Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Departamento de Semicondutores, Instrumentos e Fotônica

Arquiteturas de Redes Neurais: uma Aplicação a Memórias Associativas

Marcelo Arturo Jara Pérez

Orientador: Prof. Dr. Furio Damiani

Tese apresentada à Faculdade de Engenharia Elétrica, da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do grau de *Mestre em Engenharia Elétrica*

Dezembro de 1991

Indice

AGRADECIMENTOS

Ao Prof. Dr. Furio Damiani pela valiosa orientação que me dispensou durante este trabalho.

Meus agradecimentos ao apoio de vários colegas da FEE/UNICAMP e do DSIF/ITE. Particularmente a Norian Marranghello pelas construtivas críticas e ajuda para descifrar alguns "*enigmas*" dos processos Occam.

À Srta. Edna Servidone, nas várias dúvidas de redação e correções do texto deste trabalho.

A Paulo Berardi, do Instituto de Automação do CTI, por facilitar a utilização do sistema Multi-Transputer.

Ao professor Clésio Luiz Tozzi, do DCA/UNICAMP, por facilitar o uso do sistema Mono-Transputer, no laboratório do DCA.

Agradeço, também, o apoio financeiro, na forma de bolsas de estudo, recebido do CAPES e do FAEP/UNICAMP, durante a realização deste trabalho.

Resumo

Apesar do crescimento da potência computacional das máquinas atuais, os computadores convencionais baseados no modelo seqüencial de Von Neumann, têm demonstrado serem muito ineficientes na resolução de problemas tais como o reconhecimento de padrões e outros de natureza similar. As Redes Neurais Artificiais (ANN) apresentam características como paralelismo maciço e alta conectividade, o que permite a abordagem mais eficiente de tais problemas. A partir da publicação dos modelos neurais de J.J.Hopfield, tem sido desenvolvido muito trabalho analítico na área, embora o modelamento matemático de ANNs ainda apresente muitas dificuldades, especialmente no estudo das características dinâmicas de tais sistemas. As simulações por computador têm sido extensivamente utilizadas como ferramenta no estudo e na compreensão de ANNs, validando os estudos analíticos e às vezes resultando no único método viável para analisar modelos de tratamento matemático muito complexo. A implementação por *software* de métodos numéricos e técnicas de tratamento de matrizes é, muitas vezes, melhor desempenhada quando se utiliza um sistema de multicomputadores e técnicas de processamento paralelo, criando ferramentas de simulação mais rápidas e eficientes. No caso da simulação de ANNs, o processamento paralelo se revela como um método que permite uma abordagem mais próxima da aplicação e natureza do problema (uma rede de neurônios ou elementos de processamento altamente interconectada), resultando na diminuição do tempo de resposta da simulação. A utilização da linguagem Occam acrescenta a capacidade de representar, de forma mais "natural", a operação concorrente do conjunto de células de uma ANN e do seu comportamento dinâmico. Neste trabalho, apresenta-se o resultado do estudo de alguns modelos de redes neurais associativas e uma abordagem para a sua simulação num ambiente paralelo e distribuído. Particularmente, trata-se da implementação do modelo binário de J.J. Hopfield, operando de forma síncrona, numa máquina de arquitetura MIMD de tipo *message-passing* baseada em processadores Transputers e configurada como um hipercubo de dimensão 3.

1 Introdução.

1.1. Aspectos Gerais

Os termos **Redes Neurais Artificiais** e **Modelos Conexionistas** [1],[2] vêm sendo freqüentemente utilizados para designar sistemas de computação complexos constituídos por um grande número de células ou elementos simples de processamento, cujo comportamento dinâmico é caracterizado pela interação coletiva maciça das células elementares [3]. Por tal, este comportamento tem sido considerado análogo ao do funcionamento do sistema neural biológico do cérebro. As Redes Neurais Artificiais (designadas no contexto deste trabalho por ANNs) são sistemas cuja operação é baseada, principalmente, em algoritmos especiais, originados e/ou derivados da área da neuro-ciência, e geralmente são inspirados nos modelos dos sistemas neurais biológicos, onde a operação coletiva é altamente concorrente, o que é fundamental para o funcionamento do cérebro. As características e propriedades apresentadas pelas ANNs são semelhantes às que o cérebro humano realiza eficientemente, tais como : paralelismo maciço, recuperação associativa de informação, aprendizado, tolerância às falhas, etc.

As ANNs podem resolver uma classe de problemas computacionais, que não podem ser eficientemente abordados por máquinas convencionais seqüenciais baseadas no modelo de Von Neumann, nem mesmo por sistemas de multi-computadores de alto desempenho utilizando processamento paralelo. Os problemas abordados eficientemente pelas ANNs são geralmente do tipo de reconhecimento de padrões, envolvendo inclusive processamento de imagem e da fala, recuperação de informação desde uma memória associativa, controle adaptativo de sistemas e processos, aprendizado para a compreensão de uma linguagem natural, memória associativa para máquinas dedicadas à Inteligência Artificial [4], [5], etc.

As simulações por computador têm sido extensivamente utilizadas como ferramenta no estudo e na compreensão de ANNs, validando os estudos analíticos e, às vezes, resultando no único método viável para analisar modelos cujo tratamento matemático é complexo. A fim de melhorar o desempenho das simulações, tem se utilizado, inclusive, arquiteturas maciçamente paralelas. O desenvolvimento das tecnologias VLSI (*Very Large Scale Integration*) também tem sido aproveitado em implementações em *hardware* dedicado [6]-[10], tanto digitais quanto analógicas, tendo estimulado o estudo e as aplicações práticas de modelos de memórias associativas [11]-[15].

Uma abordagem para o estudo de ANNs surge da utilização do processamento paralelo e de máquinas do tipo de **passagem de mensagens** [16]. Neste trabalho apresenta-se o resultado do estudo de modelos de ANNs quando utilizados como memórias associativas. Particularmente, trata-se do estudo e da simulação paralela do modelo discreto da Rede Neural de Hopfield [17], que possui intrinsecamente propriedades de memória associativa [18]-[20]. A simulação por computador foi feita utilizando a linguagem Occam-2, uma linguagem com facilidades para processamento paralelo [21],[22], num ambiente constituído por processadores **Transputer** T800 [23]. A rede neural foi simulada considerando um sistema objeto formado por uma rede de Transputers configurada numa topologia hipercúbica de dimensão 3 (8 módulos de processamento, *Transputers Modules* -TRAM), mais um módulo para efeitos de comunicação com o *Host* (computador tipo PC).

O roteamento das mensagens entre os nós do sistema de multiprocessadores, baseou-se em propostas algorítmicas de Q.F.Stout & B.Wagar [24]. Neste esquema, a arquitetura da máquina é definida como um sistema MIMD de passagem de mensagens [25].

A metodologia empregada na simulação paralela está associada ao conceito PDES (Parallel Discrete Event Simulation) [26], onde as maiores dificuldades na simulação paralela consistem em resolver os problemas de *deadlock*, *starvation*, de minimizar a sobrecarga (*overhead*) devido às comunicações intensivas entre os nós - requeridas pelo modelo neural virtual- e de **mapear** a rede neural no sistema de multiprocessadores. Tal mapeamento foi realizado com base na proposta de Gosh & Hwang [27], onde a ANN é decomposta hierarquicamente em regiões, considerando a sua conectividade e as suas propriedades funcionais. Já que um sistema conexionista constitui um modelo de processamento distribuído e paralelo, a vantagem da implementação de uma ANN num sistema de multicomputadores não dedicado se traduz numa maior generalidade. A implementação é facilitada quando é utilizada uma linguagem de programação concorrente com as características de Occam, onde um sistema que utilize processos a serem executados em paralelo pode ser programado tanto num ambiente de um só processador, utilizando um só Transputer, quanto uma rede de Transputers. O modelo da Rede de Hopfield foi simulado visando sua utilização como uma memória associativa ou memória endereçada pelo conteúdo (***CAM-Content Addressable Memory***).

1.2. Modelo de Processamento de Informação Neural.

Uma Rede Neural Artificial (ANN) é um sistema de computação que consiste de um conjunto de unidades elementares ou elementos de processamento -EPs- autônomas e altamente interconectadas. Cada conexão possui um valor numérico ou peso w_{ij}, que corresponde à influência do elemento u_i sobre o elemento u_j. Os pesos de sinal positivo indicam reforçamento e os pesos negativos representam inibição. Tais pesos determinam o comportamento dinâmico da rede, junto com o algoritmo de aprendizado, o qual, por sua vez, determina os valores dos pesos de conexão w_{ij}, a partir dos vetores ou exemplos de treinamento.

Cada um dos EPs computa uma saída numérica, ou ativação, a partir das ativações daqueles diretamente conectados a ele e dos pesos w_{ij} correspondentes a cada conexão, utilizando, geralmente, o mesmo algoritmo. Dependendo do modelo considerado, as entradas a cada EP podem ser discretas, tomando os valores {0,1} ou {-1,0,1}; como também contínuas, assumindo valores no intervalo [0,1] ou [-1,+1]. Numa rede de N neurônios, u_i representa tanto o elemento de processamento quanto a sua própria ativação numérica $u_i(t)$. Cada elemento u_i computa sua nova ativação $u_i(t+\tau)$ como uma função da soma ponderada das suas entradas e toma uma decisão comparando-a a um limiar predeterminado. Se a soma é maior que tal limiar, o neurônio passa a (ou permanece em) um estado **ativo** (+1); caso contrário, ele vai a um estado **inativo** (-1 ou 0).

Considerando-se todos os limiares iguais a zero, os EPs conformam uma rede completamente conectada. A evolução do estado da rede é governada pelo seguinte processo: se P_i representa o potencial de ativação de cada EP_i, o estado do i-ésimo neurônio pode ser descrito matematicamente por:

$$P_i = \sum_{j=0}^{N} w_{ij}\ u_j(t)$$

$$u_i(t+\tau) = f(P_i)$$

onde $f(P_i)$ corresponde à função de transferência do elemento neuronal i (Fig. 1.1).

e τ representa o tempo de resposta característico do neurônio

Na Fig. 1.1 representam-se possíveis Funções de Transferência ou características de Entrada/Saída para cada neurônio, correspondentes tanto ao modelo binário quanto ao contínuo.

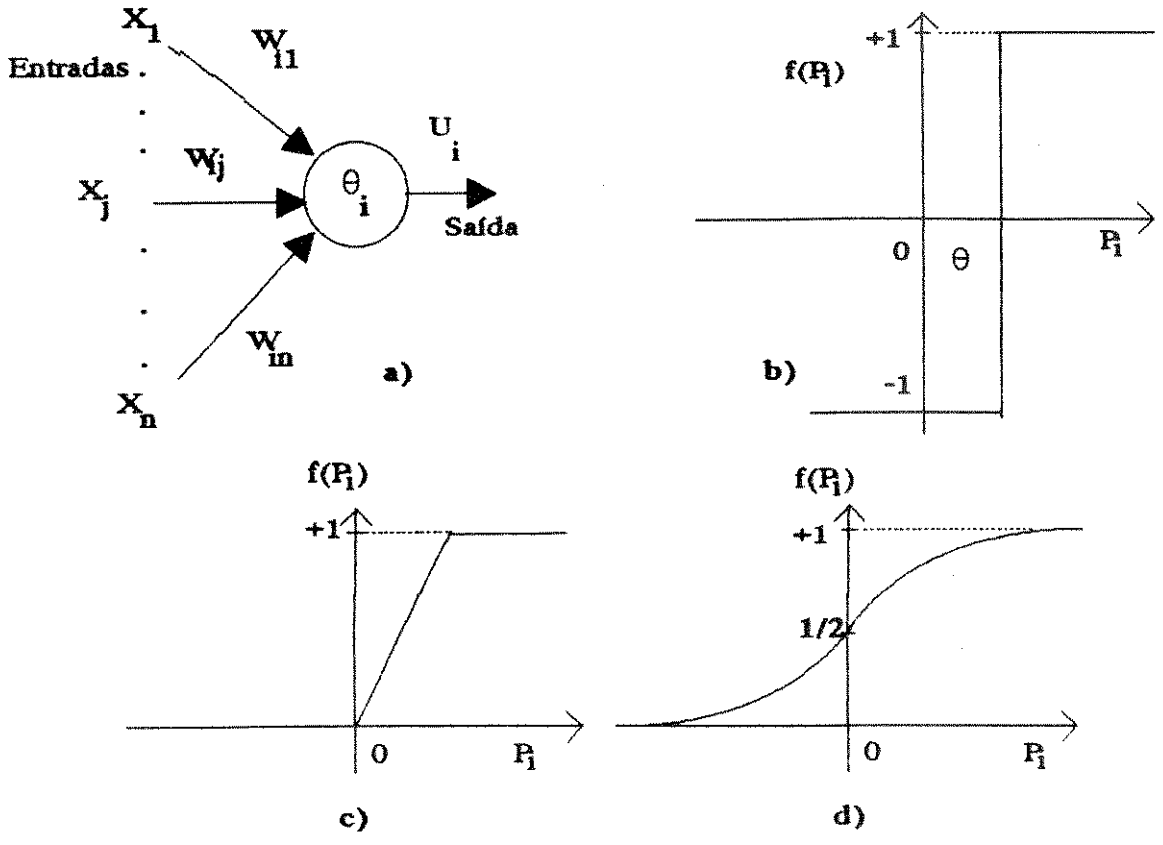

Fig. 1.1. Esquema e relações de entrada/saída de um elemento de processamento neuronal (EP).
a)modelo esquemático de um EP. **b)** função *threshold-logic* (modelo bipolar)
c) Modelamento Linear **d)** função não-linear tipo Sigmóide (modelo contínuo)
θ : Limiar do neurônio i

Se se considerar a rede neural com uma dinâmica paralela e síncrona, todos os neurônios avaliam os seus respectivos potenciais e tomam as suas decisões em forma simultânea, com o mesmo tempo de resposta τ. O estado de uma rede de N neurônios está representado por um vetor $\underline{u}$, cujos N componentes pertencem a {+1,-1 ou 0}. A dinâmica da rede é completamente definida pelos coeficientes de interação w_{ij}; a matriz de coeficientes **W** é geralmente conhecida como a **matriz sináptica do sistema**.

