e-017/1e-020, Gradient 6.21778e-016/1e-020
TRAINGDX, Epoch 700/1000, MSE 4.69953e-018/1e-020, Gradient 1.83622e-016/1e-020
TRAINGDX, Epoch 725/1000, MSE 1.38792e-018/1e-020, Gradient 5.42269e-017/1e-020
TRAINGDX, Epoch 750/1000, MSE 4.09899e-019/1e-020, Gradient 1.60141e-017/1e-020
TRAINGDX, Epoch 775/1000, MSE 1.21056e-019/1e-020, Gradient 4.72926e-018/1e-020
TRAINGDX, Epoch 800/1000, MSE 3.57518e-020/1e-020, Gradient 1.39663e-018/1e-020
TRAINGDX, Epoch 825/1000, MSE 1.05586e-020/1e-020, Gradient 4.1245e-019/1e-020
TRAINGDX, Epoch 827/1000, MSE 9.57706e-021/1e-020, Gradient 3.74105e-019/1e-020
TRAINGDX, Performance goal met.
elapsed_time = 106.3910
```

Neste caso, a variável *Elapsed-time* apresentou o tempo de 106,391 segundos para aprender o conjunto de entradas apresentado. A Figura 39 ilustra a convergência do erro pelo número de épocas treinadas durante o aprendizado por exemplos.

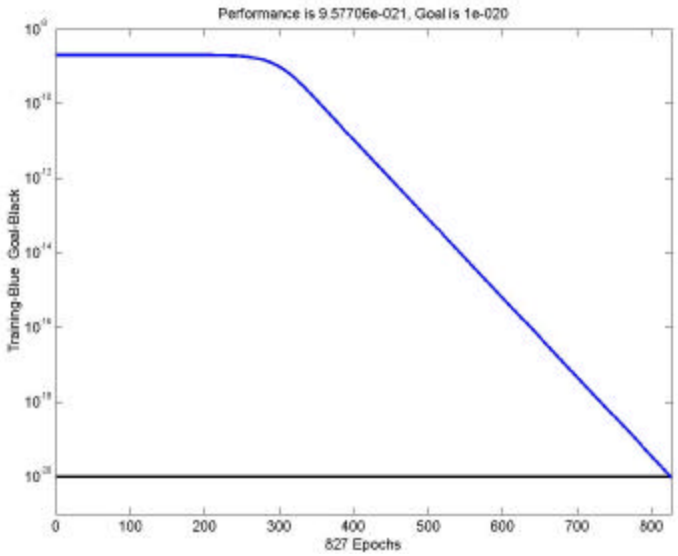


Figura 39. Convergência do erro e número de épocas

6.8.2 Resultados da RNA recorrente com três camadas

Os valores foram treinados com dados dos dias 16, 17 e 18 de dezembro. A Figura 40 apresenta o resultado dos valores treinados que posteriormente foram simulados após o treinamento. As linhas sobrepostas ilustram que a rede aprendeu os valores fornecidos como entrada.

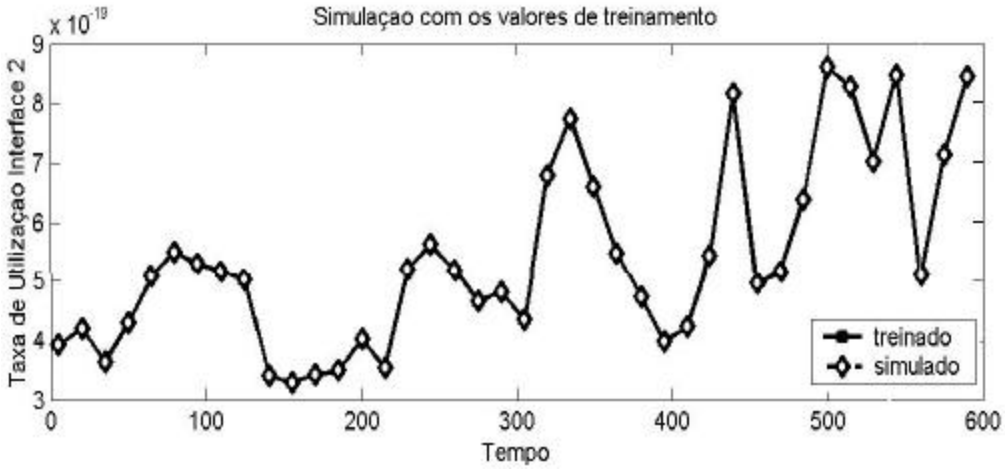


Figura 40. Resultado do treinamento da RNA – 3 camadas

Nos gráficos das Figuras 41 e 42 foram simulados amostras de dados obtidos da mesma interface de rede nos dias 19 e 20 de dezembro. Observa-se também, que estes conjuntos de dados seguem o mesmo comportamento da função de treinamento da RNA recorrente.

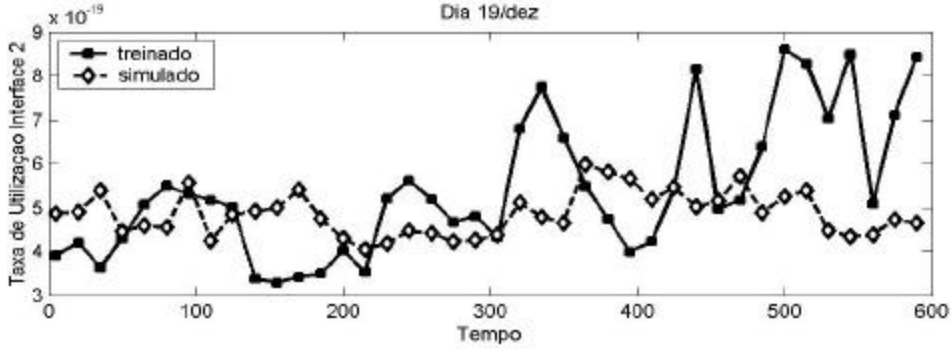


Figura 41. Simulação da rede dia 19/dez

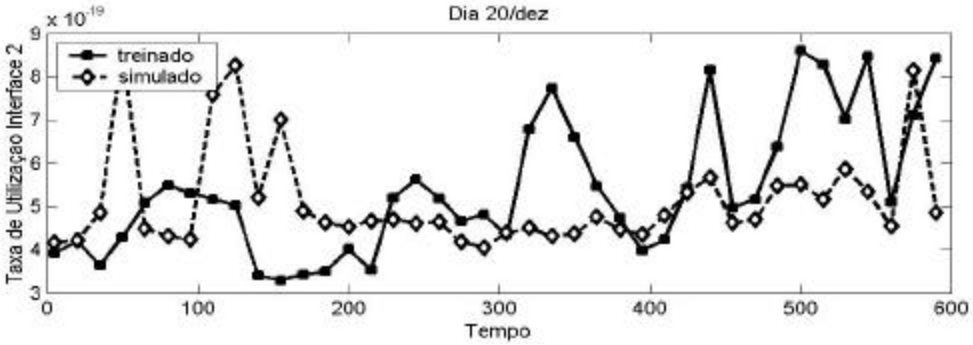


Figura 42. Simulação da rede dia 20/dez

Outro experimento realizado com a RNA de três camadas foi fornecer dados de uma interface diferente da utilizada para o treinamento. A Figura 43 ilustra que as funções apresentam comportamento diferente, como era novamente esperado.

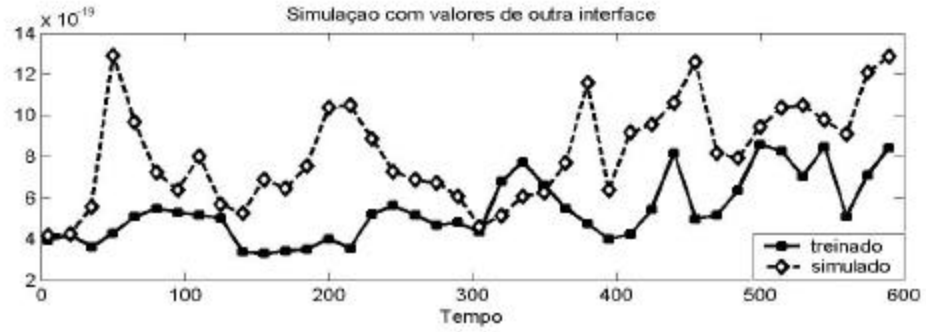


Figura 43. Simulação com amostras de dados de outra interface

6.9 Rede recorrente com cinco camadas

A Figura 44 ilustra a topologia da RNA recorrente com cinco entradas, cinco camadas intermediárias e uma saída.

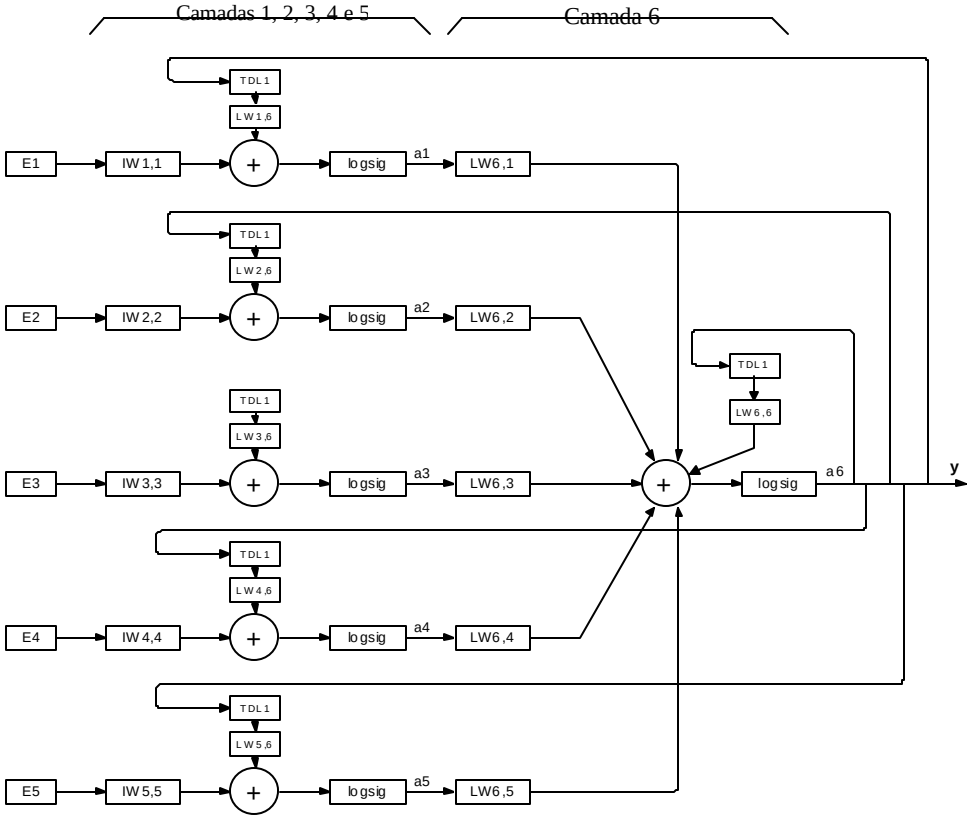


Figura 44. Rede recorrente com cinco camadas

Para o treinamento neste experimento, foram fornecidas amostras de dados coletadas durante os dias 16, 17, 18, 19 e 20 de dezembro. Para esta rede foram criadas seis realimentações, da saída para a primeira camada, da saída para a segunda camada, da saída para a terceira camada, da saída para a quarta camada, da saída para a quinta camada e da saída para ela mesma, como pode ser observado pela Figura 44.

A RNA recorrente foi treinada com dados dos dias 16, 17, 18, 19 e 20 de dezembro e posteriormente simulados, conforme pode ser observado utilizando a função TRAINGDX.