Alguns modelos de ANNs [28] podem ser, essencialmente, considerados como sistemas dinâmicos utilizadas na forma seguinte: a rede é inicializada em algum estado para logo evoluir até alcançar um estado estável ou um ciclo limite. Os estados estáveis da rede correspondem a pontos no espaço de estados aos quais a rede tende a evoluir no tempo, quando itera a partir de um estado inicial arbitrário. Tais pontos são chamados de **atratores**, ou pontos limites da rede, terminologia que é utilizada na descrição de sistemas dinâmicos não lineares. Tais atratores são caracterizados por um raio de atração, ou região de influência de certa forma e tamanho no espaço de estados. A inicialização da rede a partir de um estado na região de influência de um certo atrator,

leva-a a evoluir convergindo àquele atrator. Neste caso é alcançado um estado estável, podendo-se considerar que a rede efetuou uma computação, cujo resultado é expresso por aquele estado final. Isto é um mapeamento do tipo de muitos-a-um ou uma operação de busca associativa, característica das memórias associativas.

No grafo da Fig. 1.2 representa-se a estrutura de uma rede neural de seis elementos, completamente conectada. Seus nós representam os elementos de processamento (por ex. neurônios bi-estáveis com saída ±1), e seus arcos dirigidos representam as conexões entre neurônios e a direção do fluxo de informação entre dois elementos i e j, através de um peso w_{ij}.

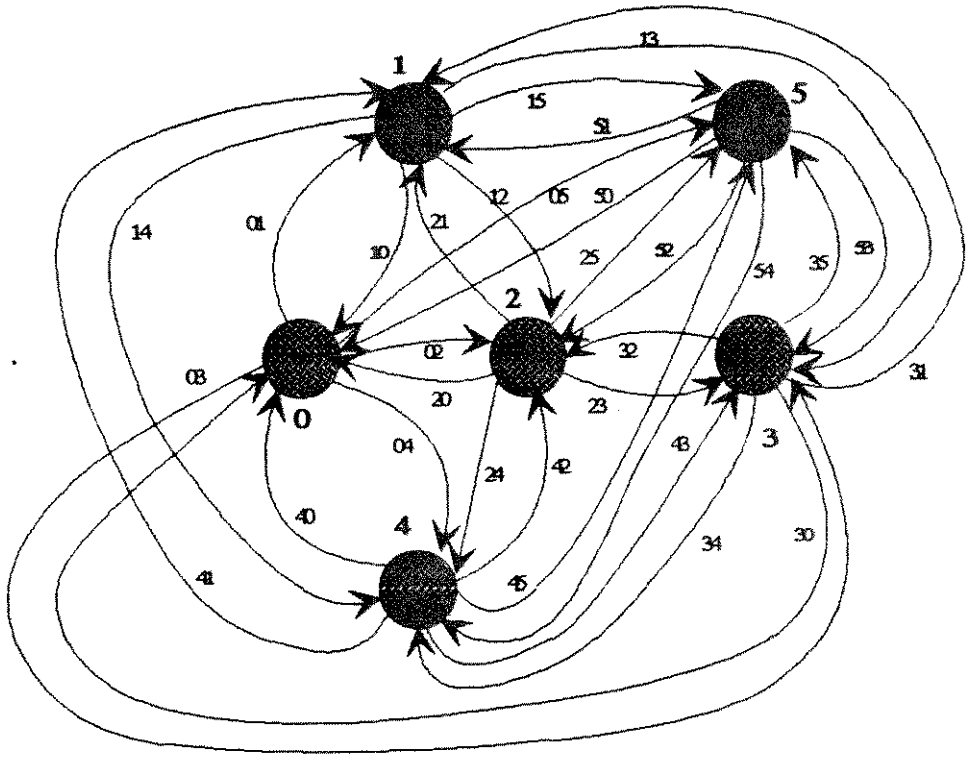

Fig. 1.2. Esquema de uma rede neural de 6 elementos de processamento, completamente conectada. O termo ij, no arco conectando dois nós, representa o peso da conexão entre os elementos i e j.

No capítulo seguinte, serão discutidos os princípios e conceitos básicos de memórias associativas. No Capítulo 3, será usado um formalismo matemático para descrever a dinâmica de certos modelos de memórias associativas. O funcionamento do modelo neural binário de Hopfield é tratado no Capítulo 4 e uma abordagem para a simulação deste modelo numa máquina de arquitetura paralela é proposta no Capítulo 5. Uma análise do desempenho dos algoritmos de comunicação descritos no Capítulo 5 é feita no Capítulo 6. Finalmente são discutidas as conclusões do trabalho.

2. Memórias Endereçadas pelo Conteúdo

2.1. Propósito e Natureza da Memória - Conceitos Fundamentais.

O conceito de memória pode ser entendido de várias formas diferentes. Usualmente, envolve um **mecanismo de armazenamento**, que utiliza um **meio de armazenamento**. Sua operação pode ser chamada de **função memória**, a qual coopera com outras funções do organismo, gerando fenômenos cuja complexidade varia desde simples memorizações até seqüências complexas de processamento.

Na ciência cognitiva, existem diferentes opiniões acerca da separabilidade da função memória das outras operações. Uma visão extrema tenta fazer uma distinção entre memória, pensamento, emoção, vontade, etc. Os processos mentais fazem uso do cérebro como um todo, portanto, não é possível separar a função memória localmente dos processadores, como o é no caso de um computador digital. Fisiologicamente, é mais preciso descrever a rede cerebral como um conjunto de **processadores adaptativos** (neurônios)[18], onde a memória está distribuída em todas as partes variáveis (sinapses) dos processadores mencionados.

2.2. Memória Distribuída.

De um ponto de vista **conexionista** [3], a informação não se encontra armazenada num lugar particular, ao invés, ela está distribuída numa rede de unidades elementares. A informação, então, é **evocada** em vez de **achada**. Mais que imaginar que as unidades neurais elementares codificam peças particulares de informação, a abordagem conexionista é de que a informação é armazenada na relação entre as unidades e que cada unidade participa na codificação de muitas memórias. O aspecto mais interessante de um sistema de memória distribuído envolve a operação caso os traços de memória armazenados não sejam independentes uns dos outros. Neste caso eles interagem, o fato de armazenar um dado na memória pode afetar os demais, e aqui é que se radica a grande força do sistema. A informação que é relativa à armazenada previamente, embora diferente, tende a **evocar** o padrão de atividade original, mesmo quando as entradas no sistema possam diferir em muitos detalhes.

As máquinas de Von Neumann estão baseadas na idéia de um só processador seqüencial central (UCP-Unidade Central de Processamento), operando sobre o conteúdo de uma **memória passiva**, na qual as estruturas de dados simplesmente esperam ser manipuladas. Na abordagem conexionista, a arquitetura básica é bastante diferente. Ao invés de uma única UCP e uma memória passiva, existe um grande

número de processadores elementares relativamente simples e altamente interconectados, os quais interagem uns com os outros em paralelo, através das próprias conexões de *hardware*. As mudanças no conteúdo da memória são feitas formando novas conexões ou mudando as intensidades delas. Isto supera o principal gargalo do modelo de Von Neumann, onde as estruturas de dados são processadas seqüencialmente via a UCP, de forma que é impossível movimentar uma grande quantidade de conhecimento em forma simultânea. A conseqüência da substituição da memória passiva por unidades interagindo simultaneamente é que o mecanismo de endereço é substituído por conexões específicas de *hardware*.

Os itens na memória distribuída correspondem a padrões de atividade distribuídos em muitas unidades de *hardware* elementares. A capacidade de ligar um ítem com outro é implementada modificando as intensidades das diferentes conexões do *hardware*, de tal forma que o padrão de atividade correspondente a um item possa ativar o padrão correspondente a um outro item.

A idéia de que um padrão de atividade representa um item na memória é explicado através do uso do termo *representação distribuída*, o qual é interpretado como uma forma de codificação de informação. A representação distribuída é um método de codificação especialmente adequado para uma máquina altamente paralela. Suponha-se que se deseje ter um sistema que possa reconhecer qualquer um dentre um número de itens. Este problema problema pode ser resolvido com uma única unidade interna que responda só quando o seu item particular ocorreu, ou com várias unidades internas, onde cada uma responde aos muitos itens de entrada possíveis. No segundo caso, tem-se que, apresentando um único item de entrada por vez, ele vai ser representado pelo padrão de atividade das unidades internas, embora nenhuma delas especifique individualmente o item de entrada. Desta forma, o padrão de atividade das unidades elementares se transforma na representação básica do item.

Estreitamente associado ao conceito de sistema de memória distribuída está o de *sistema de processamento distribuído*, onde são feitas computações complexas através da ação concorrente de uma grande quantidade de unidades elementares, cada uma processando seu próprio conjunto local de entradas.

2.3. O Conceito de Recuperação de informação baseada no Conteúdo.

As memórias baseadas no conteúdo têm proporcionado uma abordagem diferente ao *hardware*, especialmente nas operações de busca de informação. À diferença de uma memória de tipo **RAM** (*Random Access Memory*), uma **CAM** tem acesso aos dados diretamente pelo seu conteúdo, em vez de fazê-lo em forma indireta,

endereçando posições de memória e posteriormente fazendo uma comparação. Uma abordagem por *software*, amplamente utilizada, é conhecida como **codificação** ***Hashing*** [29],[30]. No entanto, o *hardware* de **CAMs** é relativamente desconhecido. De certa forma, pode-se visualizar uma **CAM** como uma memória que é uma representação da informação que ela contem, em vez de ser uma seqüência consecutiva de posições contendo dados não relacionados, tal como uma memória convencional baseada no endereçamento de posições de memória específicos.

Uma **CAM** tem sido comumente um sistema com *palavras de dados* e *rótulo fixo* que pode ser casado exatamente (*matching*) com o conteúdo da palavra. Desta forma podem se obter possíveis respostas múltiplas para só uma palavra-chave de busca.

O termo **memória associativa** descreve, segundo Chisvin [19], um sistema de recuperação e armazenamento de informação que é mais geral. Neste sistema pode-se recuperar e modificar células baseadas nos seus conteúdos, mas não se precisa de um casamento exato com a palavra-chave. Ao invés disso, podem-se utilizar diversas relações de magnitude tais como: maior, menor, intervalo entre certos limites, proximidade, etc. Em algumas aplicações como processamento de sinais, reconhecimento de padrões ou imagem, podem-se utilizar métodos de recuperação baseados no conteúdo de posições vizinhas, nas operações de busca e comparação.

Segundo Hanlon [31], uma definição para um sistema associativo seria como segue: "As memórias associativas geralmente têm sido descritas como uma coleção ou uma montagem de elementos que possuem capacidade de armazenamento de dados, e no qual o acesso é feito simultaneamente e em paralelo baseando-se no conteúdo dos dados em vez do endereçamento de posições específicas". O mesmo Hanlon menciona que o significado correto de "**associativo**", refere-se às inter-relações entre os dados, e não ao mecanismo de armazenamento, pelo que seria mais adequado utilizar o termo "**Memória Endereçada pelo Conteúdo**" (**CAM**- *Content Addressable Memory*) .

Tem sido enfatizado, em alguns estudos [4],[11]-[13,19,30,32], que a recuperação de informação por associação é, também, implementável por sistemas **físicos contínuos ou distribuídos e, portanto, as CAMs deviam ser entendidas só como um mecanismo possível para o armazenamento e recuperação de associações.**

Segundo Parhami [33], o paralelismo nas operações de busca não é essencial, pelo menos no que se refere à operação funcional. Ele propõe as seguintes definições:

CAM: Um dispositivo que armazena dados num número de células. As células podem ser acessíveis baseando-se no seu conteúdo.

Processador endereçado pelo Conteúdo: Uma CAM em que podem ser feitas transformações de dados mais sofisticadas, baseadas no conteúdo de um número de células escolhidas de acordo com os mesmos conteúdos; ou um computador ou um sistema de computadores que utilize tal tipo de memória como componente essencial para o armazenamento ou processamento dos dados.

Deve ser esclarecido que o acesso aos dados baseado no *conteúdo* significa que existe alguma comparação entre um *argumento de busca* externo e o conteúdo total ou parcial da informação armazenada em todas as células. Em princípio, é irrelevante se tais comparações são feitas através da utilização de *software*, sistemas mecânicos ou por circuitos eletrônicos paralelos. No entanto, uma CAM genuina faz todas as comparações em paralelo. Além disso, é preciso mencionar que a comparação por *matching de igualdade* entre o argumento de busca e a palavra na memória não é o único método utilizado. Se os dados armazenados possuem valores numéricos, o propósito da busca poderia ser a busca de células que satisfaçam certas condições de *relação de magnitude* com o argumento de busca (<, >, intervalo entre dois limites específicos, etc).

Uma outra possibilidade é fazer as comparações sem utilizar um argumento externo, como é o caso de encontrar um mínimo ou um máximo num conjunto de dados. Outro tipo de busca é baseado no conceito de *melhor matching* entre o argumento de busca e os dados da memória. Neste caso é preciso utilizar um método que permita quantificar a *similaridade* entre o argumento e os dados, como é por exemplo, a *Distância de Hamming* ou o *Cosseno de Direção* [18].

Na fig. 2.1 pode se visualizar uma memória conhecida como *Memória Catálogo*. Ela possui um diretório e uma memória de dados. Uma palavra-chave é inserida no diretório, a qual é utilizada como argumento de busca, comparando-a em paralelo com todas as palavras do diretório. A cada uma das posições da memória corresponde uma saída física, onde se obtem uma resposta no caso de um *matching*. Na figura 2.1 representa-se a obtenção de só uma resposta, a qual é utilizada como linha de endereço para uma memória de dados e que pode ser uma memória RAM normal. Neste caso o diretório assume o papel de decodificador de endereço.

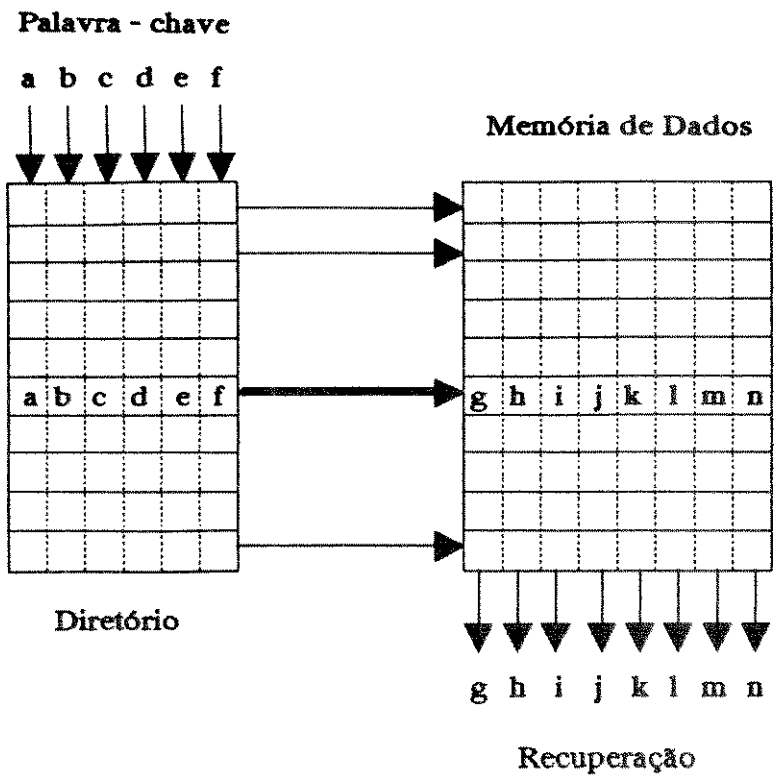

Fig. 2.1. Memória Catálogo.

Uma CAM de tipo mais geral pode ser visualizada no diagrama apresentado na fig. 2.2. Um registro particular pode ser achado casando-o com um padrão conhecido apresentado na entrada. O procedimento utiliza uma *palavra-chave*, uma *palavra-máscara* e uma *lógica de matching*. A palavra-chave constitui o padrão de comparação, enquanto a palavra máscara habilita só as partes da chave (bits) que são de interesse, ou relevantes à busca. Uma palavra, produto da combinação das palavras chave e máscara, é submetida a uma parte da memória chamada de lógica de *matching*, onde é realmente feita a comparação entre os dados. Depois de produzido o *matching* entre as palavras, os dados são colocados na saída da memória.

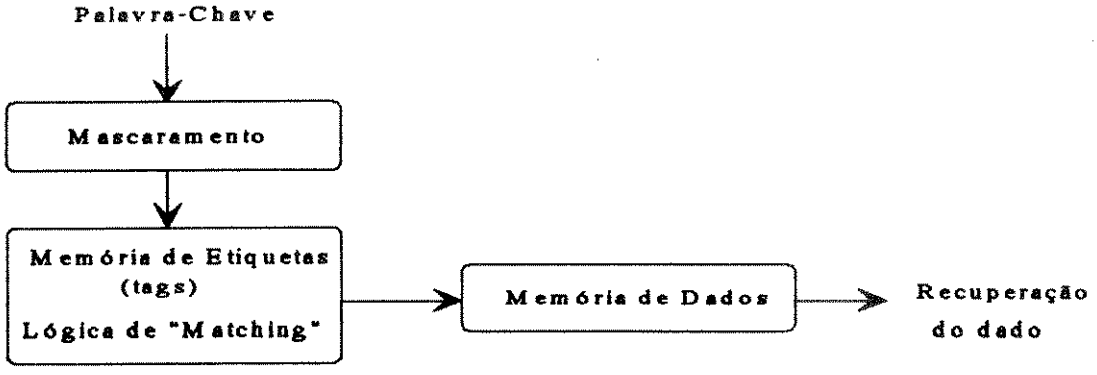

Fig. 2.2. - Diagrama de bloco de uma CAM

O arranjo de bits para uma CAM pode ser visto na fig. 2.3. Cada palavra de dados está dividida em três campos, segundo o esquema seguinte:

- *Bits rótulos*: indicam o tipo de locação (vazia ou ocupada). No caso da locação estar ocupada, o campo identifica o tipo de informação armazenada (ex. dados temporários ou código de programa).

- *Campos de etiqueta e dados*: o campo **etiqueta** é casado com a palavra chave ingressada. O campo **dado** contém a informação que deve ser recuperada ou modificada. No caso de se precisar mais flexibilidade ou se a informação **etiqueta** está incluída no dado, então os dois segmentos são tratados como um só campo. Em tal caso, o campo **dado** é comparado inteiramente com a palavra-chave de busca mascarada.

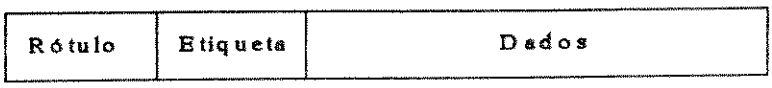

Fig. 2.3. Arranjo de bitz para uma CAM.

2.4. Princípios básicos do Endereçamento por Conteúdo.

Existem basicamente dois princípios para o endereçamento por conteúdo: um primeiro, baseado num mapeamento na memória, dependente do *hardware* e que é implementado através da utilização de *software* ou técnicas de programação, e um outro princípio utilizando construções especiais de *hardware*. Ambos os princípios existem desde a década dos anos 50 e somente neste último período é que se tem notado um avanço no *hardware* de sistemas associativos, tanto em termos de capacidade de armazenamento como em versatilidade, permitindo dar suporte a novas arquiteturas, como : máquinas de fluxo de dados, sistemas de computadores massivamente paralelos, máquinas dedicadas para Inteligência Artificial, etc.

2.4.1. Recuperação Associativa.

Nos meios que utilizam tecnologia de computadores, as associações possuem uma estrutura completamente rígida: uma ligação entre itens associados existe ou não. Esta forma de definição parece surgir de dois fatos :

- Nas ciências de informação e lógica se opera sobre variáveis e proposições de valores discretos.

- A segunda razão, que tem determinado o formato das associações, é prático. Não existe, até hoje, outra ferramenta tão poderosa quanto o computador digital, que permita implementar processos de informação artificiais. Isto tem influenciado

fortemente na seleção dos conceitos abstratos que se utilizam na descrição de processos de informação.

2.4.2. Leis de Associação Clássicas.

A forma convencional das leis de associação foram obtidas a partir de um conjunto de observações feitas sobre a memória humana, pelo filósofo grego Aristóteles. Os itens mentais, como : idéias, percepção, sensações ou sentimentos, estão ligados na memória segundo as seguintes condições:

1.- Se elas ocorrem simultaneamente ("contato espacial").

2.- Se elas ocorrem em sucessões próximas ("contato temporal").

3.- Se elas são similares.

4.- Se elas são contrárias.

É preciso distingüir as fases de operação indicadas como escrita (ou armazenamento) e leitura (ou recuperação). A simultaneidade ou sucessão próxima de sinais é necessária para produzir sua codificação num sistema físico. Num processo de recuperação, apresenta-se uma chave de entrada (argumento de busca), a qual é comparada com os vários itens da memória. O item procurado (ou parte dele) é obtido a partir da correlação destes com aquela. Tal correlação pode ser positiva (indicando similaridade) ou negativa (indicando contraste). Segundo Kohonen [30], isto parece relacionar as leis 1 e 2 com a escrita e as leis 3 e 4 com leitura, respectivamente. Um fator esquecido neste paradigma é o **contexto** no qual ocorrem as percepções primárias, o que é, em grande medida, responsável pela alta seletividade e capacidade da memória humana.

Segundo tal modelo, as características mais significativas da memória associativa humana são :

1) A informação é previamente procurada na memória com base numa medida de similaridade relativa ao padrão chave;

2) A memória tem a capacidade de armazenar representações de seqüências estruturadas;

3) A recuperação de informação desde a memória é um processo dinâmico, semelhante ao comportamento de muitos sistemas físicos contínuos no tempo.

Baseado no modelo anterior, Yoh-Han Pao [34] propôs que as características necessárias a uma memória associativa distribuída incluam o seguinte:

1) Armazenar muitos pares de padrões associados (estímulo, resposta);

2) Levar a cabo o armazenamento através de um processo de auto-organização;

3) Armazenar a informação em forma robusta, distribuída (o que implica na tolerância às falhas);

4) Gerar um padrão de resposta apropriado na saída, na recepção do padrão de estímulo associado;

5) Recuperar o padrão de resposta correto ainda quando o padrão estímulo de entrada esteja incompleto ou distorcido;

6) Acrescentar a informação à memória já existente.

2.4.3. Medidas de Similaridade.

A capacidade útil da memória para informação padronizada depende da habilidade para recuperar os itens desejados com suficiente **seletividade**. Se existe uma única representação para cada ocorrência, então a busca endereçada por conteúdo poderia estar baseada somente num **casamento exato** (***match***) entre o argumento de busca e os itens armazenados na memória.

No entanto, os padrões naturais e suas representações no domínio neuronal, estão sempre afetos a diversos tipos de ruído e erros. Tais problemas são particularmente tratados pela área de ***reconhecimento de padrões***. Por causa disso, um processo de recuperação de informação de tipo associativo deve considerar o conceito de similaridade.

Distância de Hamming. Consiste realmente em uma medida de assimilaridade entre representações digitais. Foi definida originalmente para códigos binários, embora seja facilmente aplicável à comparação de conjuntos de pares ordenados que possuem elementos com valores discretos.

Se se consideram dois conjuntos **x** e **y**, formados por elementos de valor binário, a comparação para a assimilaridade baseia-se no número de elementos diferentes entre eles, número este que é conhecido como ***distância de Hamming***.

Correlação. A comparação de sinais ou padrões de valor contínuo é freqüentemente baseada na correlação, que é outra medida de similaridade.

Supondo dois conjuntos ordenados ou seqüências de amostras de valores reais, $\mathbf{x} = (a_1, a_2, \ldots, a_n)$ e $\mathbf{y} = (b_1, b_2, \ldots, b_n)$. A correlação entre eles está definida por :

$$c = \sum_{i=1}^{n} a_i b_i \qquad (2.1)$$

Se **x** e **y** são considerados como vetores reais Euclidianos, c é o seu *produto escalar*.

Os métodos de correlação são mais bem utilizados na detecção de sinais periódicos contaminados por *ruído Gaussiano*. Já que as distribuições de padrões naturais nem sempre podem ser *Gaussianas*, devem ser considerados outros métodos como cossenos de direção [18]ou outros.

Similaridade variacional. As representações dos padrões na memória podem ser influenciadas por deformações ou transformações de escala (variação de magnitude, etc.). A comparação por similaridade é então levada a cabo considerando pequenas peças de padrões de cada vez. As peças são deslocadas em forma relativa umas às outras. Desta forma é achado o grau máximo de casamento.

O procedimento de casamento pode ser ilustrado por um exemplo de casamentos de *string*. Um *string* pode ser afetado por três tipos possíveis de erro :

1) de substituição : mudança de um símbolo em outro;
2) de inserção : ocorrência de um símbolo extra;
3) de apagamento : perda de um símbolo.

Os dois últimos tipos de erro contraem ou expandem o *string* pelo que o efeito é análogo a uma transformação de escala.

2.4.4. Mapeamento associativo.

A característica fundamental de uma recuperação associativa, a partir do ponto de vista de uma abordagem teórica de sistemas, é que não há posições de memória, pelo que não se precisa de um sistema de endereçamento. A *seletividade* da recuperação na saída, em relação aos padrões de entrada utilizados como argumento de busca, devem surgir das propriedades das funções de transformação utilizadas para representar o conteúdo da memória.

Uma transmissão simultânea (superposta) sobre uma linha única é possível, utilizando **funções ortogonais** [18] como portadoras dos sinais.

De forma similar, os traços de memória individuais num ambiente de memória distribuída podem ser de alguma forma ortogonalizados. Apesar da superposição sobre os mesmos elementos, a perda de informação pode ser desprezível.

Mapeamento Linear Ótimo. Considere o bloco do sistema indicado por **M** na fig. 2.4 na sua expressão básica : tal bloco pode ser considerado como uma rede de sinais. Um conjunto de sinais simultâneos pode ser considerado como um vetor real Euclideano de n componentes X_k , pertencente a R^n, onde k identifica cada conjunto. Um outro conjunto de sinais é obtido como saída, com algum atraso desprezível relativo a X_k e é denotado por Y_k, pertencente a R^m. Assume-se que a rede realiza uma transformação estática ou mapeamento de X_k em Y_k e, se no caso mais simples, considera-se um mapeamento linear, então deve-se considerar M como um operador de transferência, uma matriz pertencente a $R^{m \times n}$.