```
>> TRAINGDX, Epoch 0/1000, MSE 0.223285/1e-020, Gradient 5.12968/1e-020
TRAINGDX, Epoch 25/1000, MSE 2.08008e-011/1e-020, Gradient 1.04317e-009/1e-020
TRAINGDX, Epoch 50/1000, MSE 3.43906e-012/1e-020, Gradient 1.7247e-010/1e-020
TRAINGDX, Epoch 75/1000, MSE 3.02222e-012/1e-020, Gradient 1.51565e-010/1e-020
TRAINGDX, Epoch 100/1000, MSE 2.99431e-012/1e-020, Gradient 1.50166e-010/1e-020
TRAINGDX, Epoch 125/1000, MSE 2.99232e-012/1e-020, Gradient 1.50066e-010/1e-020
TRAINGDX, Epoch 150/1000, MSE 2.99217e-012/1e-020, Gradient 1.50058e-010/1e-020
TRAINGDX, Epoch 175/1000, MSE 2.99215e-012/1e-020, Gradient 1.50057e-010/1e-020
TRAINGDX, Epoch 200/1000, MSE 2.99211e-012/1e-020, Gradient 1.50055e-010/1e-020
TRAINGDX, Epoch 225/1000, MSE 2.99198e-012/1e-020, Gradient 1.50049e-010/1e-020
TRAINGDX, Epoch 250/1000, MSE 2.99154e-012/1e-020, Gradient 1.50027e-010/1e-020
TRAINGDX, Epoch 275/1000, MSE 2.99005e-012/1e-020, Gradient 1.49952e-010/1e-020
TRAINGDX, Epoch 300/1000, MSE 2.98501e-012/1e-020, Gradient 1.49699e-010/1e-020
TRAINGDX, Epoch 325/1000, MSE 2.96806e-012/1e-020, Gradient 1.48849e-010/1e-020
TRAINGDX, Epoch 350/1000, MSE 2.9119e-012/1e-020, Gradient 1.46033e-010/1e-020
TRAINGDX, Epoch 375/1000, MSE 2.73494e-012/1e-020, Gradient 1.37158e-010/1e-020
TRAINGDX, Epoch 400/1000, MSE 2.25643e-012/1e-020, Gradient 1.13161e-010/1e-020
TRAINGDX, Epoch 425/1000, MSE 1.37684e-012/1e-020, Gradient 6.90485e-011/1e-020
TRAINGDX, Epoch 450/1000, MSE 5.52303e-013/1e-020, Gradient 2.7698e-011/1e-020
TRAINGDX, Epoch 475/1000, MSE 1.70103e-013/1e-020, Gradient 8.53063e-012/1e-020
TRAINGDX, Epoch 500/1000, MSE 4.95807e-014/1e-020, Gradient 2.48645e-012/1e-020
TRAINGDX, Epoch 525/1000, MSE 1.4563e-014/1e-020, Gradient 7.30327e-013/1e-020
TRAINGDX, Epoch 550/1000, MSE 4.30033e-015/1e-020, Gradient 2.15658e-013/1e-020
TRAINGDX, Epoch 575/1000, MSE 1.27039e-015/1e-020, Gradient 6.37087e-014/1e-020
TRAINGDX, Epoch 600/1000, MSE 3.75181e-016/1e-020, Gradient 1.88148e-014/1e-020
TRAINGDX, Epoch 625/1000, MSE 1.10791e-016/1e-020, Gradient 5.55602e-015/1e-020
TRAINGDX, Epoch 650/1000, MSE 3.27171e-017/1e-020, Gradient 1.6407e-015/1e-020
TRAINGDX, Epoch 675/1000, MSE 9.66152e-018/1e-020, Gradient 4.84506e-016/1e-020
TRAINGDX, Epoch 700/1000, MSE 2.8531e-018/1e-020, Gradient 1.43077e-016/1e-020
TRAINGDX, Epoch 725/1000, MSE 8.42536e-019/1e-020, Gradient 4.22511e-017/1e-020
TRAINGDX, Epoch 750/1000, MSE 2.48805e-019/1e-020, Gradient 1.24769e-017/1e-020
TRAINGDX, Epoch 775/1000, MSE 7.34736e-020/1e-020, Gradient 3.68449e-018/1e-020
TRAINGDX, Epoch 800/1000, MSE 2.16971e-020/1e-020, Gradient 1.08804e-018/1e-020
TRAINGDX, Epoch 816/1000, MSE 9.93977e-021/1e-020, Gradient 4.98447e-019/1e-020
TRAINGDX, Performance goal met.
elapsed_time = 171.5940
```

A Figura 45 apresenta a convergência do erro e o número de épocas necessárias no treinamento da RNA recorrente proposta.

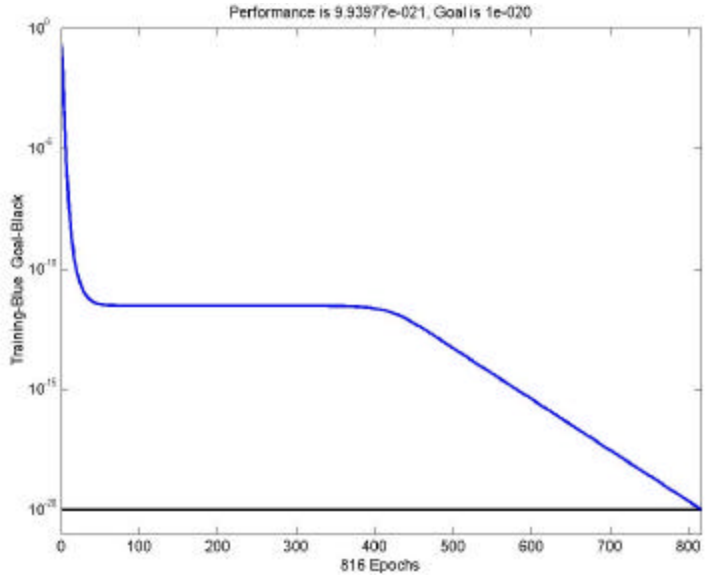


Figura 45. Convergência do erro e número de épocas

A Figura 46 apresenta a função obtida no treinamento e a função obtida na simulação com os mesmos dados de treinamento, comprovando o aprendizado.

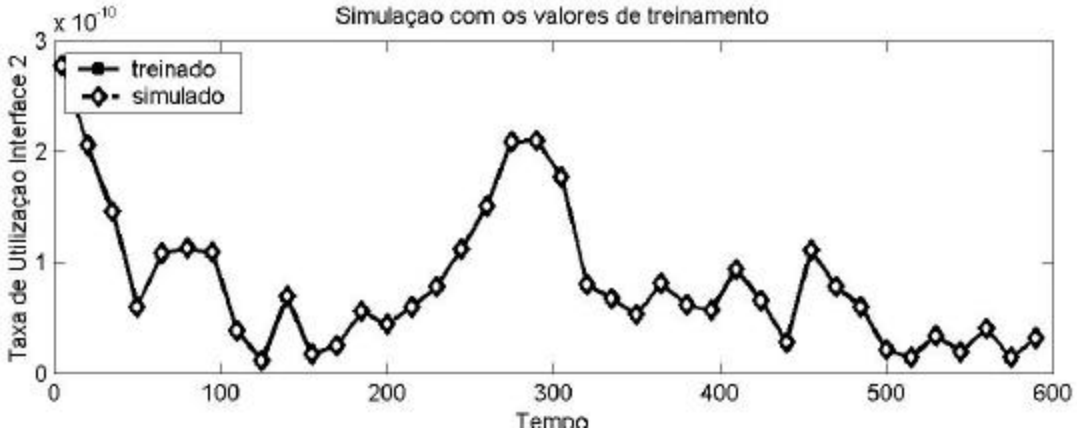


Figura 46. Resultado do treinamento da RNA – 5 camadas

6.9.1 Resultados da RNA recorrente com cinco camadas

Em seguida, a rede foi simulada com dados do dia 05 de dezembro. A Figura 47 ilustra as funções obtidas neste experimento.

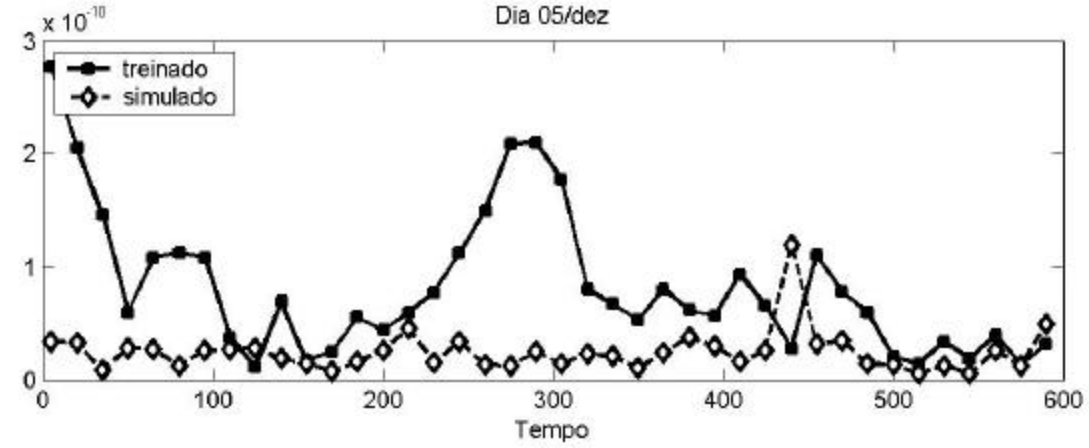


Figura 47. Simulação da rede dia 05/dez

No experimento seguinte a rede foi simulada com os dados dos dias 05, 06, 07, 10 e 12 dezembro. Nesta simulação, a função simulada refere-se ao conjunto global dos dados nos referidos dias, a qual é demonstrada através da Figura 48.

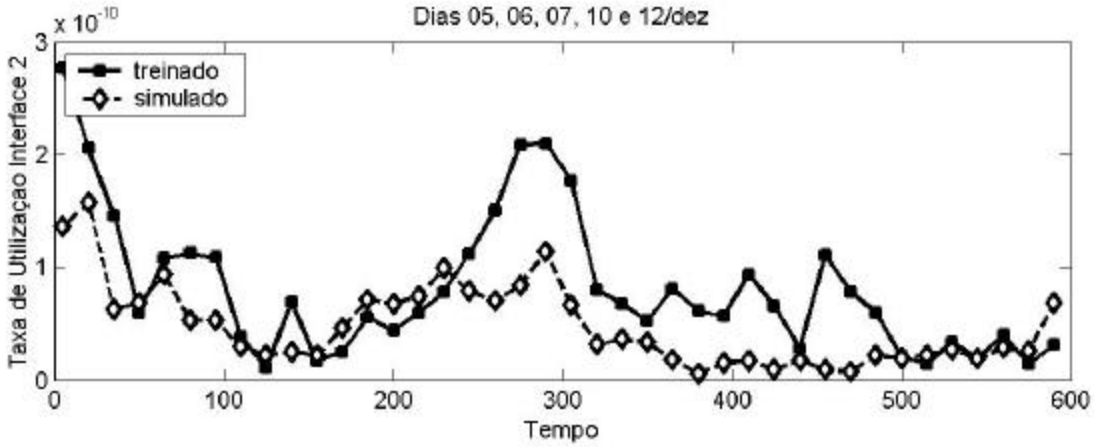


Figura 48. Simulação da rede dia 05, 06, 07, 10 e 12/dez

Observa-se que mesmo fornecendo dados de dias diferentes, a função simulada apresentou comportamento semelhante ao da função treinada. Novamente, para realmente comprovar o aprendizado do perfil de utilização da interface analisada, a RNA recorrente foi

simulada com dados de uma interface diferente da utilizada no treinamento. Neste caso, obteve-se comportamentos diferentes entre as funções simuladas e treinadas (Figura 49).

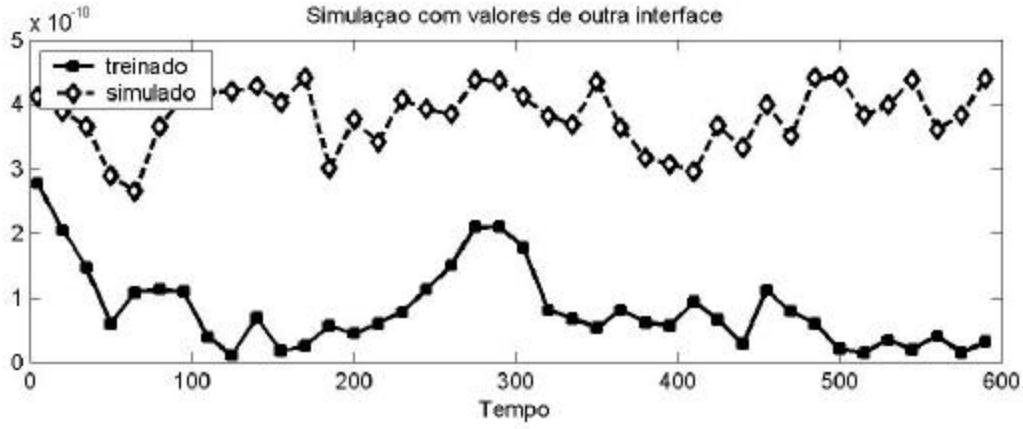


Figura 49. Simulação com amostras de dados de outra interface

6.10 Comparação com o método tradicional

Para desenvolver uma *baseline*, cálculos estatísticos (principalmente médias comparando as amostras de dados) são necessários para encontrar um perfil que represente a utilização de uma interface de rede de computadores. Treinada a rede com os exemplos obteve-se uma função de representação do perfil de utilização. A principal vantagem neste caso, é que o treinamento da rede não é um processo demorado. Para a RNA de uma camada levou 39.82s para o aprendizado de amostras de dados no período de 8 e 18h (cerca de 120). Para a RNA com três camadas (foram fornecidos 3 entradas de 120 amostras de dias diferentes), o período de treinamento foi 106.39s e para a RNA recorrente com 5 camadas levou 171.59s (foram fornecidos 5 entradas diferente de 120 amostras).

Conclui-se através dos experimentos, que as RNAs recorrentes podem ser utilizadas para o desenvolvimento de funções de *baseline*. No entanto, nem todas as topologias recorrentes podem ser utilizadas para o aprendizado através de exemplos. A rede recorrente de Elmann não apresentou bons resultados, talvez pelo fato da realimentação partir da camada intermediária para ela mesma, implicando na falta de capacidade de convergência deste tipo de RNA em situações em que a saída não é conhecida. Nas RNAs recorrentes desenvolvidas acrescentou-se realimentações da camada de saída para a camada intermediária. Foram testadas RNAs com

uma, três e cinco camadas. Os três tipos de RNAs recorrentes foram capazes de aprender o conjunto de exemplos fornecidos como entrada da rede.

Capítulo 7

Conclusões

7.1 Considerações Finais

Este trabalho apresentou uma metodologia para aplicação de técnicas de IA para auxiliar na solução de problemas na área da gerência de redes. Existem casos na literatura científica em que existe uma má aplicação das redes neurais artificiais. É comum encontrar aplicações de RNAs diretas (estáticas) na solução de problemas dinâmicos. A essência do trabalho foi explorar os dois tipos de soluções (dinâmicas ou estáticas) para problemas distribuídos, e aplicar as técnicas conforme o tipo de problema a ser abordado. Esta metodologia serve para auxiliar o desenvolvimento de agentes inteligentes na área de gerência de redes.

Conceitos importantes da IAC foram apresentados: modelos de neurônios, algumas das topologias das RNAs, aprendizados supervisionado e não supervisionado e RNAs recorrentes. Uma breve comparação entre RNAs diretas e RNAs recorrentes foi descrita. Outro aspecto importante abordado foi a capacidade das RNAS na solução de problemas complexos. Ao fato que um problema possa ser resolvido por recursos finitos chama-se *computabilidade* e a quantidade de recursos envolvidos chama-se *complexidade*. Neste sentido, apresentou-se alguns teoremas que fazem parte da teoria da complexidade conexionista.

A seguir, foram apresentados conceitos sobre gerenciamento de sistemas, principais funções da gerência de redes - FCAPS, o protocolo de gerência de redes da Internet (que normalmente, é o mais utilizado). Atribui-se a dificuldade em automatizar o processo de gerência à características como: heterogeneidade, complexidade, dinamismo e a natureza distribuída. Ainda hoje, esta atividade depende muito da atividade do administrador de redes. Existem dois

comportamentos distintos identificados nas atividades de gerência. O reativo, quando as ações são determinadas após a queda da rede. E o pró-ativo que envolve uma série de ações que previnem a queda da rede, evitando a degradação e a indisponibilidade do sistema. Cada um desses comportamentos exigem soluções apropriadas. As técnicas conexionistas podem auxiliar na tentativa de automatizar o processo de gerência. Quando utilizar RNAs diretas e RNAs recorrentes na solução de problemas da área de redes, também foram apresentadas.

No sentido de definir sistemas estáticos e sistemas dinâmicos, foram relatados alguns exemplos da teoria geral de sistemas. A teoria define sistema geral, sistema orientado, sistema funcional, sistema temporal, sistema dinâmico e sistema complexo. Exemplos foram fornecidos para ilustrar estas definições.

Na área de gerência de redes, é comum utilizar o termo agentes e gerentes para descrever processos utilizados nas atividades de gerência. Uma taxonomia descrevendo o uso de agentes de software inteligentes foi proposta. Como a aplicação de técnicas de IA pode influenciar no comportamento de agentes foi considerado. Os agentes foram classificados como passivos, ativos e autônomos. Agentes passivos não utilizam técnicas de IA e são baseados em instruções, é o caso dos agentes de gerência de redes. Agentes ativos e agentes autônomos utilizam técnicas de IA. Os agentes ativos são desenvolvidos com o auxílio de heurísticas. Na área de gerência de redes, estas heurísticas são fornecidas pelo administrador da rede. Servem para definir regras de produção ou redes neurais diretas para auxiliar na solução dos problemas. Assemelham-se aos sistemas especialistas e não possuem autonomia. Agentes autônomos possuem características dinâmicas. Devem ser desenvolvidos com o auxílio de técnicas que satisfaçam esta característica. Podem ser desenvolvidos com o auxílio de RNAs recorrentes ou através de técnicas de raciocínio baseado em casos. Estes conceitos foram considerados na metodologia de aplicação de técnicas de IA no desenvolvimento de agentes de gerência de redes. Se o problema for bem definido, existem heurísticas para auxiliar na solução do problema. Ferramentas com características estáticas podem ser utilizadas para solucionar este tipo de problema, tais como, regras de produção (IA Simbólica) ou redes neurais diretas (IA Conexionista). No entanto, se o problema não for bem definido, só poderá ser “bem” resolvido utilizando ferramentas com características dinâmicas. Neste caso, utilizam-se exemplos para a representação do problema. Alguns trabalhos relacionados que aplicam outras técnicas para a

solução de problemas na área de gerência de redes foram descritos. Bem como, algumas plataformas para o desenvolvimento de agentes inteligentes.

A metodologia foi aplicada no desenvolvimento de *baselines* para a gerência pró-ativa de redes. No método tradicional de se obter uma função *baseline* é necessário realizar uma atividade de amostragem durante vários períodos de tempo, identificando o desempenho normal da rede através de médias e cálculos estatísticos. Estabelecido o perfil da rede, utiliza-se a função para comparar o funcionamento atual com o estabelecido pelo perfil da rede. Com base nestas comparações é possível realizar um gerenciamento preventivo. Outra maneira de se obter uma função de *baseline* é através de técnicas de IA.

Foram testadas três RNAs recorrentes com diferentes números de entradas e de camadas intermediárias para testar o aprendizado através de exemplos. Foram fornecidas amostras de dados de utilização da rede no sentido de verificar a utilização das RNAs recorrentes no desenvolvimento de *baselines*. Através dos experimentos comprovou-se que o uso de RNAs recorrentes é adequado para encontrar uma função que representa o perfil de utilização da rede de computadores. A RNA com uma camada apresentou menor tempo de aprendizado porém o perfil é delineado sobre um dia de amostra de dados. Entre três camadas e cinco camadas, houve pouca diferença entre os tempos de treinamento. Nos três casos, a RNA levou cerca de 800 épocas para o aprendizado dos exemplos. A generalização da RNA recorrente pode ser comprovada pelas simulações de dias que não foram utilizados durante o treinamento. A margem de erros entre a função treinada e a função simulada apresentou pouca variação, como podem ser observados nos resultados.

Conclui-se que as RNAs recorrentes podem ser utilizadas para o desenvolvimento de funções de *baselines*. No entanto, nem todas as topologias recorrentes são adequadas para o aprendizado através de exemplos. A rede recorrente de Elmann não apresentou bons resultados, acredita-se que este resultado deve-se ao fato da realimentação ser da camada intermediária para ela mesma, implicando na falta de capacidade de convergência deste tipo de RNA em situações em que a saída não é conhecida. Nas RNAs recorrentes desenvolvidas acrescentou-se realimentações da camada de saída para a camada intermediária. Os três tipos de RNAs recorrentes foram capazes de aprender o conjunto de exemplos fornecidos como entrada da rede.

Os pontos originais deste trabalho que merecem destaque são:

- Seguindo a linha de Azevedo [27], que utilizou a teoria de sistemas para formalizar as RNAs, e Roisenberg que utilizou a mesma ferramenta para apresentar um método formal de AAs. Foi formalizado o conceito de multi-agentes como um sistema complexo, que é um conjunto de sistemas dinâmicos interligados;
- Foi proposta uma taxonomia para o paradigma de agentes no sentido de ilustrar diferentes comportamentos quando existe a aplicação de técnicas de IA.
- No sentido de contribuir com processo de automação da gerência de redes apresentou-se uma metodologia para aplicação de técnicas de IA no desenvolvimento de agentes na área de gerência de redes.
- Aplicação de RNAs recorrentes para o desenvolvimento de *baselines* de gerência pró-ativa de redes. Comprovando o aprendizado através de exemplos, obtendo como saída uma função que represente uma possível solução para o problema abordado.

7.2 Trabalhos Futuros

Com base na metodologia apresentada neste trabalho será possível o desenvolvimento dos seguintes trabalhos:

- Construir uma plataforma de gerência de redes inteligente, de forma que os agentes e gerentes sejam desenvolvidos utilizando técnicas de IA híbrida. As RNAs recorrentes possam ser aplicadas no desenvolvimento de agentes destinados a solucionar problemas mal definidos. E as RNAs diretas ou regras de produção sejam aplicadas a problemas conhecidos, definidos por heurísticas obtidas dos adiministradores de redes.
- Desenvolver um pacote em JAVA para a aplicação de técnicas de IA no desenvolvimento de agentes. Testar o desenvolvimento destes agentes nas plataformas para sistemas multi-agentes apresentadas no trabalho. Até o presente momento, os agentes desenvolvidos nestas plataformas foram testados somente com o uso de regras de produção (através do JESS).

- Empresas de pequeno e médio porte muitas vezes desconsideram as atividades de gerência de redes. Seria interessante relacionar soluções de problemas de redes de computadores às tomadas de decisões, no caso de prevenção de falhas e segurança.
- Além disso, a partir deste trabalho será possível pesquisar outras técnicas de IA que possam ser aplicadas na solução de problemas na área de gerência de redes.

Acredita-se que a parte de comunicação e distribuição das informações através destas plataformas seja muito interessante, tendo como perspectiva no futuro de testar comportamentos inteligentes com base nas técnicas conexionistas dentro destas plataformas. Ou até mesmo desenvolver, um conjunto de funções que possam ser adaptadas a estas plataformas para a utilização de técnicas conexionistas.

Anexo I – Script da rede de Elmann

Abaixo segue o código em linguagem script utilizado para testar a rede recorrente de Elmann.