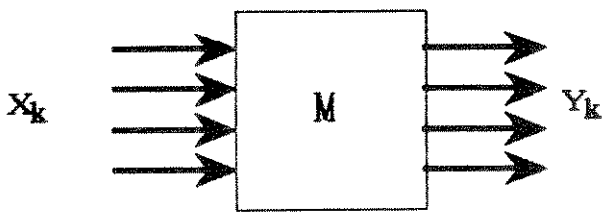

Fig. 2.4. Mapeamento Associativo.

Mapeamento Associativo não linear. A seletividade numa recuperação de informação associativa pode ser melhorada se se tomar uma transformação não linear em vez de uma linear. Isto é particularmente certo quando se tem um conjunto de padrões de entrada que são linearmente independentes. Existem, no entanto, muitas formas de transformações não lineares. Os padrões podem também ser armazenados em posições especiais separadas, para depois recuperá-las através de algum processo de casamento. Isto também é um mapeamento não linear e o desempenho na recuperação de informação vai ser fortemente dependente do critério de similaridade adotado (vide secção 2.3.2).

Também é importante mencionar que existem *estimadores* não lineares que formam parte específica dos dados, pelo que também se implementa desta forma um tipo de recuperação de informação associativa. Tais estimadores podem ser construídos, por exemplo, por técnicas de *abordagem estocástica*.

2.5. Operações elementares implementadas pela Memória Associativa.

2.5.1. Recuperação associativa, heteroassociativa e autoassociativa.

Existe a noção geral entre os pesquisadores dedicados ao cérebro, que as operações de processamento de informação podem ser expressas em termos de

funções de *filtros adaptativos* [18], [34]. A idéia é que o cérebro está organizado em um número de unidades funcionais, como por ex., grupos de células fortemente ligadas e interagindo entre elas, além de interações entre os grupos, conformando assim um sistema integrado. No entanto, a fim de analisar as operações fundamentais de processamento de informação, pode-se discutir uma peça isolada da rede neural, que poderia corresponder ao grupo mencionado. A rede pode ter entradas e saídas bem definidas (vide fig. 2.4), onde a operação básica do *filtro* é transformar um conjunto ordenado de entradas concomitantes (onde os eventos ou sinais de entrada ocorrem ao mesmo tempo), representadas por $\mathbf{x}=(x_1, x_2, x_3, \ldots, x_n)$ em outro conjunto ordenado de sinais de saída representadas por $\mathbf{y}=(y_1, y_2, y_3, \ldots, y_n)$. Como já foi mencionado, os atrasos dos sinais são desprezíveis. Se **x** e **y** são consideradas *variáveis vetoriais* multidimensionais, então **y** é uma função vetorial de **x** com alguns parâmetros adicionais. Tais parâmetros determinam as propriedades de transmissão variáveis da rede;

$$\mathbf{y} = \mathbf{y}(\mathbf{x},\mathbf{M}), \qquad (2.2)$$

onde **M** é o conjunto de parâmetros. Tal transformação se assume *adaptativa* no sentido de que os parâmetros são mudados pelo efeito dos sinais transmitidos. Os tipos destas mudanças são geralmente chamados de *efeitos memória*.

Neste caso, a recuperação associativa é considerada como um tipo particular de operação do filtro adaptativo. Pode-se distingüir entre dois tipos de operações de transformação: a **recuperação autoassociativa**, onde um padrão chave incompleto é transformado em uma versão completa (armazenada anteriormente), e a **recuperação heteroassociativa** que produz seletivamente um padrão de saída $\mathbf{x_k}$; neste último caso, os pares associados $\mathbf{x_k}$ e $\mathbf{y_k}$ podem ser escolhidos livremente, independemente um do outro. Esta é uma operação generalizada do processo simples de estímulo-resposta.

Se se tem um conjunto de vetores de entrada diferentes $\mathbf{X}=\{\mathbf{x}^{(1)},\mathbf{x}^{(2)},\ldots,\mathbf{x}^{(k)}\}$ e outro conjunto de vetores de saída $\mathbf{Y}=\{\mathbf{y}^{(1)},\mathbf{y}^{(2)},\ldots,\mathbf{y}^{(k)}\}$ então se diz que o sistema implementa uma recuperação heteroassociativa perfeita se :

$$\mathbf{x}(1) \longrightarrow \mathbf{y}(1)$$
$$\mathbf{x}(2) \longrightarrow \mathbf{y}(2)$$
$$\ldots$$
$$\mathbf{x}(k) \longrightarrow \mathbf{y}(k),$$

Da mesma forma, agora é possível definir uma recuperação autoassociativa perfeita. Neste caso, $x^{(p)}$ não é arbitrário, mas sim derivado de $y^{(p)}$, para ser recuperado. Por exemplo, se os vetores $x^{(p)}$ e $y^{(p)}$ tivessem a mesma dimensão, $x^{(p)}$ poderia ser obtido a partir de $y^{(p)}$ fazendo um sub-conjunto dos seus elementos igual a 0. Uma definição mais formal para uma memória autoassociativa ideal seria [7], um sistema que mantém cópias de conjuntos de sinais de entrada distintos $x^{(p)}$, com p=1,2, ... , k no seu estado interno, e produz uma cópia de um conjunto particular $x^{(r)}=(x_1{}^{(r)},x_2{}^{(r)}, \ldots ,x_n{}^{(r)})$, r pertencendo a {1,2,...,k}, nas suas saídas, quando quer que as entradas sejam excitadas por um conjunto de sinais $x=(x_1, x_2, \ldots, x_n)$ em que um subconjunto específico de valores x_i casa com o subconjunto correspondente de $x_i{}^{(r)}$.

2.5.2. Produção de seqüências desde uma memória associativa.

Seqüências também são implementadas por memórias associativas como as definidas na secção anterior. Para a abordagem de tal problema far-se-á uso do modelo de **máquina de estados finitos** (vide fig. 2.5).

Na teoria de autômatos, a máquina de estados finitos é uma máquina fundamental criada para a produção de seqüências de estados. Consiste de um sistema abstrato que possui um conjunto finito de estados internos S, aceita um elemento ou um conjunto finito de estados de entrada E, produzindo um elemento ou um conjunto finito de estados de saída O, conduzindo, conseqüentemente, a uma **transição de estado**. Na fig. 2.5, o bloco central representa a memória associativa, com três tipos de entrada: externas K e C, e de realimentação F. O sistema possui uma saída para recuperação de R. F é derivada a partir de R através de um atraso de tempo ; no caso mais simples se tem F(t) = R(t-1), onde t é um inteiro indicando a variável discreta de tempo. A memória possui dois modos de operação: **escrita** e **leitura**.

Os diferentes estados internos da máquina não são arbitrários, eles correspondem às réplicas dos pares de padrões de entrada recebidos (K,C). O padrão de realimentação F está associado com as entradas externas, formando uma tripla ordenada {(K(t),C(t),F(t))} , que é armazenada na memória durante o modo **escrita**.

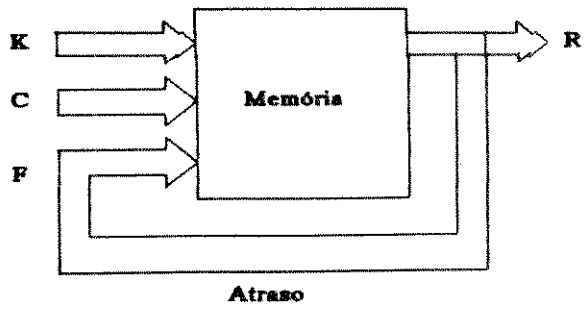

Fig. 2.5. Memoria Associativa para seqüências estruturadas.

Admite-se que a seqüência de entradas {K(t)} contem o padrão temporal central, enquanto C assume o papel de *contexto* ou fundo (*background*), o qual, geralmente, é estacionário durante uma seqüência particular. C permite distingüir entre seqüências diferentes, embora ocorra o mesmo padrão K(t) em eles.

Assumindo que é recebida a seguinte seqüência de entrada {(K(1),C),(K(2),C), ... ,(K(N),C)}, quando chega a primeira entrada (K(1),C), a entrada F não possui sinal. Neste caso, a entrada será representada por 0 (não interessa). Na busca associativa, o valor "0" é desprezível. Portanto, a seqüência de entrada efetiva será :

S={(K(1),C,0),(K(2),C,K(1)), ... , (K(N), C, K(N-1))}, que é o conjunto de triplas armazenadas.

A recuperação associativa de uma seqüência memorizada é feita no modo leitura, por ex., por meio da aplicação da entrada (K(1),C,0). As duas entradas externas possuem funções especializadas no processo seqüencial, embora a memória não faça diferença entre elas. Não são necessárias outras chaves K(k), k=2,3,... para a recuperação do resto da seqüência. Admite-se que a memória produz K(1) na sua porta de saída; tal padrão é transmitido à entrada F depois de um atraso de tempo unitário como F(2)=K(1). A leitura agora continua automaticamente, mas a entrada K(k), com k=2,3,..., atinge um valor "0" ou "vazio". A próxima entrada é (0,C,K(1)) que vai casar (*matching*) na sua parte específica com o segundo item da seqüência S. Portanto, se produz a saída K(2), e o processo continua recuperando o resto da seqüência K(3),K(4), ... ,K(N).

Tem-se mostrado assim que a memória associativa pode armazenar várias seqüências independentes, cada uma recuperável por sua própria chave de entrada, constituída por um padrão (K,C).

Deve-se considerar que um padrão de entrada pode *casar* com mais de uma tripla armazenada na memória. Em tal caso produz-se uma situação de casamento múltiplo,

onde não é muito claro qual é o estado sucessor. O modelo da fig. 2.5 é o mais simples, com *laços* de realimentação. Pode-se considerar um modelo com várias vias de realimentação, caso em que é possível recuperar seqüências mais complicadas.

2.6. Modelos Matriciais de Memória Associativa.

A idéia básica nestes tipos de modelos é a de um vetor de estado, ou seja, as representações ativas atuais dentro do sistema são codificadas como padrões de atividades presentes simultaneamente sobre o conjunto de todos os elementos contidos no sistema. Geralmente tais elementos são considerados como os neurônios, ou estreitamente relacionados aos neurônios e a sua atividade é interpretada como a freqüência de disparo ou algo estreitamente relacionado à freqüência de disparo.

Associador linear simples.

Supondo que se tem dois conjuntos de n neurônios, A e B, onde todo neurônio em A se projeta a cada neurônio em B. Um neurônio j em A está conectado ao neurônio i em B, através de uma sinapse com uma intensidade ou peso w_{ij}. O interesse primário é no comportamento do conjunto de atividades neuronais individuais simultâneas em um grupo de neurônios. Utiliza-se o termo padrão de *atividades individuais* porque, atualmente, o conhecimento a respeito da fisiologia do córtex cerebral indica que tais células são altamente individualistas.

Na teoria de processos de informação, um **padrão** é entendido como um conjunto ordenado de números reais, os quais representam valores de sinais espacial ou temporalmente adjacentes. Na teoria de reconhecimento de padrões, os conjuntos mencionados são representados por uma generalização dos vetores da geometria espacial (pertencente a R^n) e são chamados de **vetores de representação**. Um vetor de representação formal assume a forma $(x_1, x_2, \ldots, x_n)$, ou seja, um arranjo linear de números, isto é, um vetor $\mathbf{x}$ pertencente a R^n, de coordenadas $x_1, x_2, \ldots, x_n$.

Uma memória associativa distribuída foi descrita na fig. 2.4, onde $\mathbf{x_k}$ e $\mathbf{y_k}$ representam o padrão associado (estímulo, resposta) e **M** representa a memória.

Seja um par de padrões associados $(\mathbf{x_k}, \mathbf{y_k})$, onde $\mathbf{x_k}$ é um vetor no espaço n-dimensional e $\mathbf{y_k}$ é um vetor no espaço m-dimensional. Pode-se então formar um produto diádico do tipo:

$$\mathbf{M} = \mathbf{y_k} \cdot \mathbf{x_k}^T, \qquad (2.3)$$

onde a matriz **M** comporta-se como uma memória associativa, com as características mencionadas por Yoh-Han Pao no item 2.4.2.

Se $\underline{x}_k$ e $\underline{y}_k$ são vetores colunas, o armazenamento é formado por :

$$\mathbf{M} = \underline{y}_k \cdot \underline{x}_k^T$$

e a recuperação de informação é conseguida através da seguinte operação :

$$M\underline{x}_k = \underline{y}_k \cdot \underline{x}_k^T \cdot \underline{x}_k = \langle \underline{x}_k^T \cdot \underline{x}_k \rangle \underline{y}_k \quad \text{ou}$$

$M\underline{x}_k = \underline{y}_k$, se o valor do produto escalar $\langle \underline{x}_k^T \cdot \underline{x}_k \rangle$ é unitário.