```
% Rede Recorrente para testar problemas de redes de computadores
echo on
%  pause % Pressione qualquer tecla...
% Definiçao do Problema - Exemplos de utilizaç~ao da rede X tempo de 10 em
10s
%======================
p1 = [0.47 0.46 0.13 0.20 0.15 0.21 0.16 0.09 0.07 0.15 0.13 0.14 0.12 0.09
0.12 0.20 0.17 0.18  0.20 0.19 0.08 0.09 0.10  0.12   0.99 0.15 0.23 0.26
0.22 0.27 0.48 0.53 0.33 0.61 0.52  0.29 0.21 0.19  0.34 0.21 0.51 0.14];
p2 = 10:10:(length(p1)*10);
time=p2;
plot(p2, p1);
pause % pressione qualquer tecla para continuar...
% DEFINE A REDE DE ELMAN
% ========================
net = newelm([0 1; 1 (length(p1)*10)],[400 1],{'logsig','logsig'});
net.trainParam.epochs = 500;
net.trainParam.goal = 0.01;
pause %pressione qualquer tecla para continuar...
tic
[net,Y, E, tr] = adapt(net,[p1;p2]);
toc
pause
plot(E);
title('Taxa de Erro');
xlabel('Saidas dos Neuronios');
ylabel('Valor do Erro');
pause
plot(p2, p1, p2, Y);
title('Treinamento da Rede Recorrente');
xlabel('Saidas dos Neuronios ');
ylabel('Utilizaçao');
pause  %rede treinada
a = sim(net,[p1; p2]);
pause % pressione qualquer tecla para continuar...
plot(p2, a);
title('Teste para Redes');
xlabel('Saidas dos Neuronios');
ylabel('Valores de Utilizaçao');
pause % pressione qualquer tecla para continuar...
P1= [ 0.50  0.23  0.40  0.20   0.89  0.90  0.09 0.00 0.00 0.10 0.30 1.0 0.8
0.2 0.8 0.09 0.03];
P2= 10:10:(length(P1)*10);
a = sim(net,[P1; P2]);
pause % pressione qualquer tecla para continuar...
plot(P2, a);
title('Teste para Redes');
xlabel('Saidas dos Neuronios');
ylabel('Valores de Utilizaçao');
echo off
disp('End of rede1')
```

Anexo II – Script das RNAs recorrentes

Programa script do MatLab do desenvolvimento da aplicação de rede

```
%Leitura dos dados
clear;
leitura_arquivo5;
leitura_arquivo6;
leitura_arquivo7;
leitura_arquivo10;
leitura_arquivo12;
leitura_arquivo16;
leitura_arquivo17;
leitura_arquivo18;
leitura_arquivo19;
leitura_arquivo20;
leitura_arquivo21;
leitura_udesc;
% Aplicacao para a comprovacao da capacidade de aprender atraves de
exemplos.
% Os exemplos fornecidos foram obtidos da monitoraçao de um servidor de
% redes de computadores do
% CAV/UDESC.
% Aplicacao desenvolvida por Analucia Schiaffino Morales De Franceschi
% Criacao de uma rede neural recorrente utilizando as funcoes do MATLAB
% Definicao do numero de entradas e numero de camadas da rede neural
% recorrente
%   Neural Network object:
%   architecture:
%         numInputs: 0
%         numLayers: 0
%       biasConnect: []
%      inputConnect: []
%      layerConnect: []
%     outputConnect: []
%     targetConnect: []
%       numOutputs: 0  (read-only)
%        numTargets: 0  (read-only)
%    numInputDelays: 0  (read-only)
%    numLayerDelays: 0  (read-only)
%  subobject structures:
%           inputs: {0x1 cell} of inputs
%            layers: {0x1 cell} of layers
%           outputs: {1x0 cell} containing no outputs
%           targets: {1x0 cell} containing no targets
%            biases: {0x1 cell} containing no biases
%      inputWeights: {0x0 cell} containing no input weights
%      layerWeights: {0x0 cell} containing no layer weights
%    functions:
%          adaptFcn: (none)
%           initFcn: (none)
%        performFcn: (none)
%          trainFcn: (none)


%    parameters:
%        adaptParam: (none)
%         initParam: (none)
%      performParam: (none)
```

```
%        trainParam: (none)
%    weight and bias values:
%                IW: {0x0 cell} containing no input weight matrices
%                LW: {0x0 cell} containing no layer weight matrices
%                 b: {0x1 cell} containing no bias vectors
%    other:
%    userdata: (user stuff)%
% Cria um objeto rede
 net=network;

% Definicao dos parametros de entrada e numero de camadas
     net.numInputs=1;
     net.numLayers=2;
% Configuracao de  conexoes das entradas, das camadas, das saidas e dos
% bias
     net.biasConnect=[0; 0];
     net.inputConnect=[1; 0];
     net.layerConnect=[0 1; 1 1];
     net.outputConnect=[0 1];
     net.targetConnect=[0 1];
% Configura-se a quantidade de neuronios por entrada,
% neste caso tem-se duas camadas de entradas com 1 neuronio cada e o
% valor limite para cada entrada
      net.inputs{1}.range=[0 1];

% Numero de neuronios da primeira camada para a camada de saida, a funcao
% de transferencia e a inicializacao dos neuronios da primeira camada
      net.layers{1}.size=500;
      net.layers{1}.transferFcn='logsig';
      net.layers{1}.initFcn='initnw';

%Numero de neuronios da segunda camada -  a camada de saida
      net.layers{2}.transferFcn='logsig';
      net.layers{2}.initFcn='initnw';

% Definiçao de atrasos - Sao atribuidos atrasos para caracterizar o
% dinamismo da arquitetura neural
      net.layerWeights{1,2}.delays=1;
      net.layerWeights{2,2}.delays=1;

%Definicao de funçoes
      net.initFcn = 'initlay';
      net.performFcn='mse';
      net.trainFcn='trainlm';
      net.adaptFcn='traingdx';
%Inicializacao dos pesos
      net=init(net);

pause % Pressione qualquer tecla para visualizar o treinamento da rede...

     net.adaptParam.goal=1e-20; % Define o erro maximo admitido
     net.adaptParam.epochs=1000; %Define o numero de epocas
     net.adaptParam.min_grad=1e-20;
     net.performParam=10; %Parametro de performance

% Intervalos de tempo de 5 em 5 min - tempo de cada intervalo de coleta
dos dados
```

```
t=5:5:600;
    % Treinando .... para a Interface 2

    % Entradas para treinamento da Interface 2

taxa=[taxa_18_Int2];

tic
[net,y,e] = adapt(net, taxa);
toc

pause % Pressione qualquer tecla para visualizar a simulacao da rede

    d=sim(net,taxa);

    j=1;e=[];
    for i=1:3:120,
        e(j)=d(i);
        j=j+1;
    end

    j=1;b=[];
    for i=1:3:120,
        b(j)=y(i);
        j=j+1;
    end

    t=5:15:600;
    plot(t,e,'-ks','LineWidth',2,...
                 'MarkerEdgeColor','k',...
                 'MarkerFaceColor','k',...
                 'MarkerSize',5)

             hold on

             plot(t,b,'--kd','LineWidth',2,...
                 'MarkerEdgeColor','k',...
                 'MarkerFaceColor','w',...
                 'MarkerSize',7)

                ylabel('Taxa de Utilizaçao Interface 2');
                xlabel('Tempo');
                title('Simulaçao dos dados treinados');

             hold off

             h = legend('treinado','simulado',3);

pause %Pressione qualquer tecla...

%Simulaçao da rede da Interface 2

    t=5:15:600;
    % plot(t,y,'--', t,a, '-');
```

```
taxasim=[taxa_19_Int2];
a=sim(net,taxasim);

j=1;
for i=1:3:120,
    b(j)=a(i);
    x(j)=y(i);
    j=j+1;
end

taxasim=[taxa_20_Int2];

d=sim(net,taxasim);

j=1;e=[];
for i=1:3:120,
    e(j)=d(i);
    j=j+1;
end


subplot(2,1,1); plot(t,x,'-ks','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','k',...
            'MarkerSize',5)

        hold on

        plot(t,b,'--kd','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','w',...
            'MarkerSize',7)

   %       ylabel('Taxa de Utilizaçao Interface 2');
           ylabel('Taxa de Utilizaçao Interface 2');
           xlabel('Tempo');
           title('Dia 19/dez');

        hold off

        h = legend('treinado','simulado',3);

        subplot(2,1,2); plot(t,x,'-ks','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','k',...
            'MarkerSize',5)

        hold on

        plot(t,e,'--kd','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','w',...
            'MarkerSize',7)

           ylabel('Taxa de Utilizaçao Interface 2');
           xlabel('Tempo');
            title('Dia 20/dez');
```

```
            hold off

            h = legend('treinado','simulado',3);
```

%Programa da RNA – 3 camadas

```
       % Leitura dos dados
       clear;
       leitura_arquivo5;
       leitura_arquivo6;
       leitura_arquivo7;
       leitura_arquivo10;
       leitura_arquivo12;
       leitura_arquivo16;
       leitura_arquivo17;
       leitura_arquivo18;
       leitura_arquivo19;
       leitura_arquivo20;
       leitura_arquivo21;
       leitura_udesc;

% Aplicacao para a comprovacao da capacidade de aprender atraves de
% exemplos.
% Os exemplos fornecidos foram obtidos da monitoraçao de um servidor de
% redes de computadores do CAV/UDESC
% Aplicacao desenvolvida por Analucia Schiaffino Morales De Franceschi

% Criacao de uma rede neural recorrente utilizando as funcoes do MATLAB
% Definicao do numero de entradas e numero de camadas da rede neural
% recorrente


%   Neural Network object:
%
%    architecture:
%
%         numInputs: 0
%         numLayers: 0
%       biasConnect: []
%      inputConnect: []
%      layerConnect: []
%     outputConnect: []
%     targetConnect: []
%       numOutputs: 0  (read-only)
%        numTargets: 0  (read-only)
%    numInputDelays: 0  (read-only)
%    numLayerDelays: 0  (read-only)

%  subobject structures:

%           inputs: {0x1 cell} of inputs
%            layers: {0x1 cell} of layers
%           outputs: {1x0 cell} containing no outputs
```

```
%           targets: {1x0 cell} containing no targets
%            biases: {0x1 cell} containing no biases
%      inputWeights: {0x0 cell} containing no input weights
%      layerWeights: {0x0 cell} containing no layer weights

%    functions:

%          adaptFcn: (none)
%           initFcn: (none)
%        performFcn: (none)
%          trainFcn: (none)

%    parameters:

%        adaptParam: (none)
%         initParam: (none)
%      performParam: (none)
%        trainParam: (none)

%    weight and bias values:

% IW: {0x0 cell} containing no input weight matrices
% LW: {0x0 cell} containing no layer weight matrices
% b: {0x1 cell} containing no bias vectors

% other:

% userdata: (user stuff)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Cria um objeto rede
    net=network;

% Definicao dos parametros de entrada e numero de camadas
     net.numInputs=3;
     net.numLayers=4;

% Configuracao de  conexoes das entradas, das camadas, das saidas
% e dos bias
     net.biasConnect=[0; 0; 0; 0];
     net.inputConnect=[1 0 0; 0 1 0; 0 0 1; 0 0 0];
     net.layerConnect=[0 0 0 1; 0 0 0 1; 0 0 0 1; 1 1 1 1];
     net.outputConnect=[0 0 0 1];
     net.targetConnect=[0 0 0 1];
% Configura-se a quantidade de neuronios por entrada,
% neste caso tem-se tres camadas de entradas com 1 neuronio cada e o
% valor limite para cada entrada
     net.inputs{1}.range=[0 1];
     net.inputs{2}.range=[0 1];
     net.inputs{3}.range=[0 1];

% Numero de neuronios da primeira camada para a camada de saida, a
% funcao de transferencia e a
% inicializacao dos neuronios da primeira camada
       net.layers{1}.size=500;
       net.layers{1}.transferFcn='logsig';
       net.layers{1}.initFcn='initnw';
```

```
% Numero de neuronios da segunda camada para a camada de saida, a
% funcao de transferencia e a
% inicializacao dos neuronios da primeira camada
       net.layers{2}.size=500;
       net.layers{2}.transferFcn='logsig';
       net.layers{2}.initFcn='initnw';

% Numero de neuronios da segunda camada para a camada de saida, a funcao
% de transferencia e a inicializacao dos neuronios da primeira camada
       net.layers{3}.size=500;
       net.layers{3}.transferFcn='logsig';
       net.layers{3}.initFcn='initnw';

%Numero de neuronios da segunda camada -  a camada de saida
       net.layers{4}.transferFcn='logsig';
       net.layers{4}.initFcn='initnw';

% Definiçao de atrasos - Sao atribuidos atrasos para caracterizar o
% dinamismo da arquitetura neural
       net.layerWeights{1,4}.delays=1;
       net.layerWeights{2,4}.delays=1;
       net.layerWeights{3,4}.delays=1;
       net.layerWeights{4,4}.delays=1;

%Definicao de funçoes
       net.initFcn = 'initlay';
       net.performFcn='mse';
       net.trainFcn='trainlm';
       net.adaptFcn='traingdx';

%Inicializacao dos pesos
       net=init(net);

%pause % Pressione qualquer tecla para visualizar o treinamento da
% rede...

       net.adaptParam.goal=1e-20; % Define o erro maximo admitido
       net.adaptParam.epochs=1000; %Define o numero de epocas
       net.adaptParam.min_grad=1e-20;
       net.performParam=10; %Parametro de performance

% Intervalos de tempo de 5 em 5 min - tempo de cada intervalo de coleta
% dos dados

       t=5:5:600;


       % Treinando .... para a Interface 2

       % Entradas para treinamento da Interface 2

       taxa=[taxa_16_Int2; taxa_17_Int2; taxa_18_Int2];

       tic
       [net,y,e] = adapt(net, taxa);
       toc
```

```
pause % Pressione qualquer tecla para visualizar a simulacao da rede
       t=5:15:600;
       a=sim(net,taxa);
       j=1;b=[];x=[];
       for i=1:3:120,
           b(j)=a(i);
           x(j)=y(i);
           j=j+1;
        end



       plot(t,x,'-ks','LineWidth',2,...
       'MarkerEdgeColor','k',...
       'MarkerFaceColor','k',...
       'MarkerSize',5)

        hold on

        plot(t,b,'--kd','LineWidth',2,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','w',...
        'MarkerSize',7)

         ylabel('Taxa de Utilizaçao Interface 2');
         xlabel('Tempo');
         title('Simulaçao com valores de outra interface');
         hold off
         h = legend('treinado','simulado',4);

       pause


       %Simulaçao da rede da Interface 2

           t=5:15:600;
           taxasim=[taxa_19_Int2; taxa_20_Int2; taxa_21_Int2];
           a=sim(net,taxasim);

           j=1;b=[];x=[];
           for i=1:3:120,
               b(j)=a(i);
               x(j)=y(i);
               j=j+1;
           end

           taxasim=[taxa_05_Int2; taxa_05_Int2; taxa_05_Int2];
           d=sim(net,taxasim);

           j=1;e=[];
           for i=1:3:120,
               e(j)=d(i);
               j=j+1;
           end
```

```
subplot(2,1,1); plot(t,x,'-ks','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerFaceColor','k',...
'MarkerSize',5)

hold on

plot(t,b,'--kd','LineWidth',2,...
'MarkerEdgeColor','k',...
'MarkerFaceColor','w',...
'MarkerSize',7)

ylabel('Taxa de Utilizaçao Interface 2');
xlabel('Tempo');
title('Dias 19, 20 e 21/dez');


hold off

h = legend('treinado','simulado',4);


             subplot(2,1,2); plot(t,x,'-ks','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','k',...
            'MarkerSize',5)

        hold on

        plot(t,e,'--kd','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','w',...
            'MarkerSize',7)

           ylabel('Taxa de Utilizaçao Interface 2');
            xlabel('Tempo');
            title('Dia 05/dez');


        hold off

        h = legend('treinado','simulado',4);
```

% Programa da RNA – 5 camadas

```
% Leitura dos dados
clear;
leitura_arquivo5;
leitura_arquivo6;
leitura_arquivo7;
leitura_arquivo10;
leitura_arquivo12;
leitura_arquivo16;
leitura_arquivo17;
leitura_arquivo18;
```

```
leitura_arquivo19;
leitura_arquivo20;
leitura_arquivo21;
leitura_udesc;

% Aplicacao para a comprovacao da capacidade de aprender atraves de
exemplos.
% Os exemplos fornecidos foram obtidos da monitoraçao de um servidor de
redes de computadores do
%
% Aplicacao desenvolvida por Analucia Schiaffino Morales De Franceschi
%
% Criacao de uma rede neural recorrente utilizando as funcoes do MATLAB
% Definicao do numero de entradas e numero de camadas da rede neural
recorrente
%
%
%   Neural Network object:
%
%    architecture:
%
%        numInputs: 0
%        numLayers: 0
%      biasConnect: []
%     inputConnect: []
%     layerConnect: []
%    outputConnect: []
%    targetConnect: []
%      numOutputs: 0  (read-only)
%       numTargets: 0  (read-only)
%    numInputDelays: 0  (read-only)
%    numLayerDelays: 0  (read-only)

%  subobject structures:

%          inputs: {0x1 cell} of inputs
%           layers: {0x1 cell} of layers
%          outputs: {1x0 cell} containing no outputs
%          targets: {1x0 cell} containing no targets

%           biases: {0x1 cell} containing no biases
%     inputWeights: {0x0 cell} containing no input weights
%     layerWeights: {0x0 cell} containing no layer weights

%    functions:

%         adaptFcn: (none)
%          initFcn: (none)
%       performFcn: (none)
%         trainFcn: (none)

%    parameters:

%        adaptParam: (none)
%         initParam: (none)
%      performParam: (none)
%        trainParam: (none)
```