Os símbolos <.> representam a natureza escalar do produto $\langle \underline{x}_k^T \cdot \underline{x}_k \rangle$, o que contrasta com a natureza diádica do produto $\underline{y}_k \cdot \underline{x}_k^T$.

$$\underline{X}_k = \begin{bmatrix} Xk1 \\ Xk2 \\ . \\ . \\ . \\ Xkn \end{bmatrix} \qquad \underline{Y}_k = \begin{bmatrix} Yk1 \\ Yk2 \\ . \\ . \\ . \\ Ykn \end{bmatrix} \qquad \mathbf{M} = \underline{y}_k \underline{x}_k^T \qquad \begin{bmatrix} Yk1\,Xk1 & Yk1\,Xk2 & \dots & Yk1\,Xkm \\ Yk2\,Xk1 & Yk2\,Xk2 & \dots & Yk2\,Xkm \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ Ykn\,Xk1 & Ykn\,Xk2 & \dots & Ykn\,Xkm \end{bmatrix}$$

$$\underline{X}_k^T = [\,Xk1\ Xk2 \dots Xkm\,]$$

Recuperação $= \mathbf{M}\,\underline{x}_k = \underline{y}_k \underline{x}_k^T \underline{x}_k$

$$= \begin{bmatrix} Yk1\,Xk1 & Yk1\,Xk2 & \dots & Yk1\,Xkm \\ Yk2\,Xk1 & Yk2\,Xk2 & \dots & Yk2\,Xkm \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ Ykn\,Xk1 & Ykn\,Xk2 & \dots & Ykn\,Xkm \end{bmatrix} \begin{bmatrix} Xk1 \\ Xk2 \\ . \\ . \\ . \\ Xkn \end{bmatrix}$$

$$= \begin{bmatrix} Yk1 \\ Yk2 \\ . \\ . \\ . \\ Ykn \end{bmatrix} (Xk1^2 + Xk2^2 + \dots + Xkm^2)$$

$$= \begin{bmatrix} Yk1 \\ Yk2 \\ . \\ . \\ . \\ Ykn \end{bmatrix} = \underline{Y}k \longrightarrow \textbf{Recuperação Perfeita}$$

2.7. Memórias Associativas geradas aleatoriamente.

Estes modelos têm sido tratados extensivamente por Shun-Ichi Amari em [35], [36] e [37]. Entre as versões mais conhecidas contam-se:

(1) memória associativa de correlação cruzada;

(2) memórias de correlação concatenadas em cascata;

(3) memórias associativas bilaterais;

(4) memória associativa de auto-correlação;

Considere-se a relação de entrada-saída de um elemento neural definida por:

$$y = \mathrm{sgn}(\Sigma\, w_i\, x_i - \theta) \qquad (2.4)$$

onde

$\mathbf{x}=(x_1,\ldots,x_n)$ representa o vetor de entradas (onde cada x_i assume valores ± 1)

w_i representa o peso de conexão entre elementos neurais.

sgn é a operação *sinal* que assume o valor +1 ou -1, dependendo do operando; e

θ é o limiar do elemento neurônio (para simplificar faz-se $\theta=0$)

Amari estudou modelos de redes onde os pesos de conexão w_i (ou forças sinápticas) são gerados aleatoriamente, sendo logo fixados. Desta forma, dada uma lei de probabilidade que determina os pesos, é construído um conjunto de redes geradas aleatoriamente. Amari utilizou um método chamado por ele de neurodinâmica estatística para indicar o comportamento de vários modelos de memórias associativas. Tal método será tratado mais detalhadamente no Cap. 3. A seguir, serão descritos, de forma geral, os principais modelos discutidos por Amari.

2.7.1. Memória associativa de correlação cruzada.

Este modelo consiste de uma rede de **n** elementos, onde é recebido um vetor sinal de entrada $\mathbf{x}=(x_1,\ldots,x_n)$ e é gerado um vetor sinal de saída $\mathbf{y}=(y_1,\ldots,y_n)$, onde y_i representa a saída do i-ésimo elemento e w_{ij} é a força sináptica do j-ésimo componente de um sinal **x** submetido ao i-ésimo neurônio. Numa memória associativa de correlação, a matriz de pesos W é definida por:

$$w_{ij} = 1/n \sum_{\mu=1}^{m} q_i^{\mu}\, s_j^{\mu} \qquad (2.5)$$

$$W = 1/n \sum_{\mu=1}^{m} q^{\mu}\, (s^{\mu})^T \qquad (2.6)$$

A expressão (2.5) está escrita na forma de componentes e a (2.6) é a notação na forma de vetor-matriz.

Se (2.4) é escrita como :

$$y = T\, x = \mathrm{sgn}\,(W\, x) \qquad (2.7)$$

onde T é um operador não-linear determinado pela matriz sináptica W, então, para um conjunto de pares de vetores (s^1,q^1), ... ,(s^m,q^m), numa memória associativa de correlação cruzada espera-se obter o vetor de saída q^μ, quando é submetido o vetor de entrada s^μ (μ=1,...,m). Isto significa que se deve satisfazer a relação: $T\, s^\mu = q^\mu$.

A relação anterior deve ser mantida para qualquer μ, o que implica recuperar q^μ, dado s^μ. Em (2.6) q^μ e s^μ representam vetores colunas, e $(s^\mu)^T$ é a transposta de s^μ.

Os componentes do vetor q^μ e s^μ são determinados aleatoriamente e de maneira independente, de tal forma que cada componente assume o valor de +1 ou -1 com uma probabilidade de 1/2. Isto significa que a matriz W é gerada de forma aleatória.

As propriedades e características deste tipo de rede são tratadas mais detalhadamente no capítulo 3.

2.7.2. Memórias associativas de correlação em cascata.

Nesta secção é descrito o modelo constituído por uma série de memórias associativas de correlação cruzada conectadas em cascata. Esta rede é formada pela concatenação de l redes $R_1,R_2,...,R_k,...R_l$, de tal forma que a saída x_l da (l-1)-ésima rede é a entrada da l-ésima rede R_l (vide Fig. 2.6).

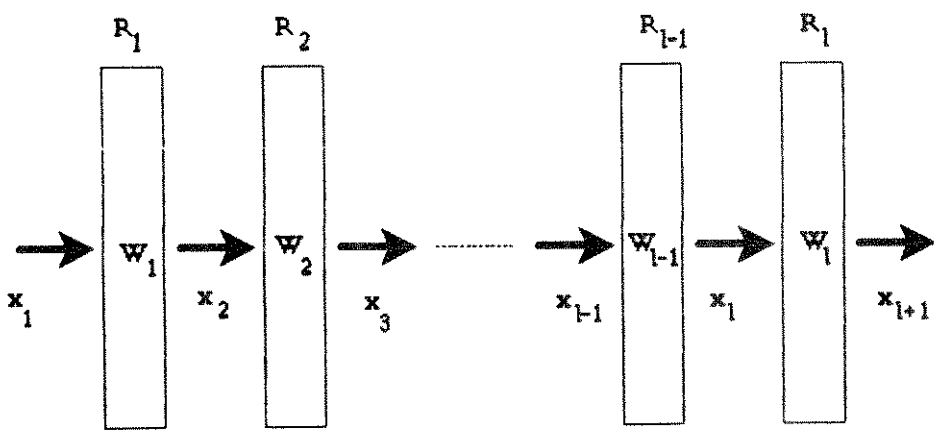

Fig.2.6. Esquema de uma série de memórias associativas concatenadas em cascata.

Se uma rede R_l recebe uma entrada x_l e emite uma saída x_{l+1}, então tem-se:

$$x_{l+1} = T_l\, x_l, \qquad (2.8)$$

onde T_l é uma transformação não-linear. A equação (2.8) pode ser expressa por:

$$T_l\, x = \mathrm{sgn}\,(W_l\, x) \qquad (2.9)$$

Sejam m seqüências compostas por vetores descritas por $S^1=\{s^1_1,...,s^1_{k+1},...\}$, $S^2=\{s^2_1,...,s^2_{k+1},...\}$, $S^m=\{s^m_1,...,s^m_{k+1},...\}$. Se s^μ_1 é submetida à primeira rede R_1 e é mantida a expressão $s^\mu_{l+1}=T_l s^\mu_l$ para μ=1,...,m, o sistema recupera a sequência S^μ através da saída $\{s\mu_{l+1}\}$ emitida pela rede R_l. Se a entrada x_1 à rede R_1 é próxima de

$s^{\mu}{}_1$, espera-se que $x_{l+1}=T_l x_l$ se aproxime cada vez mais de $s^{\mu}{}_{l+1}$, o que é considerado como uma propriedade de redução de ruído. No seu estudo, Amari supõe que a matriz sináptica W_l é determinada por:

$$W_l = 1/n \sum_{\mu=1}^{m} s^{\mu}{}_{l+1} (s^{\mu}{}_l)^T \qquad (2.10)$$

onde cada um dos componentes de $s^{\mu}{}_l$ é determinado de forma aleatória e independente.

2.7.3. Memória Associativa Cíclica

Esta memória é obtida acrescentando-se uma conexão que realimenta a memória associativa em cascata desde a saída da última memória até a entrada da primeira (vide Fig.2.7).

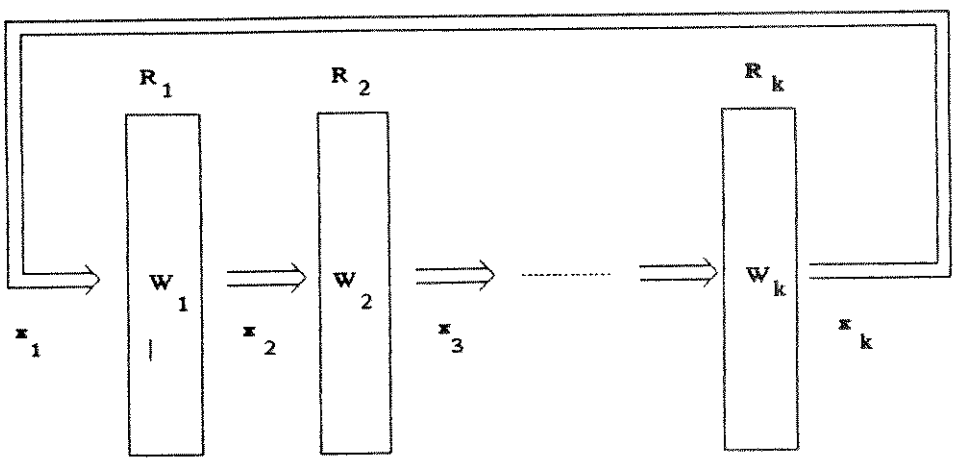

Fig.2.7. Esquema de uma série de memórias associativas concatenadas em cascata.

Esta memória é conhecida na literatura como k-AM [36], quando composta por k redes conectadas na forma de um anel.

Se um sinal é transformado através da passagem pelos componentes R_l da rede, seqüencialmente, um por vez, então tem-se a expressão:

$$x^{(t)}{}_{l+1} = T_l\, x^{(t)}{}_l \qquad (2.11)$$

onde t é o número de vezes que é reciclado o circuito da rede e l é calculado de forma tal que $x^{(t)}{}_{k+1}$ seja igual a $x^{(t+1)}{}_1$.

Sejam m seqüências de vetores $s^{\mu}=\{s^{\mu}{}_1, s^{\mu}{}_2, \ldots, s^{\mu}{}_k\}$; $\mu=1,\ldots,m$. Espera-se que se mantenha a relação $s^{\mu}{}_{l+1}=T_l\, s^{\mu}{}_1$, para todo $\mu=1,\ldots,m$ e $l=1,\ldots,k,(k+1=1)$. A matriz de conexões é dada da mesma forma que em (2.10). Nesta rede deseja-se uma propriedade de redução de ruído de forma que, dada uma entrada x_1, o sinal x_l converge a $s^1{}_l$ quando é submetido um x_1 próximo a $s^1{}_1$.

2.7.4. Memória Autoassociativa.

Se é considerada uma rede tal que a saída é realimentada na sua entrada (Fig.2.8), o modelo pode ser considerado como uma 1-AM. Dada uma matriz de conexões W, o comportamento dinâmico da rede é expresso por:

$$\mathbf{x}^{t+1} = T\mathbf{x}^t = \mathrm{sgn}(W\mathbf{x}^t) \qquad (2.12)$$

onde $\mathbf{x}^t$ representa o vetor de estados da rede no tempo t.

Dado m padrões $\mathbf{s}^1,\ldots,\mathbf{s}^m$, tem-se uma memória de tipo auto-associativa quando a matriz sináptica é dada por:

$$W = 1/n \sum_{\mu=1}^{m} \mathbf{s}^\mu (\mathbf{s}^\mu)^T \qquad (2.13)$$

Neste modelo, espera-se que todos os vetores $\mathbf{s}^\mu$ representem pontos de equilíbrio do comportamento dinâmico da rede, de tal forma que se satisfaça a relação $\mathbf{s}^\mu = T\mathbf{s}^\mu$, para $\mu=1,\ldots,m$. O comportamento dinâmico deste modelo é tratado no capítulo3.

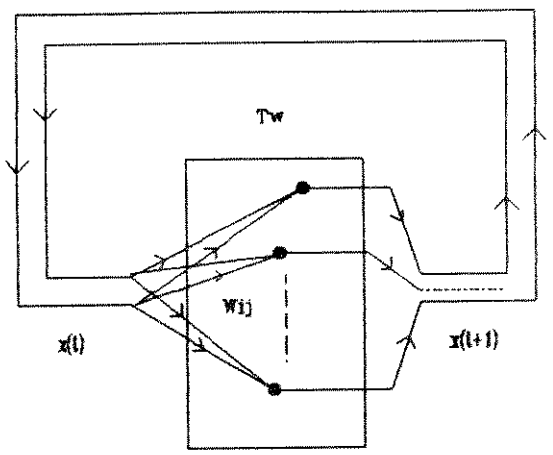

Fig.2.8. Esquema de uma memória autoassociativa.

3. Fundamentos Teóricos da Neurocomputação

3.1. Aspectos Gerais

Neste capítulo são tratados os aspectos relativos ao tratamento matemático de algumas arquiteturas de sistemas neurais artificiais. Os fundamentos matemáticos pretendem predizer e mostrar as capacidades e limitações do processamento de informação dos modelos supondo *a priori*, que seja possível o processamento de informação através da utilização da dinâmica de muitas células de processamento elementares, interagindo mutuamente de forma paralela, onde cada elemento do tipo neurônio possui a capacidade de aprender. Conseqüentemente, os métodos de análise tratados aqui utilizam a representação distribuída de informação com capacidade de aprendizagem.