```
%    weight and bias values:

%                 IW: {0x0 cell} containing no input weight matrices
%                 LW: {0x0 cell} containing no layer weight matrices
%                  b: {0x1 cell} containing no bias vectors

%    other:

%          userdata: (user stuff)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
% Cria um objeto rede
    net=network;

% Definicao dos parametros de entrada e numero de camadas
     net.numInputs=5;
     net.numLayers=6;

% Configuracao de  conexoes das entradas, das camadas, das saidas e dos
bias
     net.biasConnect=[0; 0; 0; 0; 0; 0];
     net.inputConnect=[1 0 0 0 0; 0 1 0 0 0; 0 0 1 0 0; 0 0 0 1 0; 0 0 0
     0 1; 0 0 0 0 0];
     net.layerConnect=[0 0 0 0 0 1; 0 0 0 0 0 1; 0 0 0 0 0 1; 0 0 0 0 0
     1; 0 0 0 0 0 1; 1 1 1 1 1 1];
     net.outputConnect=[0 0 0 0 0 1];
     net.targetConnect=[0 0 0 0 0 1];

% Configura-se a quantidade de neuronios por entrada,
% neste caso tem-se cinco camadas de entradas com 1 neuronio cada e o
valor
% limite para cada entrada
      net.inputs{1}.range=[0 1];
      net.inputs{2}.range=[0 1];
      net.inputs{3}.range=[0 1];
      net.inputs{4}.range=[0 1];
      net.inputs{5}.range=[0 1];

% Numero de neuronios da primeira camada para a camada de saida, a funcao
de
% transferencia e a inicializacao dos neuronios da primeira camada
      net.layers{1}.size=500;
      net.layers{1}.transferFcn='logsig';
      net.layers{1}.initFcn='initnw';
% Numero de neuronios da segunda camada para a camada de saida, a funcao
de
% transferencia e a
% inicializacao dos neuronios da primeira camada
      net.layers{2}.size=500;
      net.layers{2}.transferFcn='logsig';
      net.layers{2}.initFcn='initnw';

% Numero de neuronios da segunda camada para a camada de saida, a funcao
de
% transferencia e a inicializacao dos neuronios da primeira camada
      net.layers{3}.size=500;
```

```
      net.layers{3}.transferFcn='logsig';
      net.layers{3}.initFcn='initnw';
% Numero de neuronios da segunda camada para a camada de saida, a funcao
de
% transferencia e ainicializacao dos neuronios da primeira camada
      net.layers{4}.size=500;
      net.layers{4}.transferFcn='logsig';
      net.layers{4}.initFcn='initnw';
% Numero de neuronios da segunda camada para a camada de saida, a funcao
de
% transferencia e a inicializacao dos neuronios da primeira camada
      net.layers{5}.size=500;
      net.layers{5}.transferFcn='logsig';
      net.layers{5}.initFcn='initnw';

%Numero de neuronios da segunda camada -  a camada de saida
      net.layers{6}.transferFcn='logsig';
      net.layers{6}.initFcn='initnw';

%Definiçao de atrasos - Sao atribuidos atrasos para caracterizar o
dinamismo da
% arquitetura neural

      net.layerWeights{1,6}.delays=1;
      net.layerWeights{2,6}.delays=1;
      net.layerWeights{3,6}.delays=1;
      net.layerWeights{4,6}.delays=1;
      net.layerWeights{5,6}.delays=1;
      net.layerWeights{6,6}.delays=1;


%Definicao de funçoes
      net.initFcn = 'initlay';
      net.performFcn='mse';
      net.trainFcn='trainlm';
      net.adaptFcn='traingdx';

%Inicializacao dos pesos
      net=init(net);

     net.adaptParam.goal=1e-20; % Define o erro maximo admitido
     net.adaptParam.epochs=1000; %Define o numero de epocas
     net.adaptParam.min_grad=1e-20;
     net.performParam=10; %Parametro de performance

% Intervalos de tempo de 5 em 5 min - tempo de cada intervalo de coleta
dos
% dados

t=5:5:600;

% Treinando .... para a Interface 2

% Entradas para treinamento da Interface 2

   taxa=[taxa_05_Int2; taxa_06_Int2; taxa_07_Int2; taxa_10_Int2;
taxa_12_Int2];
```

```
   tic
   [net,y,e] = adapt(net, taxa);
   toc

pause % Pressione qualquer tecla para visualizar a simulacao da rede

%Simulaçao da rede da Interface 2

    t=5:15:600;

    taxasim=[taxa_udesc_Int2; taxa_udesc_Int2; taxa_udesc_Int2;
    taxa_udesc_Int2; taxa_udesc_Int2];

    a=sim(net,taxasim);

    j=1;
    for i=1:3:120,
        b(j)=a(i);
        x(j)=y(i);
        j=j+1;
    end

      taxasim=[taxa_05_Int2; taxa_06_Int2; taxa_07_Int2; taxa_10_Int2;
      taxa_12_Int2];



    d=sim(net,taxasim);



    j=1;e=[];
    for i=1:3:120,
        e(j)=d(i);
        j=j+1;
    end

    subplot(2,1,1); plot(t,x,'-ks','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','k',...
                'MarkerSize',5)

            hold on

            plot(t,b,'--kd','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','w',...
                'MarkerSize',7)

       %       ylabel('Taxa de Utilizaçao Interface 2');
              xlabel('Tempo');
              title('Dia 05/dez');

            hold off

            h = legend('treinado','simulado',2);
```

```
        subplot(2,1,2); plot(t,x,'-ks','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','k',...
            'MarkerSize',5)

        hold on

        plot(t,e,'--kd','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','w',...
            'MarkerSize',7)

           ylabel('Taxa de Utilizaçao Interface 2');
           xlabel('Tempo');
            title('Dias 05, 06, 07, 10 e 12/dez');

        hold off

        h = legend('treinado','simulado',2);

erro=(y-a).^ 2;
erro=max(erro)
```

Anexo III – Exemplo de arquivo de *log*

Um exemplo de um arquivo de log obtido pela execução do MIB BROWSER do dia 18 de dezembro de 2002.

```
Data: Qua 18 Dez 2002 08:04:04
EntradaOctetos: 1180241333 12 3902032132 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3966907494 307 1102367667 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:09:05
EntradaOctetos: 1181104369 12 3902838862 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3967738938 307 1103188505 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:14:05
EntradaOctetos: 1181702618 12 3904309360 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3969212464 307 1103754040 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:19:06
EntradaOctetos: 1183158090 12 3908087884 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3973045821 307 1105155212 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:24:06
EntradaOctetos: 1185109196 12 3910721655 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3975740485 307 1107036492 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:29:07
EntradaOctetos: 1186792486 12 3916014923 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3981098177 307 1108620786 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:34:07
EntradaOctetos: 1187971976 12 3919144066 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3984272201 307 1109741860 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:39:08
EntradaOctetos: 1188978527 12 3920590087 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3985741566 307 1110711488 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:44:08
EntradaOctetos: 1189628132 12 3921813002 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3986976039 307 1111331398 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:49:09
EntradaOctetos: 1190726506 12 3923368527 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3988560333 307 1112377767 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:54:09
EntradaOctetos: 1192044060 12 3925677745 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3990910301 307 1113622051 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 08:59:10
EntradaOctetos: 1193658832 12 3932587261 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 3997907950 307 1115109578 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:04:10
EntradaOctetos: 1196347828 12 3942567543 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4008028840 307 1117638582 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:09:11
EntradaOctetos: 1198604759 12 3951940724 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4017513905 307 1119742369 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0
```

Data: Qua 18 Dez 2002 09:14:11
EntradaOctetos: 1200483581 12 3961722556 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4027395953 307 1121476173 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:19:12
EntradaOctetos: 1202401174 12 3965560571 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4031287253 307 1123323259 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:24:12
EntradaOctetos: 1206506320 12 3974519590 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4040440976 307 1127223330 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:29:13
EntradaOctetos: 1209573561 12 3982182030 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4048289820 307 1130112233 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:34:14
EntradaOctetos: 1213261975 12 3987988608 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4054183808 307 1133660885 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:39:14
EntradaOctetos: 1218571291 12 3995497366 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4061891176 307 1138715055 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:44:15
EntradaOctetos: 1220697631 12 4010031823 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4076584910 307 1140687396 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:49:15
EntradaOctetos: 1224141829 12 4018642299 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4085313325 307 1143981169 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:54:16
EntradaOctetos: 1229212268 12 4026428932 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4093206344 307 1148912710 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 09:59:16
EntradaOctetos: 1235266404 12 4031613765 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4098500758 307 1154818556 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:04:17
EntradaOctetos: 1246757044 12 4038047392 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4105098986 307 1166087569 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:09:18
EntradaOctetos: 1260456836 12 4048316282 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4115636183 307 1179461127 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:14:18
EntradaOctetos: 1266317667 12 4058827771 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4126295957 307 1185120310 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:19:19
EntradaOctetos: 1272111240 12 4068826301 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4136505905 307 1190707027 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:24:19
EntradaOctetos: 1274308684 12 4074103199 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4141837437 307 1192800070 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:29:20
EntradaOctetos: 1276658926 12 4081021765 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4148904037 307 1194995799 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:34:20
EntradaOctetos: 1278881281 12 4089176519 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4157217271 307 1197065379 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:39:21
EntradaOctetos: 1280637992 12 4103439132 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4171634349 307 1198645914 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:44:21
EntradaOctetos: 1283593731 12 4120328241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4188709092 307 1201412282 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:49:22
EntradaOctetos: 1285890401 12 4135576104 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4204136431 307 1203554135 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:54:22
EntradaOctetos: 1288127667 12 4154256925 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4223005558 307 1205600546 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 10:59:23
EntradaOctetos: 1290771288 12 4177664614 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4246650637 307 1207982614 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:04:23
EntradaOctetos: 1293801125 12 4198719549 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4267911989 307 1210772179 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:09:24
EntradaOctetos: 1296593897 12 4208705730 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4278048338 307 1213402525 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:14:24
EntradaOctetos: 1300201282 12 4215658874 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4285118050 307 1216860993 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:19:25
EntradaOctetos: 1302372400 12 4222601758 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 4292165527 307 1218913237 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:24:25
EntradaOctetos: 1305145612 12 4238850627 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 13648576 307 1221468317 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:29:26
EntradaOctetos: 1308854974 12 4262648971 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 37711163 307 1224900558 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:34:26
EntradaOctetos: 1313479786 12 4276341224 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 51593976 307 1229299077 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:39:27
EntradaOctetos: 1318164878 12 4286883397 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 62321683 307 1233743510 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:44:27
EntradaOctetos: 1322222643 12 1147583 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 71719801 307 1237592188 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:49:28
EntradaOctetos: 1326762952 12 12437041 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 83203208 307 1241896223 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:54:28
EntradaOctetos: 1331486510 12 24559868 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 95495334 307 1246389749 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 11:59:29
EntradaOctetos: 1333914212 12 32771207 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 103839675 307 1248659991 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:04:29
EntradaOctetos: 1337888980 12 45078570 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 116293265 307 1252442695 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:09:30
EntradaOctetos: 1342519123 12 53084611 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 124431640 307 1256902376 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:14:30
EntradaOctetos: 1345388499 12 60293762 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 131738396 307 1259659042 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:19:31
EntradaOctetos: 1348010576 12 71684669 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 143267926 307 1262134340 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:24:31
EntradaOctetos: 1350393926 12 78424790 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 150120955 307 1264386277 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:29:32
EntradaOctetos: 1353022312 12 82989526 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 154744782 307 1266907705 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:34:32
EntradaOctetos: 1355758017 12 88336859 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 160162910 307 1269543516 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:39:33
EntradaOctetos: 1358190337 12 92548625 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 164458349 307 1271878245 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:44:33
EntradaOctetos: 1360643078 12 97621464 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 169597881 307 1274230699 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:49:34
EntradaOctetos: 1362723771 12 102291351 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 174338520 307 1276220832 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:54:34
EntradaOctetos: 1365003507 12 105810985 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 177918936 307 1278416518 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 12:59:35

EntradaOctetos: 1366872504 12 109065620 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 181215452 307 1280210983 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:04:35
EntradaOctetos: 1368327406 12 111049855 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 183219264 307 1281615829 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:09:36
EntradaOctetos: 1371484174 12 113076626 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 185296126 307 1284693791 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:14:36
EntradaOctetos: 1373768033 12 115964546 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 188234776 307 1286905008 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:19:38
EntradaOctetos: 1380914539 12 119251242 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 191608155 307 1293936662 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:24:39
EntradaOctetos: 1391032216 12 124343152 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 196827854 307 1303876498 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:29:39
EntradaOctetos: 1399837778 12 128513691 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 201101577 307 1312554911 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:34:40
EntradaOctetos: 1409769724 12 133228237 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 205934835 307 1322346483 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:39:40
EntradaOctetos: 1421801956 12 142856732 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 215747783 307 1334185400 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:44:41
EntradaOctetos: 1430949749 12 150816633 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 223858361 307 1343140135 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:49:41
EntradaOctetos: 1452199552 12 159420201 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 232740526 307 1363932639 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 13:54:42
EntradaOctetos: 1464600502 12 169796408 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 243347237 307 1375933262 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0
Data: Qua 18 Dez 2002 13:59:42
EntradaOctetos: 1467811293 12 179221240 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 252937262 307 1378924025 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:04:43
EntradaOctetos: 1471742502 12 189983555 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 263880541 307 1382581705 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:09:43
EntradaOctetos: 1475642756 12 199088011 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 273144873 307 1386259733 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:14:44
EntradaOctetos: 1479815254 12 208036030 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SaídaOctetos: 282269346 307 1390211757 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:19:45
EntradaOctetos: 1483262673 12 216896131 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 291248162 307 1393483601 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:24:45
EntradaOctetos: 1485939886 12 224985052 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 299468092 307 1396018267 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:29:46
EntradaOctetos: 1488821633 12 237779792 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 312415702 307 1398722601 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:34:46
EntradaOctetos: 1492276115 12 249276529 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 324072367 307 1401988719 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:39:47
EntradaOctetos: 1497795458 12 262412057 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 337407367 307 1407275267 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:44:47
EntradaOctetos: 1501739072 12 273798288 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 348951386 307 1411025643 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:49:48
EntradaOctetos: 1506040953 12 286425659 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 361775285 307 1415123473 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:54:48
EntradaOctetos: 1508196566 12 293228416 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 368686701 307 1417139376 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 14:59:49
EntradaOctetos: 1510682323 12 305110743 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 380733302 307 1419463179 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:04:49
EntradaOctetos: 1512532398 12 316525500 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 392287610 307 1421161707 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:09:50
EntradaOctetos: 1515287929 12 325625779 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 401550323 307 1423753284 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:14:50
EntradaOctetos: 1519894659 12 342846099 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 419000092 307 1428122476 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:19:51
EntradaOctetos: 1525336024 12 361271277 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 437624725 307 1433345513 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:24:51
EntradaOctetos: 1530422260 12 380307536 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 456910321 307 1438220876 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:29:52
EntradaOctetos: 1533610248 12 395314923 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SaídaOctetos: 472066196 307 1441233095 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:34:52
EntradaOctetos: 1535425623 12 402550104 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 479397164 307 1442920235 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:39:53
EntradaOctetos: 1538131477 12 408774607 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 485731816 307 1445486624 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:44:53
EntradaOctetos: 1540245320 12 416663226 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 493731067 307 1447461179 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:49:54
EntradaOctetos: 1542665975 12 424100383 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 501278921 307 1449757122 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:54:54
EntradaOctetos: 1547541809 12 431764224 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 509067126 307 1454465446 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 15:59:55
EntradaOctetos: 1551176115 12 440743393 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 518206940 307 1457919061 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:04:55
EntradaOctetos: 1554793487 12 453155234 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 530785681 307 1461360309 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:09:56
EntradaOctetos: 1558777902 12 462958805 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 540738479 307 1465177103 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:14:56
EntradaOctetos: 1561679138 12 472422476 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 550385265 307 1467908319 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:19:57
EntradaOctetos: 1566087866 12 479439065 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 557505461 307 1472173632 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:24:57
EntradaOctetos: 1571948524 12 504481063 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 582804954 307 1477775745 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:29:58
EntradaOctetos: 1585417506 12 517715009 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 596258406 307 1490990803 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:34:59
EntradaOctetos: 1608153496 12 524635140 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 603339704 307 1513433713 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:39:59
EntradaOctetos: 1635233918 12 532834941 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 611739564 307 1540144924 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:45:00
EntradaOctetos: 1644112789 12 541954533 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SaídaOctetos: 621031260 307 1548764034 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:50:00
EntradaOctetos: 1647526691 12 555051016 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 634309733 307 1551964196 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 16:55:01
EntradaOctetos: 1650626999 12 573338737 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 652829967 307 1554827520 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:00:01
EntradaOctetos: 1655159218 12 590411932 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 670067566 307 1559134351 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:05:02
EntradaOctetos: 1659422793 12 615729010 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 695660105 307 1563132557 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:10:02
EntradaOctetos: 1663005159 12 644916864 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 725126673 307 1566411163 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:15:03
EntradaOctetos: 1665969590 12 661411006 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 741781321 307 1569169819 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:20:03
EntradaOctetos: 1668975215 12 679461087 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 760001246 307 1571948169 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:25:04
EntradaOctetos: 1672238923 12 698583159 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 779282442 307 1574979888 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:30:04

EntradaOctetos: 1675864936 12 717115536 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 797988022 307 1578358799 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:35:05
EntradaOctetos: 1677896521 12 736477585 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 817512759 307 1580169899 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:40:05
EntradaOctetos: 1680352525 12 757965480 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 839168137 307 1582398763 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:45:06
EntradaOctetos: 1683091400 12 777705890 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 859082292 307 1584876790 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:50:06
EntradaOctetos: 1685688155 12 795740463 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 877238644 307 1587280911 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 17:55:07
EntradaOctetos: 1689011048 12 822343939 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 904042477 307 1590308946 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 18:00:08
EntradaOctetos: 1692500917 12 847156094 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SaídaOctetos: 929044053 307 1593552056 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Data: Qua 18 Dez 2002 18:05:08
EntradaOctetos: 1695307789 12 866411236 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SaídaOctetos: 948487704 307 1596151259 0 0 0 0 0 0 0 0 72 72 72 72 72 48 48 0

Anexo IV – Script de leitura dos arquivos de log

Arquivo do programa Script para a leitura dos dados do log.

```
fid=fopen('dados16_12.log');
s2=fseek(fid,0,1);
s3=ftell(fid); %indica o fim do arquivo


position=fseek(fid,0,-1);%retorna para o inicio do arquivo
cont=ftell(fid);
DadosEntradaInt1=[];    DadosSaidaInt1=[];    DadosEntradaInt2=[];
DadosSaidaInt2=[];
B=[]; BB=[]; C=[]; CC=[];
% percorre o arquivo em busca das informaçoes de entrada
while cont<s3,
   A=fscanf(fid, '%s', 1);
   cont=ftell(fid);
   if (strcmp(A, 'EntradaOctetos:')==1)
       B=fscanf(fid, '%s', 1);
       B=str2num(B);
       DadosEntradaInt1=[DadosEntradaInt1 B];

       BB=fscanf(fid, '%s', 1);
       BB=fscanf(fid, '%s', 1);
       BB=str2num(BB);
       DadosEntradaInt2=[DadosEntradaInt2 BB];
   end
   if (strcmp(A, 'SaídaOctetos:')==1)
       C=fscanf(fid, '%s', 1);
       C=str2num(C);
       DadosSaidaInt1=[DadosSaidaInt1 C];

       CC=fscanf(fid, '%s', 1);
       CC=fscanf(fid, '%s', 1);
       CC=str2num(CC);
       DadosSaidaInt2=[DadosSaidaInt2 CC];

   end
end
fclose(fid);

%Calcular a taxa de utilizaçao Int1=10000000 bps  Int2=1544000 bps

n=length(DadosSaidaInt2);

for i=1:n-1,

    entrada1=DadosEntradaInt1(i+1)-DadosEntradaInt1(i);
    saida1=DadosSaidaInt1(i+1)-DadosSaidaInt1(i);

    if entrada1<0
        entrada1=DadosEntradaInt1(i+1);
    end

    if saida1<0
        saida1=DadosSaidaInt1(i+1);
    end
```

```
    % taxa de utilizaçao de um intervalo de tempo da Interface 1
    taxa1=(8*(entrada1+saida1))/(300*10000000);


    entrada2=DadosEntradaInt2(i+1)-DadosEntradaInt2(i);
    saida2=DadosSaidaInt2(i+1)-DadosSaidaInt2(i);

    if entrada2<0
        entrada2=DadosEntradaInt2(i+1);
    end

    if saida2<0
        saida2=DadosSaidaInt2(i+1);
    end

    % taxa de utilizaçao de um intervalo de tempo da Interface 2
    taxa2=(8*(entrada2+saida2))/(300*1544000);