Um método matemático poderoso, que permite a análise de modelos gerais de redes neurais, foi proposto por Shun-Ichi Amari [36]. Na sua proposta teórica Amari elucida os princípios básicos do processamento de informação maciçamente paralelo. Sua teoria pode ser aplicada tanto para explicar o funcionamento de modelos biológicos, quanto para analisar e projetar sistemas artificiais de neurocomputação.

A seguir, serão descritos: o método geral para a análise de redes neurais simples (com 1 nível ou camada de elementos de processamento); a neurodinâmica de redes neurais (com conexões recorrentes) e as características básicas do modelo da memória associativa de auto-correlação. Finalmente, será tratada sucintamente a proposta de Amari para uma teoria geral de aprendizagem em computação neural.

3.2. Transformação Neural numa ANN simples de 1 nível.

3.2.1. Características da Rede simples

Se se considerar uma ANN de 1 nível e **k** elementos de processamento, que recebem os mesmos sinais de entrada $\mathbf{x}=(x_1,x_2,\ldots,x_n)$, as saídas são dadas por $z=(z_1,z_2,\ldots,z_k)$. A matriz sináptica é designada por W e constituída pelas conexões w_{ji}, as quais conectam o i-ésimo componente de entrada x_i com o j-ésimo neurônio. A saída z_j do j-ésimo neurônio é dada por:

$$z_j = f\left(\sum_{i=1}^{n} w_{ji}\, x_i - \theta_j\right), \qquad j=1, \ldots, k \qquad (3.1)$$

onde f(.) é uma função de transferência não linear e θ_j é o limiar do neurônio j.

A eq. (3.1) descreve o comportamento de uma rede simples de 1 nível (ou camada) sem conexões recorrentes. Esta rede está representada na Fig. 3.1.

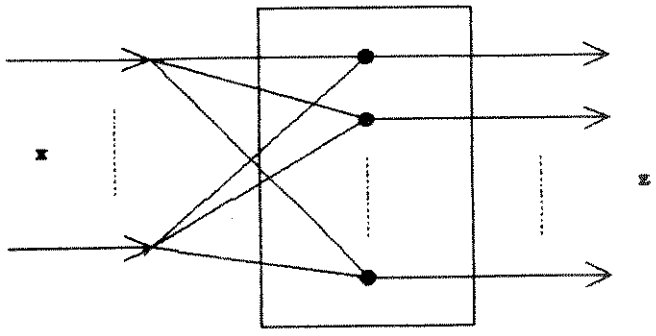

Fig.3.1. Esquema de uma rede simples com 1 nível de Elementos de Processamento.

Se um conjunto de sinais de saída é designado pelo vetor $z=(z_1, \ldots, z_k)$, a rede neural transforma um vetor de entrada $\mathbf{x}$ num vetor de saída z. A transformação é escrita como :

$$z = T_W \mathbf{x} \qquad (3.2)$$

onde T_w representa um mapeamento não linear definido por (3.1) e $W=(w_{ij})$ representa a matriz de conexões ou matriz sináptica. Desta forma, uma rede de 1 nível define uma transformação ou mapeamento de um espaço de sinais de entrada $X=\{\mathbf{x}\}$ num espaço de sinais de saída $Z=\{z\}$ denotada por

$$T_w : X \rightarrow Z$$

A fim de facilitar o estudo do mapeamento T_w realizado pela rede neural, Amari utilizou um modelo de neurônio binário simples, de tal modo que os sinais de entrada e saída assumem valores binários +1 e -1. Desta forma a função f(.) é dada pela função *sgn* (sinal) definida por :

$$\mathrm{sgn}(u) = \begin{cases} 1, & u \geq 0 \\ -1, & u < 0 \end{cases}$$

Com o propósito de simplificar a análise, fez-se $\theta_j = 0$, tal que

$$z = T_w \mathbf{x} = \mathrm{sgn}(W \mathbf{x})$$

Uma possibilidade para abordar o estudo das características de T_w é definir a capacidade da rede de 1-nível pelo número máximo m de pares de entrada-saída $(\mathbf{s}_i, \mathbf{q}_i)$, i=1, ... ,m, viabilizando a existência de uma rede que realize a transformação de entrada-saída para quase todos os pares, com $\mathbf{q}_i$ dado pela expressão seguinte:

$$\mathbf{q}_i = T_w \mathbf{s}_i \qquad i = 1, \ldots, m.$$

No seu trabalho, Amari utilizou o método estatístico para abordar o estudo de redes neurais complexas (de grande tamanho). Nesse caso, os pesos de conexão são

considerados como se fossem determinados de forma aleatória, de acordo com uma distribuição de probabilidade. Tal método estatístico foi utilizado para ilustrar as características do mapeamento T_w. Na sua abordagem são tratados dois casos típicos :

1.- Redes completamente aleatórias, nas quais os componentes w_{ij} são distribuídos independentemente.

2.- Redes de memória associativa, onde os w_{ij} não são independentes, mas estão determinados por um número pequeno de parâmetros aleatórios.

3.2.2. Mapeamento de redes completamente aleatórias.

Nesta secção são descritas algumas propriedades de redes completamente aleatórias. Amari introduziu em [31] o conceito de redes completamente aleatórias como aquelas onde os pesos w_{ij} o os limiares θ_i são variáveis aleatórias, mais especificamente, tais redes são definidas pela matriz W, cujos componentes w_{ij} são a realização de variáveis aleatórias independentes sujeitas a uma distribuição normal $N(\varpi,\sigma^2)$ com média ϖ e variância σ^2. Quando todos os pesos w_{ij} são variáveis aleatórias independentes sujeitas a uma mesma distribuição de probabilidade, e quando todos os limiares θ_i são também variáveis aleatórias independentes sujeitas a uma outra distribuição de probabilidade, a rede aleatória é chamada de simétrica. O estímulo de entrada à rede é definido por :

$$u_i = \sum_{j=1}^{n} w_{ij} x_j$$

e a saída é descrita por :

$$z_i = \mathrm{sgn}\, u_i$$

Sendo os componentes determinados aleatoriamente, dado $\mathbf{x}$, os componentes u_i (i=1, ... ,k) também estão distribuídos de forma aleatória. Assim, u_i é uma combinação linear de w_{ij}, estando distribuída de forma normal.

Se $\mathbf{x}$ é mapeado em z, de acordo com

$$z = T_w \mathbf{x} ,$$

espera-se que os sinais $\mathbf{x}'$, pertencentes a uma vizinhança de $\mathbf{x}$, sejam mapeados a uma vizinhança de z. A distância normalizada entre $\mathbf{x}$ e $\mathbf{x}'$ está definida por:

$$D_x(\mathbf{x}, \mathbf{x}') = 1/2n \left(\sum_{i=1}^{n} | x_i - x_i' | \right) \qquad (3.3)$$

D_x é conhecida como a **distância de Hamming normalizada**. A distância entre z e z', denotada por $D_z(z,z')$, é definida de forma similar. Na Fig. 3.2 está representado o

particionamento dos espaços de sinais **x** e **z**. O mapeamento é feito através de T_w e as distâncias normalizadas estão esquematizadas por uma circunferência de raio D_x e D_z, respectivamente.

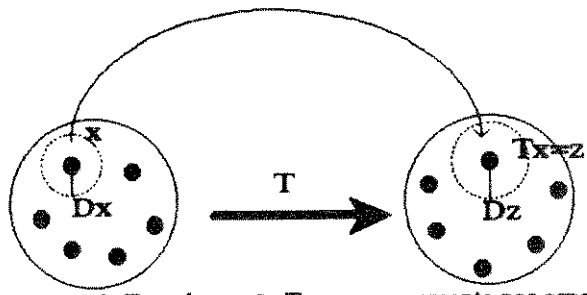

Fig.3.2. Esquema da Transformação T_w e o mapeamento nos espaços X e Z.

Se **x'** é um sinal a uma distância D_x de **x**, é desejável quantificar a distância D_z entre $z=T_w\,x$ e $z'=T_w\,x'$. A relação entre ambas as distâncias define a estabilidade do mapeamento T_w. Se se considerar **x'** como uma versão ruidosa de **x**, com uma taxa de ruído D_x, a taxa de ruído da saída z' é dada por D_z. Quando $D_z < D_x$, isto implica uma redução do ruído pela transformação T_w. Caso contrário, quando $D_z > D_x$, tem-se uma amplificação do ruído. Tal tipo de rede é convenientemente utilizada para a detecção de pequenas diferenças entre sinais similares. No caso de redes completamente aleatórias tem-se que [30] :

$$Dz = \frac{2}{\P}\,\mathrm{Sen}^{-1}\sqrt{Dx} \qquad (3.4)$$

para D_x pequenos, tem-se a aproximação seguinte :

$$Dz = \frac{2}{\P}\sqrt{Dx} \qquad (3.5)$$

A derivada aproximada é dada por :

$$\frac{dDz}{dDx} = \frac{1}{\P}\,\frac{1}{\sqrt{Dx}} \qquad (3.6)$$

A expressão (3.6) tende ao infinito quando $D_x \to 0$. Isto significa que uma vizinhança de **x** muito pequena é expandida e mapeada numa vizinhança muito grande de $z = T_w\,x$.

Tal tipo de mapeamento é utilizável no reconhecimento de diferenças entre sinais similares, uma vez que tais diferenças entre sinais são aumentadas nos sinais de saída. Esta utilização é completamente contrária à propriedade desejada da redução do ruído acrescentado a **x**, ao ser transformado e mapeado num sinal de saída z.

3.2.3. Mapeamento Associativo

Dado m pares de sinais $(\mathbf{s}^1,\mathbf{q}^1)$, ... ,$(\mathbf{s}^m,\mathbf{q}^m)$, define-se uma rede que transforma $\mathbf{s}^\mu$ em $\mathbf{q}^\mu$ (μ=1,2, ... ,m) e que satisfaz a relação:

$$T_w\, \mathbf{s}^\mu = \mathbf{q}^\mu ; \qquad \mu = 1, \ldots , m$$

o que define a relação de entrada-saída da rede.

Deseja-se que este tipo de rede possua propriedades que permitam uma forte capacidade de redução de ruído, de tal forma que ao ingressar na entrada uma versão de $\mathbf{s}^\mu$ contaminada por ruído, este seja reduzido ao mínimo na sua saída pela transformação T_w. Uma forma de construir tal tipo de rede é a memória de matriz de correlação, tal como aquela proposta por Kohonen [38], sendo a matriz de correlação W definida pela expressão seguinte:

$$W = 1/n \left[\sum_{\mu=1}^{m} \mathbf{q}^\mu (\mathbf{s}^\mu)^T \right] \qquad (3.7)$$

onde $(\mathbf{s}^\mu)^T$ é a transposição do vetor coluna $\mathbf{s}^\mu$. Na sua forma componente, (3.7) pode ser expressa por:

$$w_{ij} = 1/n \left[\sum_{\mu=1}^{m} (q_i^\mu\, s_j^\mu) \right]$$

onde s_j^μ é o j-ésimo componente de $\mathbf{s}^\mu$. Se os $\mathbf{s}^\mu$ são mutuamente ortogonais, a expressão $W\, \mathbf{s}^\nu = \mathbf{q}^\nu$ é mantida para qualquer valor de ν.

A capacidade da rede é especificada pelo número máximo de pares de sinais que podem ser memorizados. A fim de ilustrar o desempenho típico de uma rede associativa de correlação, é tratado o caso onde $\mathbf{s}^\mu$ e $\mathbf{q}^\mu$ são vetores gerados aleatoriamente. Os componentes s_i^μ e q_i^μ são determinados de forma aleatória e independente, para serem iguais a +1 ou -1, com a mesma probabilidade 1/2. Conseqüentemente, a matriz $W=(w_{ij})$ é determinada de forma aleatória, embora, neste caso os componentes w_{ij} não sejam independentes.

Se $\mathbf{x}$ é uma versão ruidosa de um dos padrões armazenados $\mathbf{s}^\mu$, por ex. $\mathbf{s}^1$, cuja distância de $\mathbf{s}^1$ é D_x, deseja-se quantificar a distância D_z entre a saída $T_w \cdot \mathbf{x}$ e a saída desejada $\mathbf{q}^1$,

$$D_z = D_z(T_w \cdot \mathbf{x})$$

Quando para alguma função K(.) é mantida a relação

$$D_z = K(D_x) ,$$

tal função representa a estabilidade da rede de memória associativa de tipo correlação. Particularmente, K(0) descreve a precisão do mapeamento $\mathbf{s}^\mu \rightarrow \mathbf{q}^\mu$.

Amari demonstrou que neste tipo de rede, a taxa de erro da saída D_z é dada pela expressão :

$$D_z = \phi \{ - (1-2D_x) / \sqrt{D_x} \} = \text{Prob} \{ N_i > 1-2D_x \} \quad (3.8)$$

onde $N_i = 1/m \left(\sum_{\mu=2}^{m} \sum_{j=1}^{n} q_i^{\mu} s_j^{\mu} x_j^{\mu} \right)$

n é o número de dimensão,

r é a taxa do número de pares de padrões armazenados, dada por r=m/n; e

$\phi(.)$ é definido como o erro integrado [36].