   taxa_16_Int1(i)=taxa1;
   taxa_16_Int2(i)=taxa2;
end
```

Glossário

Agente (gerência de redes)

Existem várias definições para agentes em gerência de redes. Agente é um processo que troca informações de gerência entre a estação gerente e os objetos gerenciados. processos individuais que realizam tarefas e as comunicam a outros processos chamados gerentes

Agentes autônomos

Agentes autônomos de software são programas, dotados de inteligência que agem como um parceiro melhorando a eficiência do trabalho do ser humano. Pode operar em um ambiente de apenas um computador ou estar distribuído, nesse caso vários agentes interagindo em um ambiente distribuído formam uma IA Distribuída. Em hardware, são sistemas que possuem uma interação duradoura com um ambiente dinâmico externo, estando normalmente instalados fisicamente em mecanismos. São dotados de rodas ou outros meios de locomoção e sensores projetados para funcionarem por longos períodos de tempo, operando em um ciclo que envolve aquisição de informações do ambiente e a geração de dados de saída.

Algoritmo de aprendizado

O aprendizado em RNAs consiste no ajuste dos valores dos pesos das conexões. Quando esses valores são desconhecidos utiliza-se um algoritmo (ou regra) de aprendizado para aproximá-los. Entre as principais regras de aprendizado destacam-se: Aprendizado Hebbiano, Regra Delta, Retropropagação (ou "Backpropagation", em inglês), Aprendizado competitivo, Aprendizado por Reforço ou Aprendizado Aleatório.

Barramento

Meio físico que interconecta os equipamentos de uma rede de computadores. Ver Ethernet.

Bridge

Roteador que conecta duas ou mais redes e transmite os pacotes de uma para a outra. Normalmente operam a nível físico da rede, por exemplo, uma ponte que liga duas redes Ethernet e transmite de um cabo para outro pacotes que não são locais. Pontes diferem de repetidores porque armazenam e enviam pacotes completos enquanto repetidores transmitem sinais elétricos.

CMIP (Common Management Information Protocol)

Protocolo OSI para gerência de redes.

CMISE (Common Management Information Service)

Definição de serviços oferecidos pelo protocolo CMIP.

Conexão

Ligação lógica entre dois ou mais usuários de um serviço.

CSMA/CD (Carrier Sense Multiple Acess with Collision Detection)

É um dos protocolos de transmissão de redes locais mais utilizados, em que períodos de transmissão são seguidos de intervalos de contenção. Quando duas estações escutam o meio e "sentem" que está desocupado e tentam transmitir simultaneamente ambas detectam a colisão, param de transmitir e aguardam um período de contenção e tentam transmitir novamente.

DME (Distributed Management Environment)

É um conjunto de especificações para produtos de gerência distribuída, proposta pela OSF (Open Software Foundation). Fornece uma estrutura que habilita um sistema consistente e um esquema de gerência para ambiente distribuído com produto de diversos fabricantes.

Ethernet

Tecnologia de rede local inventada pela Xerox Corporation Palo Alto Research Center. Um Ethernet é um cabo coaxial passivo, em que as interconexões mantém os componentes ativos. É um sistema de entrega do melhor esforço (best-effort delivery) que utiliza CSMA/CD. A Xerox Corporation , A Digital Equipment Corporation e a Intel Corporation publicaram padrões para Ethernet a 10Mbps.

Gateway (Roteador)

Computador dedicado que liga dois ou mais redes e roteia pacotes de um para outro. Em particular, um gateway Internet roteia datagramas IP entre as redes que ele interconecta. Gateways também roteiam pacotes para outros gateways até que os pacotes sejam entregues no destinatário final diretamente através do meio físico. O termo normalmente é atribuído a máquinas que transferem informações de uma rede a outra , tal como "mail gateway".

Gerente (gerência de redes)

É o processo que troca informações de gerência com o agente. Nas plataformas de gerência normalmente é ativado pelo administrador da rede para tomadas de decisão.

Host (no ambiente de redes)

É um sistema terminal (ou final de linha).

IETF (*Internet Engineering Task Force*)

Força tarefa encarregada das resoluções normativas da Internet. São os encarregados de

elaborar e discutir os RFCs (*Request for Comments*).

IP (*Internet Protocol*)

Protocolo que oferece serviço não orientado a conexão correspondendo a camada de transporte do conjunto de protocolos da Internet.

ISO (*International Standards Organization*)

Organização internacional que RASCUNHA, discute, propõe e especifica padrões para protocolos de redes. A ISO é mais conhecida pelo modelo de referência das sete camadas que descreve uma organização conceitual de protocolos.

JAVA

Plataforma para desenvolvimento de aplicações para World Wide Web.

Java Beans

É um modelo de componentes desenvolvido em 1996 pela Sun Microsystems para a plataforma JAVA. Os componentes desevolvidos a partir deste modelo chamam-se *beans* e são compostos por classes escritas em JAVA. Um bean pode ser conectado a outros beans, notifica e trata a ocorrência de eventos, possibilita a sua isnpeção e configuração e permite o armazenamento do seu código e da sua configuração.

MIB (*Management Information Base*)

Uma coleção de objetos que podem ser acessados através do protocolo de gerência de redes.

Neurônio

No contexto biológico são células que compõem o sistema nervoso. É o sistema que comanda o funcionamento de todos nossos órgãos, aparelhos e sistemas. No contexto da Inteligência Artificial, são chamados de neurônios artificiais. É uma técnica de solução de problemas inspirada nos neurônios biológicos. Um neurônio apenas não pode ser utilizado para solucionar nada. Unidos entre vários formam as redes neurais (ou neuronais) artificiais.

NIST (*National Institute of Standards and Technology*)

Órgão do Departamento de Comércio dos EUA encarregado de padronizações. Antigo National Bureau of Standards.

OMNIPoints (*Open Management Interoperability Points*)

Definem um conjunto de padrões que podem ser usados por implementadores e fabricantes de redes. Suas versões são liberadas de dois em dois anos. O OMNIPoint 1 oferece o SNMP e o CMIP como duas opções de implementação. No sentido de permitir uma maior flexibilidade os fabricantes são encorajados pelas resoluções dos OMNIPoints.

OSI (*Open Systems Interconnection*)

Esforço internacional para facilitar as comunicações entre computadores de diferentes fabricantes.

Pesos sinápticos

Excitações (podem ser positivas ou negativas) de entrada dos neurônios artificiais.

Plataforma de gerência

Programas de gerência de redes que abrangem as cinco áreas funcionais do modelo de referência OSI.

Protocolo

Descrição formal de formatos de mensagens e regras que duas ou mais máquinas devem seguir para trocar suas mensagens. Os protocolos podem descrever tanto baixos níveis de detalhamento quanto a nível de aplicação.

Regras de Produção

Técnica declarativa e procedimental utilizada para representação do conhecimento em programas de Inteligência Artificial Simbólica. Normalmente, encontram-se na forma SE <condição ou condições> ENTÃO <realiza determinada ação>.

RFC (*Request for Comments*)

Série de documentos que descrevem o conjunto de protocolos da Internet e experimentos relacionados.

Serviço

Representa um conjunto de funções oferecidos ao usuário por um provedor. O serviço torna-se disponível através do SAP (Service Access Point). O provedor é visto como uma entidade abstrata e é representado pela interface do SAP. Os serviços podem ser orientado à conexão (possui três fases bem definidas, estabelecimento da conexão, transferência dos dados e liberação da conexão). Ou não orientado à conexão que possui uma única fase de endereçamento adicionada a transferência de dados.

SNMP (*Simple Network Management Protocol*)

Protocolo de aplicação que fornece serviço de gerência no conjunto de protocolos Internet.

SNMPv2 (*Simple Network Management Protocol version 2*)

Versão menos popular do SNMP. Deve-se ao fato de existir MIBs do SNMP que não são compatíveis com a nova versão. Além disso, é mais complicado e com custos mais elevados

para implantação. A vantagem é permitir de interações entre gerentes e novas operações de gerência, tal como, GetBulkRequest, que permite obter vários dados apenas com uma operação.

TCP (*Transmission Control Protocol*)

Protocolo de transporte que fornece o serviço orientado a conexão no conjunto de protocolos da Internet.

Threshold (ou limiar)

O threshold ou limiar é um importante conceito em gerência de redes. É uma forma de prevenir a rede de algum tipo de problema. Se este limiar for ultrapassado o gerente da rede é informado através de um alarme. Em alguns dispositivos de rede (tais como alguns modems) tem limiares de performance pré-determinados e alguns mecanismos de testes para assegurar que o desempenho do dispositivo seja satisfatório.

TMN (*Telecommunications Management Network*)

Protocolo de gerência para redes de telecomunicações. A proposta da TMN é fornecer um conjunto de interfaces padrões para facilitar o gerenciamento de funções de operação, administração e manutenção e de elementos de rede.

UDP (*User Datagram Protocol*)

Protocolo de transporte que fornece serviço não orientado à conexão no conjunto de protocolos da Internet.

Referências Bibliográficas

[1] ABOELELA, E.; DOULIGERIS, C. Fuzzy Temporal reasoning Model for Event Correlation in Network Management. In: IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS, 24th, 1999. **Proceedings...** USA: IEEE Press, 1999. p.150-160.

[2] AGOULMINE, N. Les Architectures Distribuees de Gestion de Reseaux et de Services. In: SEMINÁRIO FRANCO BRASILEIRO DE SISTEMAS INFORMÁTICOS DISTRIBUÍDOS – ARQUITETURAS MULTIMÍDIAS PARA AS TELECOMUNICAÇÕES, 2º. , 1997, Fortaleza. **Anais...** Brasil: 1997. p. 55-80

[3] ALBUQUERQUE, F. **TCP/IP Internet - Protocolos e Tecnologias.** 3. ed. Rio de Janeiro: Axcel Books, 2001.

[4] ALBUQUERQUE, F. **TCP/IP Internet – Programação de Sistemas Distribuídos – HTML, JAVASCRIPT e JAVA.** Rio de Janeiro: Axcel Books, 2001.

[5] ARBIB, M. A. **Brains, Machines and Mathematics**. USA: McGraw-Hill, 1964.

[6] BARRETO, J.M. **Conexionismo e a resolução de problemas**. Monografia. (Concurso Professor Titular) Departamento de Informática e Estatística. Universidade Federal de Santa Catarina, Florianópolis, 1996.

[7] BARRETO, J.M. **Inteligência Artificial No limiar do Século XXI Abordagem Híbrida Simbólica, Conexionista e Evolucionária.** 2. ed. Florianópolis: Editora ρρρ, 1999.

[8] BARRETO, J.M. **Inteligência Artificial No limiar do Século XXI Abordagem Híbrida Simbólica, Conexionista e Evolucionária.** 3. ed. Florianópolis: Editora ρρρ, 2001.

[9] BARRETO, J.M, M. Roisenberg, F.M. de Azevedo. Developing Artificial Neural Networks for Autonomous Agents Using Evolutionary Programming. In: IASTED INTERNATIONAL CONFERENCE ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING, 1998, Cancun. **Proceedings...** México: IASTED Press, 1998. P. 283-286.

[10] BAZZAN, A.L. Coordination Among Individually-Motivated Agents: An Evolutionary Approach. In: ADVANCES IN ARTIFICIAL INTELLIGENCE. 13TH BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 1996, Curitiba. **Proceedings...** Brazil: D.L. Borges, C.A.A. Kaestner (eds.), 1996.

[11] BELO, O., RIBEIRO, A. A Distributed Multi-Agent System Environment A Web Approach. In: IASTED INTERNATIONAL CONFERENCE, ARTIFICIAL INTELLIGENCE, EXPERT SYSTEMS AND NEURAL NETWORKS, 1996. **Proceedings**... IASTED Press, 1996.

[12] BIGUS, J.P; BIGUS, J. **Constructing Intelligent Agents Using Java.** 2. ed. USA: John Wiley and Sons, Inc., 2001.

[13] BRASIL, L. M. Algoritmos de Extração de Regras de Redes Neurais Artificiais. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 3º, 1997, Florianópolis. **Anais...** Brasil: Editora da UFSC, 1997.

[14] BROOKS, R. A. Artificial Life and Real Robots In*:* FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. p. 3-9.

[15] BURROWS, T. L.; NIRANJAN, M. The Use of Feed-forward and Recurrent Neural Networks for System Identification. Cambridge University Engineering Department. **Technical Report** (CUED/F-INFENG/TR158)**,** dec, 1993.

[16] CAMPIONE, M., WALRATH, K. **The JAVA™ Tutorial – Object-Oriented Programming for the Internet**. USA : Addison-Wesley, 1996.

[17] CARBONELL, J.G.; MICHALSKI, R.S.; MITCHELL, T. An Overview of Machine Learning. In: *Machine Learning: an artifical intelligence approach*, 1984. **Proceedings...** R.S. Michalski, J.C. Carbonell, T.M. Mitchell ed. Springer-Verlag, 1984. P.3-23.

[18] CARISSIMI, L.S, BASTOS, E.L., WESTPHALL, C.B. Definição de Gerentes e Agentes para Gerência de Redes. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE, 21º. **Anais**... Brasil: 1993. p.397-410.

[19] CHAN, P. **The Java ™ Developers Almanac 1999**. USA: Addison-Wesley, 1999.

[20] COLAJANNI, M., YU, P.S., DIAS, M. Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems, **IEEE Transactions on Parallel and Distributed Systems**, v. 9, n. 6, jun., 1998.

[21] COMER, D. E. **Internetworking with TCP/IP vol. I: Principles, Protocols and Architecture.** USA: Prentice-Hall, 1988.

[22] COMER, D.E., STEVENS, D.L. **Internetworking with TCP/IP vol. II: Design, Implementation and Internals.** USA: Prentice-Hall, 1991.

[23] COMER, D.E., STEVENS, D.L. **Internetworking with TCP/IP vol. III: Client-Server Programming and Applications.** USA: Prentice-Hall, 1993.

[24] DAO, B. V., DUATO, J. , YALAMANCHILI, S. Dynamically Configurable Message Flow Control for Fault-Tolerant Routing. **IEEE Transactions on Parallel and Distributed Systems**, v. 10, n.1, jan, 1999.

[25] DAUBER, S. M. Finding Fault. **BYTE**, USA, p. 207-214, mar., 1991.

[26] DAVIDGE, R. Looking at Life. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE, 1994. **Proceedings....** F.J. Varela, P. Bourgine (eds), A Bradford Book, The MIT Press, 1994. P. 448-455, 2nd ed.

[27] DE AZEVEDO, F. M. Uma Proposta de Modelos Formais de Neurônios e Redes Neurais Artificiais. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 3º, 1997,

Florianópolis. **Anais...** Brazil: Editora da UFSC, 1997. P.503-514

[28] DE AZEVEDO, F.M., BRASIL, L.M., OLIVEIRA, R.C.L. de. **Redes Neurais com Aplicações em Controle e Sistemas Especialistas**. Florianópolis : Visual Books Editora, 2000.

[29] DE FRANCESCHI, A. S. M. **Desenvolvimento de Agentes Autônomos para Gerência de Redes**. 2000. 144f. Monografia (Exame de Qualificação em Engenharia Elétrica. Sistemas de Informação), Universidade Federal de Santa Catarina, 2000.

[30] DE FRANCESCHI, A. S. M. **Uma Aplicação de Desempenho para Validar a Gerência Pró-ativa de Redes.** 1996. 120f. Dissertação (Mestrado em Sistemas de Computação), Universidade Federal de Santa Catarina, Florianópolis, 1996.

[31] DE FRANCESCHI, A.S.M., BARRETO, J.M. Autonomous Agents based on Recurrent Neural Networks Applied to Computer Network Management. In: INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS, ANALYSIS AND SYNTHESIS, 5th, 1999, Orlando. **Proceedings...** Florida: ISAS Ed., 1999.

[32] DE FRANCESCHI, A.S.M., BARRETO, J.M. Distributed Problem Solving Based on Recurrent Neural Networks Applied to Computer Network Management. In*:* INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 1999, Cheju. **Proceedings...** Korea: IEEE Press, 1999. Vol. III.

[33] DE FRANCESCHI, A.S.M., DA ROCHA, M. A., WEBER, H. L., WESTPHALL C. B. Proactive Network Management Using Remote Monitoring and Artificial Intelligence Techniques. In: INTERNATIONAL SYMPOSIUM ON COMPUTER COMMUNICATIONS, 1997, Alexandria. **Proceedings...** Egypt: IEEE Press, 1997.

[34] DE FRANCESCHI, A.S.M., DA ROCHA, M. A., WEBER, H. L., WESTPHALL C. B. Employing Remote Monitoring and Artificial Intelligence Techniques to Develop the Proactive Network Management. In: INTERNATIONAL WORKSHOP ON APPLICATION OF NEURAL NETWORKS IN TELECOMMUNICATIONS, 1997, Melbourne. **Procedings...** Australia: IEEE Press, 1997. p.116-123.

[35] DE FRANCESCHI, A.S.M., KORMANN, L. F., WESTPHALL, C.B. Performance Evaluation for Proactive Network Management. In: INTERNATIONAL ON COMMUNICATIONS CONFERENCE, 1996, Dallas. **Proceedings...** Texas: IEEE Press, 1996. Vol. I

[36] DE FRANCESCHI, A.S.M., KORMANN, L. F., WESTPHALL, C.B. A Performance Application for Proactive Network Management In: INTERNATIONAL WORKSHOP ON MANAGEMENT SYSTEMS, 2nd, 1996, Toronto. **Proceedings...** Canadá: IEEE Computer Society Press, 1996. P.15-20.

[37] DEMUTH, H.; BEALE, M. **Neural Network Toolbox For Use with MATLAB® - Users Guide Version 4**. USA : The Math Works, 2001. (ISBN 0-534-95049-3)

[38] EL-DARIEBY, M.; BIESZCZAD, A. Intelligent Mobile Agents: Towards Network Fault Management Automation. In: INTEGRATING MANAGEMENT APPLICATIONS. **Proceedings...** USA: IEEE Press, 1999.

[39] ETZIONI, O. Intelligence Without Robots (A Reply to Brooks). **AI Magazine**, dec., 1993.

[40] FALQUETO, J.; BARRETO, J.M., BORGES, P.S.S. Amplification of Perspectives in the Use of Evolutionary Computation. In: INTERNATIONAL SYMPOSIUM ON BIO-INFORMATICS AND BIOMEDICAL ENGINEERING -BIBE2000, 2000, Arlington. **Proceedings...** USA: IEEE Press, 2000. P.150-157

[41] FALQUETO, J., LIMA, W. C., BORGES, P. S. S., BARRETO, J. M. The Measurement of Artificial Intelligence: An IQ for Machines? In: IASTED INTERNATIONAL CONFERENCE ON MODELING, SIMULATION AND CONTROL, 2001, Innsbruck, Austria. **Proceedings...** Calgary, Zurich : Acta Press, 2001. Vol. I. P. 409-413. ISBN 0-88986-316-4; ISSN 1025-8973.

[42] FALQUETO, J., LIMA,W.C., BORGES,P.S.S., BARRETO, J.M. O desenvolvimento de uma métrica para sistemas de IA – Considerações. In: CONGRESO LATINOAMERICANO DE ENGENHARIA BIOMÉDICA, 2º, 2001, La Habana. **Anais...** Cuba, 2001.

[43] *FCAPS White Paper*. FutureSoft Transforming Networks. http://www.futsoft.com/pdf/fcapswp.pdf

[44] FINN, T. LABROU, Y., MAYFIELD, J. **KQML as an agent communication language**. Disponível em: finn@cs.umbc.edu em 13 de março de 1995.

[45] FRANKLIN, S. Autonomous Agents as Embodied AI. **Cybernetics and Systems**, USA, v.28, n.6, p. 499-520, 1997.

[46] GARDNER, R.A., GARDNER, B.T. Feed forward: the Ethological Basis of Animal Learning. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 399-401

[47] GOLDSCHLAGER, L.; LISTER, A. **Computer science: a modern introduction.** New Jersey: Prentice-Hall, 1982.

[48] GOLDSZMIDT, G., YEMINI, Y. Evaluation Management Decisions via Delegation. In: IFIP/ISINM'93 – INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 1993. **Proceedings...** USA: IEEE Press, 1993.

[49] HAYES, C. C. Agents in a Nutshell. **IEEE Transactions on Knowledge and Data Engineering**, v. 11, n.1, jan/feb, 1999.

[50] HOFFNER, Y. The Management of Monitoring in Object-based Distributed Systems. In: IFIP/ISINM'93 – INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 1993. **Proceedings**... USA: IEEE Press, 1993.

[51] HOFSTADTER, D. R. **Fluid Concepts and Creative Analogies**. Basic Books, 1995.

[52] HOFSTADTER, D. R. **Gödel, Escher, Bach: an Eternal Golden Braid.** Vintage Books, may, 1989.

[53] HOOD, C.; JI, C. Intelligent Agents for Proactive Fault Detection. **IEEE Internet Computing**. p. 65-72, mar/apr, 1998.

[54] HORN, J. Measuring the Evolving Complexity of Stimulus-Response Organisms. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 365-374

[55] HUANG, R., MA, J. , TSUBOI, E. Communication Network Design via a Genetic Algorithm based Learning Algorithm. University of Aizu. In: INTERNATIONAL CONFERENCE, ARTIFICIAL INTELLIGENCE, EXPERT SYSTEMS AND NEURAL NETWORKS, 1996. **Proceedings...** IASTED Press, 1996.

[56] HUARD, J.-F.; LAZAR, A.A. **Fault Isolation based on Decision-theoretic Troubleshooting**. Research Technical Report CU/CTR/TR/442-96-08. Columbia University, NY, 1996.

[57] JANDER, M. Proactive LAN Management. **Data Communications**, p.49-55, mar., 1993.

[58] **Java Dynamic Management Kit**. Reference Guide. Sun Microsystems. Disponível em: http://www.sun.com/software/java_dynamic . Acesso em: 10 de agosto de 2000..

[59] KAELBLING, L.P. An Adaptable Mobile Robot. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings**... A Bradford Book, The MIT Press, 1994. p. 41-47.

[60] KOCH, F. **Agentes Autônomos para Gerenciamento de Redes de Computadores**. 1997. 120f. Dissertação (Mestrado em Sistemas de Computação), Universidade Federal de Santa Catarina, Florianópolis, 1997.

[61] KURFESS, F.; SHAH, D. P. Monitoring Distributed Process with Intelligent Agents. In: IEEE CONFERENCE AND WORKSHOP ON ENGINEERING OF COMPUTER-BASED SYSTEMS, 1998. **Proceedings**... IEEE Press, 1998.

[62] LAWRENCE, J. Untangling Neural Nets, **Dr. Dobb's Journal**, USA, p.38-44, apr., 1990.

[63] LEINWAND, A., CONROY, K.F. **Network Management – A Practical Perspective**. 2. ed., USA: Addison-Wesley, 1996.

[64] LEJTER, M.; DEAN, T. A Framework for the Development of Multiagent Architectures. **IEEE Expert**, p.47-59, dec., 1996.

[65] LEMAY, L.; PERKINS, C.L. **Teach yourself JAVA in 21 days**. SamsNet Publishing, 1996.

[66] LESSER, V. R. Cooperative Multiagent Systems: A Personal View of the State-of-Art. **IEEE Transactions on Knowledge and Data Engineering**, v. 11, n.1, jan/feb, 1999.

[67] LÉVY, P. **As Tecnologias da Inteligência - o futuro do pensamento na era da informática**. Tradução Carlos Irineu da Costa. Rio de Janeiro: editora34, 1993.

[68] MADRUGA, E.L. **Ferramentas de Apoio à Gerência de Falhas e Desempenho em Contexto Distribuído**. 1994. Dissertação. (Mestrado em Ciência da Computação), Universidade Federal do Rio Grande do Sul, Porto Alegre, 1994.

[69] MAES, P. Learning Behavior Networks from Experience. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 48-57.

[70] MANDRICK, B. Selectionist Systems as Cognitive Systems. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 441-447.

[71] MANIONE, R., MONTANARI, F. Validation and Extension of Fault Management Applications through Environment Simulation. In: IFIP/ISINM'95 – INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 1995. **Proceedings**... IEEE Press, 1995.

[72] MAXION, R.A., FEATHER, F.E. A Case Study of Ethernet Anomalies in a Distributed Computing Environment, **IEEE Transactions on Reliability**, v.39, n.4, oct, 1990.

[73] MCCULLOCH, W.S., PITTS, W.H. A Logical Calculus of Ideas Immanent in Nervous Activity, **Bulletins of Mathematical Biophysics**, v.5, p.115-133, 1943.

[74] MINSKY, M.L., PAPERT, S.A. **Perceptrons: an introduction to computational geometry**. MIT Press, 1988.

[75] MOTOROLA CODEX, **The Basics Book of OSI and Network Management,** Motorola University Press, Addison-Wesley, 1993.

[76] MOURATA, M., TAKAGI. H. Two-Layer Modeling for Local Area Networks. **IEEE Transactions on Communications**, v. 36, n. 9, p. 1022-1034, sep., 1988.

[77] MOUSSALE, N.M., VICCARI, R.M., CORREA M. Intelligent tutoring systems modeled through the mental states. In: ADVANCES IN ARTIFICIAL INTELLIGENCE,BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 13º. ,SBIA´96, 1996, Curitiba. **Anais...** Paraná: SBC, 1996.

[78] MYERS, W. Artificial Neural Networks are Coming. **IEEE Expert**, p.3-6, apr, 1990.

[79] NOGUEIRA, J.M.S. O Estado Atual da Pesquisa e Desenvolvimento em Gerenciamento de Redes no Brasil. 2o Seminário Franco Brasileiro de Sistemas Informáticos Distribuídos. In: SEMINÁRIO FRANCO BRASILEIRO DE SISTEMAS INFORMÁTICOS DISTRIBUÍDOS – ARQUITETURAS MULTIMÍDIAS PARA AS TELECOMUNICAÇÕES, 2º. , 1997, Fortaleza. **Anais...** Brasil: 1997. P. 81-98.

[80] OATES, T. Fault **Identification in Computer Networks: A Review and a New Approach.** Computer Science Technical Report 95-113, Experimental

Knowledge Systems Laboratory, Computer Science Department, LGRC, University of Massachusetts, Amherst, MA, USA.

[81] PFEIFER, R., VERSCHURE, P. Distributed Adaptative Control: A Paradigm for Designing Autonomous Agents. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 21-29.

[82] POLYA, G. **A Arte de Resolver Problemas.** Interciência. Tradução de "How to solve it: A New Aspect of Mathematical Method", Rio de Janeiro: Princeton University Press, 1975.

[83] PRAS, A. **Network Management Architectures**. CTIT Ph. Thesis no. 95-02, Centre for Telematics and Information Technology, The Netherlands, 1995.

[84] REILLY, D.L., COOPER, L.N. An Overview of Neural Networks: Early Models to Real World Systems. **An Introduction to Neural and Electronic Networks**, Academic Press, p.227-247, 1990.

[85] *RFC2571 An Architecture for Describing SNMP Management Frameworks* - Network Working Group, April 1999. Disponível em: http://www.ietf.org/rfc. Acesso em: 27 de dezembro de 2002.

[86] RICH, E., KNIGHT, K. **Inteligência Artificial**. Tradução Maria Claudia S. R. Ratto. 2. ed. São Paulo: McGraw-Hill, 1993.

[87] ROISENBERG, M. **Emergência da Inteligência em Agentes Autônomos através de Modelos Inspirados na Natureza**. 1998. Tese (Doutorado em Sistemas de Informação), Universidade Federal de Santa Catarina, Florianópolis, 1998.

[88] ROISENBERG, M., BARRETO, J.M, DE AZEVEDO, F.M, BRASIL, L.M. On a Formal Concept of Autonomous Agents. In: IASTED INTERNATIONAL CONFERENCE APPLIED INFORMATICS, 16th, 1998, Germany. **Proceedings**... IASTED Press, 1998.

[89] ROISENBERG, M., BARRETO, J.M, DE AZEVEDO, F.M. Uma Proposta de Modelização para Agentes autônomos Baseada na Teoria de Sistemas. In: SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 3º, 1997, Vitória. **Anais...** P.500-507.

[90] ROISENBERG, M., BARRETO, J.M, DE AZEVEDO, F.M. A Neural Network that Implements Reactive Behavior Autonomous Agents. In: IASTED INTERNATIONAL CONFERENCE ARTIFICIAL INTELLIGENCE, EXPERT SYSTEMS AND NEURAL NETWORKS, 1996, Honolulu, Hawaii. **Proceedings**... P. 245-248.

[91] ROSE, M.T. **The Simple Book – An Introduction to Management of TCP/IP based Internets**. USA: Prentice-Hall, 1994.

[92] SCALABRIN, E.E., VANDENBERGHE, L., DE AZEVEDO, H., BARTHÈS, J-P.A. A generic model of cognitive agent to develop open systems. In: ADVANCES IN ARTIFICIAL INTELLIGENCE,BRAZILIAN SYMPOSIUM ON ARTIFICIAL INTELLIGENCE, 13º. ,SBIA´96, 1996, Curitiba. **Proceedings...**

[93] SCHWEITZER, C.M. **Desenvolvimento de Baselines com o Uso de Simulação para Automação da Gerência de Redes**. 1996. Monografia. (Conclusão de Curso, Ciência da Computação), Universidade Federal de Santa Catarina, Florianópolis, SC, 1996.

[94] SHORTLIFFE, E. H. **Computer based Medical Consultation: MYCIN**. New York: American-Elsevier, 1976.

[95] SMITHERS, T. Taking Eliminative Materialism Seriously: A Methodology for Autonomous Systems Research. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 31-39.

[96] SPINNEY, B. **Ethernet Tips and Techniques: for designing, Installing and Troubleshooting your Ethernet Network**, 2nd edition, CBM Books, 1995.

[97] STEVENSON, D. W. Network Management – What it is and What it isn't. *White Paper*. Disponível em: http://netman.cit.buffalo.edu/Doc/Dstevenson. Acesso em: 30 de março de 1999.

[98] TANEMBAUM, A.S. **Computer Networks.** USA: Prentice-Hall, 4. ed.

[99] VIEIRA, E.S., SARI, S.T. Prototipação de um Subagente Adaptativo Baseado em Redes Neurais. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, XIV, 1996, Fortaleza. **Anais...**

[100] VORUGANTI, R.R. A Global Network Management for the 90's. **IEEE Communications Magazine,** p. 74-83, aug., 1994.

[101] WEBB, B., SMITHERS, T. The Connection between AI And Biology in the Study of Behaviour. In: FIRST EUROPEAN CONFERENCE ON ARTIFICIAL LIFE. F.J. Varela, P. Bourgine (eds). **Proceedings...** A Bradford Book, The MIT Press, 1994. P. 421-427.

[102] WELD, D. ETZIONI, O. The First Law of Robotics. In: INTERNATIONAL CONFERENCE ON INTELLIGENCE, 12th, 1994. **Proceedings...**

[103] WIETGREFE, H.; TUCHS, K.-D.; JOBMAN, K; CARLS, G.; FRÖHLICH, P.; NEJDL, W.; STEINFELD, S. Using neural networks for Alarm Correlation in Cellular Phone Networks. In: INTERNATIONAL WORKSHOP ON APPLICATIONS OF NEURAL NETWORKS TO TELECOMMUNICATIONS 3, 1997, Melbourne, Australia. **Proceedings...** USA: Lawrence Erlbaum Associates Publishers, 1997. p.248-255.

[104] YOO, Y.-D. A Multi-agent Approach for a Multi-layered Student Model. In: IASTED INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, EXPERT SYSTEMS AND NEURAL NETWORKS, 1996. **Proceedings**... IASTED Press, 1996.

[105] ZHENG, Q., SHIN, K.G. Fault-Tolerant Real-Time Communication in Distributed Computing Systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 9, n.5, may, 1998.

[106] ZOMAYA, A.Y., CLEMENTS, M., OLARIU, S. A Framework for Reinforcement-Based Scheduling in Parallel Processor Systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 9, n. 3, mar., 1998.

[107] ZURADA, J.M. **Introduction to Artificial Neural Systems**, West Publishing Co., USA, 1992.