Se r não é muito grande, D_z é desprezível comparado com D_x. A derivada de D_z é dada por :

$$\frac{dDz}{dDx} = \sqrt{\frac{2}{\pi r} \exp \left\{ - \frac{(1-2Dx)^2}{2r} \right\}} \quad (3.9)$$

Na Fig. 3.3. está representada a taxa de erro de saída D_z, em função de D_x, obtida a partir de (3.8). Desta figura, pode-se deduzir que a rede possui uma forte propriedade de redução de ruído, no sentido de que o ruído acrescentado a s^{μ} é fortemente reduzido pelo mapeamento T_w.

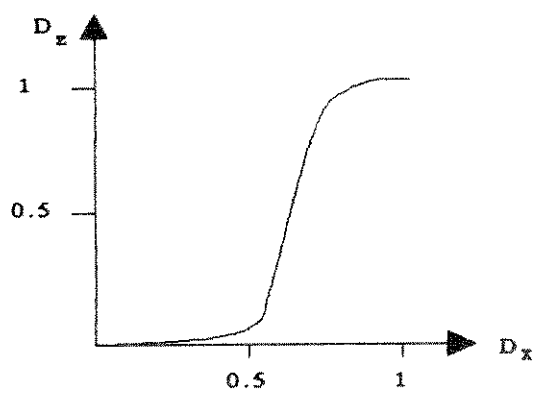

Fig.3.3. Esquema da Eq. (3.8).

3.3. Neurodinâmica de Redes Recorrentes.

3.3.1. Características básicas.

Nesta secção será tratada uma rede neural com conexões recorrentes. A arquitetura geral da rede é representada na Fig. 3.4.

Se w_{ij} representa o peso de conexão do j-ésimo neurônio ao i-ésimo neurônio, $x_i(t)$ é o estado ou a saída do neurônio i no tempo t, onde cada elemento de

processamento assume os valores +1 ou -1. Quando cada neurônio trabalha de forma síncrona, em tempos discretos t=0,1,2, ... , o comportamento da rede é descrito como :

$$\mathbf{x}_i(t) = \mathbf{sgn}\left(\sum_{j=1}^{n} \mathbf{w}_{ij}\,\mathbf{x}_j(t) - \theta_i + \mathbf{I}_i\right) \qquad (3.10)$$

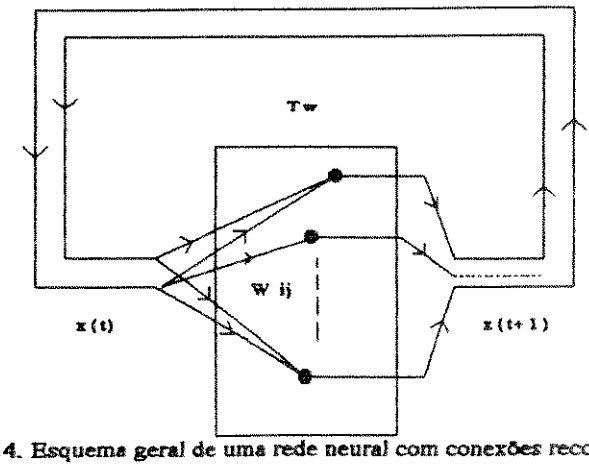

Fig.3.4. Esquema geral de uma rede neural com conexões recorrentes.

Em (3.10), θ_i representa o limiar do neurônio i e I_i representa uma soma ponderada de um estímulo externo (*bias*) ingressando no i-ésimo elemento. A fim de simplificar a análise, I_i foi incluído no termo θ_i, colocando $\theta_i - I_i$ como o novo termo θ_i, e desprezando I_i. Desta forma, utilizando o operador linear T_w, tem-se

$$T_w\ \mathbf{x} = \mathbf{sgn}\ (W\,\mathbf{x} - \theta)$$

a equação (3.10) é então escrita como:

$$\mathbf{x}(t+1) = T_w\,\mathbf{x}(t) \qquad (3.11)$$

O vetor $\mathbf{x}(t)$ é considerado o estado da rede no tempo t, e a expressão (3.11) representa a equação da transição de estado, que descreve o comportamento da rede. O espaço de estado X consiste de 2^n vetores $\mathbf{x}$, cujos componentes estão representados, neste caso, pelos valores ± 1.

A transição de estado T_w define um mapeamento de X nele mesmo, onde $T_w\mathbf{x}$ é o estado seguinte a $\mathbf{x}$. Pode-se construir um grafo que represente as transições de estado, com arcos dirigidos ligando dois nós $\mathbf{x}$ e $T_w\mathbf{x}$. Cada nó (representando o estado) possui só um arco dirigido dele ao estado seguinte. As propriedades dinâmicas desta rede estão representadas na Fig. 3.5.

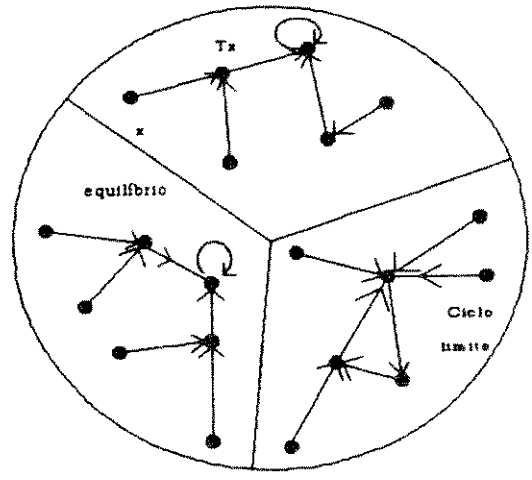

Fig.3.5. Grafo representando o espaço de transições de estado de T_w.

O estado **x** representa um ponto de equilíbrio de X, quando se mantem a seguinte condição em iterações sucessivas :

$$\mathbf{x} = T_w \mathbf{x} ;$$

quando **x** possui um arco fechado dirigido a si mesmo, este representa um ponto de equilíbrio.

Uma seqüência $\{x_1, \ldots, x_k\}$ é considerada um ciclo de período k, quando se mantem a seguinte condição:

$$\mathbf{x}_{t+1} = T_w \ \mathbf{x}^t \qquad t = 1, \ldots, k\text{-}1$$

$$\mathbf{x}_1 = T_w \mathbf{x}_k$$

A condição anterior é representada na Fig. 3.5 por um *laço* de *k* iterações no grafo de transições de estado.

3.3.2. Comportamento dinâmico de Redes Recorrentes.

No estudo das características dinâmicas de redes recorrentes aleatórias, Amari introduziu o conceito de variável de estado macroscópica e de estado microscópico.

A análise teórica [35], [36] foi feita para redes aleatórias compostas por elementos simples, do tipo de ativação limiar (on-off) ou de neurônios formais do tipo de McCulloch-Pitts [35], através da extração de parâmetros que dependem das estatísticas dos pesos de conexão e dos limiares dos elementos de processamento. Amari demonstrou que tais parâmetros são suficientes para determinar as características dinâmicas das redes, pelo que o critério de estabilidade é dado explicitamente nos termos daqueles parâmetros.

Na análise das características das redes, foram escolhidos como parâmetros estatísticos a variância σ^2_w e a esperança E_w dos pesos w_{ij}, bem como a esperança E_θ e a variância σ^2_θ dos limiares θ_i. As equações que governam o comportamento dinâmico das redes aleatórias recorrentes foram derivadas nos termos de tais parâmetros.

O estado microscópico de uma rede aleatória é descrito pelos potenciais $u_i(t)$ dos elementos componentes. O potencial representa o estado de um elemento e, no caso de redes biológicas, este pode ser comparado ao potencial médio no tempo da membrana de um neurônio. No caso da análise estatística das redes, na maioria das vezes, o que interessa não é o comportamento microscópico de cada elemento, mas o comportamento macroscópico geral da rede, o qual é descrito por equações de estado.

Uma variável de estado macroscópica é função do estado microscópico $\mathbf{x}$ e resume algumas características médias do estado. Um exemplo é representado pelo nível de atividade definido por:

$$A(\mathbf{x}) = 1/n \ (\sum x_i) \qquad (3.12)$$

$A(\mathbf{x})$ representa a taxa de neurônios excitados. O nível de atividade no tempo t é descrito por:

$$A_t = A\{\mathbf{x}(t)\}$$

e é desejado que para a maioria das redes geradas de forma aleatória, se mantenha uma equação dinâmica do tipo

$$A_{t+1} = F(A_t) \qquad (3.13)$$

A fim de descobrir características microscópicas comuns no grafo de transições de estado, é conveniente definir uma variável macroscópica que seja uma função de dois ou mais estados macroscópicos. Por exemplo, pode-se utilizar a distância $D(\mathbf{x},\mathbf{y})$ entre dois estados, que é definida por:

$$D_t = D\{\mathbf{x}(t),\mathbf{y}(t)\} \qquad (3.14)$$

onde $\mathbf{x}(t)=T_w{}^t\mathbf{x}(0)$ e $\mathbf{y}(t)=T_w{}^t\mathbf{y}(0)$ são as seqüências das transições de estado começando especificamente em $\mathbf{x}(0)$ e $\mathbf{y}(0)$. Desta forma, pode-se ter uma relação dinâmica do tipo:

$$D_{t+1} = G(D_t) \qquad (3.15)$$

para assim estudar a forma como uma diferença inicial no estado se desenvolve no transcurso da dinâmica de transições. Mais especificamente, para dois estados $\mathbf{x}$ e $\mathbf{y}$, separados por uma distância D, pode ser interessante, por exemplo, averiguar a distância D' entre os seus estados seguintes $T_w\mathbf{x}$ e $T_w\mathbf{y}$.

Amari investigou tais características em [35],[36], incluindo na sua análise tanto redes assimétricas quanto simétricas, nas quais se impõe a condição de simetria $w_{ij}=w_{ji}$.

3.4. Memória asociativa de auto-correlação.

3.4.1. Características Básicas.

Considere-se uma rede neural com conexões recorrentes e uma matriz sináptica determinada por :

$$w_{ij} = (1/n) \sum_{\mu=1}^{m} s_i^{\mu} s_j^{\mu}, \qquad W = (1/n) \sum_{\mu=1}^{m} s^{\mu} (s^{\mu})^T.$$

Este modelo é conhecido como uma memória associativa de auto-correlação, no qual espera-se que os m vetores s^{μ} sejam os estados de equilíbrio da rede. Desta maneira, os m vetores s^{μ} são memorizados na rede conformando os estados de equilíbrio. Os vetores (ou padrões) são armazenados de forma distribuída e superposta, podendo ser recuperados através de um processo de recuperação dinâmica. Se os vetores são mutuamente ortogonais, tem-se que

$$Ws^{\mu} = n\, s^{\mu}; \qquad \text{ou } T_w = s^{\mu},$$

onde os vetores $s^1,\ldots,s^m$ são os estados de equilíbrio da rede. Com o propósito de determinar o comportamento dinâmico do modelo, é conveniente analisar o caso onde $s^m=(s_i^{\mu})$, $\mu=1,\ldots,m$, é um conjunto de m vetores determinados de forma aleatória e independente e onde cada componente s_i^{μ} assume o valor +1 ou -1, com uma probabilidade de 0,5. No caso onde os s^{μ} são gerados aleatoriamente, a menos que o número m de padrões seja muito pequeno, eles não são necessariamente estados de equilíbrio. Neste caso, $W=(w_{ij})$ é uma matriz aleatória dependendo de s^{μ}, onde os seus componentes w_{ij} não são mutuamente independentes. Pelo contrário, eles estão fortemente correlacionados.

Se os padrões s^{μ} são os estados estáveis da rede, ao inicializá-la num estado x próximo de s^{μ}, espera-se que o estado da rede convirja para s^{μ}, através do mecanismo de transição de estados. Ainda que s^{μ} não seja exatamente um ponto de equilíbrio da rede, se existem estados de equilíbrio numa pequena vizinhança de s^{μ}, ao inicializar a rede em x e ir evoluindo através da transição de estados, o estado da rede vai se aproximando de s^{μ} e é atraído a um dos padrões, permanecendo próximo a s^{μ}. Este é o mecanismo considerado como um modelo de memória associativa, no qual um número de padrões s^{μ} são armazenados na rede de uma forma distribuída e superposta.

A seguir, será brevemente analisado o comportamento dinâmico desse modelo, segundo a visão de Nakano [39] e Amari [35].

3.4.2. Comportamento Dinâmico da memória associativa.

Se é considerado um padrão do tipo

$$s^1 = (1, \ldots, 1)$$

o comportamento dinâmico da transição de estado na vizinhança de $\mathbf{s}^1$ é dado por:

$$\mathbf{x}(t+1) = T_w\, \mathbf{x}(t)$$

Os componentes dos outros padrões $\mathbf{s}^\mu$ (para μ=1) são determinados de forma aleatória e independente. Se $\mathbf{x}$ é um estado cuja distância de $\mathbf{s}^1$ é dado por D, o estado seguinte é expresso por $\mathbf{x}'=T_w\mathbf{x}$.

A atividade de um padrão $\mathbf{s}^\mu$ é dado por

$$a^\mu = (1/n)\, \mathbf{s}^\mu\, \mathbf{x} \qquad (3.16)$$

onde

$$a^1 = (1/n)\, \mathbf{s}^1\, \mathbf{x} = 1 - 2D \qquad (3.17)$$

O vetor $\mathbf{u}$, definido por $\mathbf{u}=W\mathbf{x}$, permite definir o vetor $\mathbf{x}'=\mathbf{sgn}(\mathbf{u})$, cujos componentes são definidos por:

$$u_i = (1/n) \sum_{\mu,j} s_i^\mu\, s_j^\mu\, x_j = (1\text{-}2D) + N_i, \qquad (3.18)$$

onde

$$N_i = (1/n) \sum_{\mu=2}^{m} s_i^\mu\, \mathbf{s}^\mu\, \mathbf{x} = \sum_{\mu=2}^{m} s_i^\mu\, a^\mu. \qquad (3.19)$$

O vetor $N=(N_i)$ é considerado um componente de ruído (*crosstalk*), o que produz uma interferência na recuperação correta de $\mathbf{s}^1$, por influência da superposição dos outros padrões $\mathbf{s}^\mu$.

No comportamento dinâmico do processo de recuperação de um padrão armazenado na memória, quando a rede é inicializada num estado $\mathbf{x}$, numa vizinhança de $\mathbf{s}^1$, espera-se que o estado seguinte, $T_w\mathbf{x}$, fique mais próximo de $\mathbf{s}^1$ do que o estado $\mathbf{x}$. Amari determinou, teoricamente, a distância D' entre $\mathbf{s}^1$ e $T_w\mathbf{x}$, demonstrando que $\mathbf{x}$ vai se aproximando de $\mathbf{s}^1$ através da transição de estados. Quando o valor de r=m/n não é muito grande (r=m/n expressa a taxa entre os padrões armazenados e o número de neurônios), tem-se que D'< D.

Seja D_t a distância entre $\mathbf{x}(t)=T_w{}^t\mathbf{x}(0)$ e $\mathbf{s}^1$; logo, $D_{t+1}=\mathbf{x}(t+1)$ define o processo dinâmico de recuperação do padrão $\mathbf{s}^1$, mostrando a forma com que o valor x(t) vai se aproximando de $\mathbf{s}^1$, por meio da transição de estados.

Uma característica interessante no processo de recuperação é que existe um valor limite r_l=0,15 (demonstrado por Hopfield em [17], usando simulações por computador; discutido em [40] por Tarassenko). De tal modo que quando $r<r_l$, D_t se aproxima muito do valor 0, se o valor D_0 está dentro de um limite $D^*(r)$ determinado por r. Além disso, quando a rede não consegue recuperar $\mathbf{s}^1$, no caso de $D_0>D^*(r)$, o estado seguinte $\mathbf{x}(1)$ se aproxima de $\mathbf{s}^1$ fazendo, inclusive, $D_1<D^*(r)$.

3.4.3. Capacidade da Memória Associativa.

A condição que garante que s^{μ} é um estado de equilíbrio foi dada por Mc Eliece et al. [41] e A. Kuh & B. Dickinson [42]. Um teorema que define a capacidade da memória associativa foi demonstrado por S. Amari em [36], tal teorema determina que o número m de padrões armazenados, necessários para que s^{μ} seja um estado de equilíbrio com probabilidade muito próxima a 1, é dada pela expressão:

$$m = n / (2 \log n) \qquad (3.20)$$

onde n é o número de neurônios da rede.

O teorema mostra que a capacidade da memória associativa necessária para recuperar, exatamente, o padrão s^{μ}, é muito pequena. Se, pelo contrário, aceita-se a recuperação de um padrão muito próximo de s^{μ}, a capacidade m da rede é de aproximadamente 0,15n.

3.5 Fundamentos do aprendizagem neural.

Um neurônio possui a capacidade de modificar os seus pesos de conexão $\mathbf{w}=(w_1,...,w_n)$, dependendo dos sinais de entrada $\mathbf{x}=(x_1,...,x_n)$ recebidos e dos sinais de ensino associados àquelas entradas (algumas vezes chamadas de sinais de erro). Em certos casos, os sinais de ensino não são fornecidos, e o neurônio modifica os seus pesos, dependendo somente do seu próprio estado e dos sinais de entrada. Este último caso é chamado de aprendizado não supervisionado, e o esquema de aprendizado é conhecido como **auto-organização**. Na construção de uma teoria geral do aprendizado neural, Amari [36] considerou uma situação onde um neurônio recebe sinais de entrada $\mathbf{x}$ de uma fonte de informação I, aos quais precisa se adaptar. Tais sinais de entrada são considerados como um conjunto de sinais de treinamento $\mathbf{x}_{\alpha}$, $\alpha=1,...,k$ (vide Fig. 3.5).

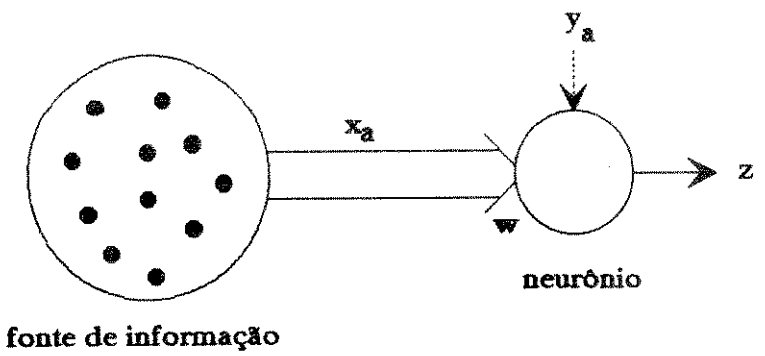

Fig.3.5. Esquema geral de aprendizado ($x_a=x_{\alpha}$; $y_a=y_{\alpha}$).

A freqüência relativa de $\mathbf{x}_\alpha$, é descrita por p_α, $\Sigma p_\alpha = 1$, onde o sinal de ensino associado com a entrada $\mathbf{x}_\alpha$ é y_α. A fonte de informação I, que provem do meio, é descrita por uma estrutura de pares de sinais, da seguinte forma :

$$I = \{(\mathbf{x}_\alpha, y_\alpha, p_\alpha),\ \alpha=1,...k\};$$

a expressão anterior pode ser escrita, de forma mais geral, como:

$$I = \{\mathbf{x}, y, p(\mathbf{x},y))\},$$

onde $p(\mathbf{x},y)$ representa a distribuição de probabilidade do par $(\mathbf{x},y)$. No caso de sistemas não supervisionados, tem-se :

$$I = \{(\mathbf{x}, p(\mathbf{x},y))\}.$$

Na regra de aprendizado geral assumida por Amari, o vetor de pesos sinápticos $\mathbf{w}$ é incrementado em proporção ao produto da entrada $\mathbf{x}$ pelo sinal de aprendizado r, onde r é determinado como uma função de $\mathbf{x}$, $\mathbf{w}$ e y, quando este último é fornecido. Através da escolha adequada da função r, a maioria das regras de aprendizagem propostas são formuladas de acordo com o esquema que é descrito a seguir.

Para o caso de tempo discreto, uma regra de aprendizado geral pode ser descrita como

$$\mathbf{w}(t+1) = (1-e)\,\mathbf{w}(t) + cr\,\mathbf{x}(t) \qquad (3.21)$$

e para o caso contínuo tem-se

$$\frac{d\mathbf{w}}{dt} = -e\,\mathbf{w}(t) + cr\,\mathbf{x}(t) \qquad (3.22)$$

O termo $(1-e)\mathbf{w}(t)$ pode ser substituído em (3.21) por $(1-er)\mathbf{w}$, em (3.22) $-e\mathbf{w}(t)$ pode ser substituído por $-er\mathbf{w}$. Desta forma, o termo de decaimento é eliminado, quando o sinal de aprendizagem r=0. A equação anterior é conhecida como equação de aprendizagem.

Dependendo da forma de r, a aprendizagem da rede assume características particulares. Por exemplo, no caso de r=z, dado de acordo com a expressão

$$r = z = f(\mathbf{w}\,\mathbf{x} - \theta)$$

a regra de aprendizagem é conhecida como regra de Hebb. Quando f(.) é dada pela *função de passo* definida por

$$f(u) = \begin{cases} 1, & u > 0 \\ 0, & \text{caso contrário} \end{cases}$$

o peso $\mathbf{w}$ é incrementado em proporção à entrada $\mathbf{x}$, somente quando a entrada não é inibitória. Se r é dado pelo sinal de erro r=y-z, tem-se o caso da regra de aprendizagem do Perceptron [45]. Quando o sinal de aprendizado r é o mesmo sinal de ensino y, r=y, tem-se a regra de aprendizado de correlação, a qual é utilizada no modelo da memória associativa.

Nestes esquemas, descreve-se o comportamento geral do aprendizado de somente um elemento neural. Nele, a fonte de informação I é fixa. Quando é tratado o aprendizado numa rede, o comportamento dela é modificado de forma cooperativa, devido à ação conjunta de vários neurônios, onde cada um deles pode modificar seus pesos de conexão, de acordo com uma equação ou regra de aprendizado. Conseqüentemente, a fonte de informação externa I, associada a cada neurônio, varia com a modificação dos pesos de conexão dos neurônios, já que I é dada pelo comportamento dos outros neurônios da rede. Dois exemplos amplamente conhecidos de redes com capacidade de aprendizagem supervisionada são a *Backpropagation* e as redes multicamadas, formadoras de mapas topológicos de informação [37], [43].

3.6. Características gerais do mapeamento topológico.

No mapeamento das características topológicas de uma rede neural, existem representações internas do meio exterior, captadas por um sistema sensorial que decodifica eventos, objetos ou relações, combinando suas características. Tais características são representadas como padrões de atividade dos neurônios no tempo-espaço num campo neural. O campo neural representa uma rede na qual os neurônios são arranjados num espaço bidimensional, como num córtex cerebral. Em geral, dado um conjunto de sinais de informação I, uma rede neural é capaz de formar, automaticamente, uma representação interna dos sinais. Particularmente, ao se ter um sinal em I, representado por um estímulo local, numa posição específica de um campo neural, é criado um mapa dos sinais nesse campo. Neste tipo de mapeamento, conhecido como *mapeamento cortical* ou *mapa neural de informação*, é de interesse estudar as relações topológicas entre o mapa e o espaço original I de sinais; a resolução e a estabilidade do mapeamento; o fator de amplificação dos sinais no mapa; etc.

Um algoritmo simples, para a formação de mapas de características topológicas, foi desenvolvido por T. Kohonen [18], [43], [44]. Kohonen estudou extensivamente as operações de transformação do tipo de recuperação auto-associativa e hetero-associativa de informação, como uma generalização do processo simples de estímulo-resposta, independentemente da implementação física particular. Uma das propriedades mais interessantes descobertas por ele é a descrição da operação de memórias associativas distribuídas (vide Cap. 2, secção 2.2).

3.7. Memórias Associativas Bidirecionais (BAM).

3.7.1. Características Gerais.

As Memórias Associativas Bidirecionais são constituídas por uma rede neural não-linear com 2 níveis de unidades de processamento elementares (neurônios). As BAMs têm sido amplamente utilizadas como associadoras de padrões, principalmente devido aos estudos originais de Kosko [46]. Kosko estendeu o processo de codificação unidirecional a um processo de 2-direções. O seu processo bidirecional para memórias hetero-associativas amplia as memórias auto-associativas de uma direção, como no modelo de J.J. Hopfield. Na secção seguinte será sucintamente descrita a operação de uma BAM discreta, a qual possui a importante propriedade de recuperar pares de vetores de treinamento na presença de ruído.

3.7.2. Operação da BAM discreta.

Uma BAM conecta dois níveis ou camadas de neurônios. Cada elemento pertencente a um nível é conectado a todos os elementos do segundo nível e o tipo de conexão é simétrica, $w_{ij}=w_{ji}$. Kosko e outros autores [46],[47] descobriram que o desempenho das BAMs é melhorado se são utilizados vetores e matrizes bipolares, em vez de vetores e matrizes constituídos por elementos que assumem valores binários. As matrizes e vetores bipolares são obtidos a partir dos seus semelhantes binários, substituindo os elementos de valor 0 por elementos de valor -1. Por ex.: um padrão binário A=(101100) é substituído pelo padrão X=(1-111-1-1).

Se existem N pares de vetores de treinamento do tipo:

$$\{(A_1,B_1), (A_2,B_2), \ldots , (A_i,B_i), \ldots , (A_N,B_N)\}$$

onde

$$Ai = (a_{i1}, a_{i2}, \ldots ,a_{in})$$

$$Bi = (b_{i1}, b_{i2}, \ldots, b_{ip})$$

podendo a_{ij}, b_{ij} assumir os estados ON ou OFF.

No modo binário, ON=1, OFF=0.

No modo bipolar, ON=1, OFF=-1.

É formada uma matriz de correlação da seguinte forma:

$$M = \sum_{i=1}^{n} X_i^T Y_i \qquad (3.23)$$

onde $X_i(Y_i)$ é o modo bipolar de $A_i(B_i)$.

Na operação da BAM deseja-se recuperar um dos pares mais próximos (A_i, B_i), quando é apresentado qualquer par (α, β) à rede, como uma condição inicial.

A partir do valor (α,β), é determinada uma seqüência finita (α',β'), (α'',β''), ... até que seja alcançado um ponto de equilíbrio final (α_F,β_F).

onde

$$\beta' = \phi(\alpha M)$$

$$\alpha' = \phi(\beta' M)$$

$$\phi(F) = G = (g_1, g_2, \cdots, g_n)$$

$$F = (f_1, f_2, \cdots, f_n)$$

$$g_i = 1, \quad f_i > 0$$

$$g_i = \begin{cases} 0 \text{ (binário)}, & f_i < 0 \\ -1 \text{ (bipolar)} & \end{cases}$$

$$g_i = g_i \text{ prévio}, \quad \text{se } f_i = 0$$

Para qualquer ponto (α,β) é definida uma função de energia dada por

$$E = -\alpha M \beta^T \qquad (3.24)$$

Kosko demonstrou em [38] que, em cada ciclo de decodificação, a energia dada pela expressão (3.24) diminui, e que o número dos pares (a',b') mais aqueles obtidos a partir de iterações seguintes, é finito. Desta forma, haverá um número finito de ciclos de decodificação até se chegar num par (α_F,β_F) com a propriedade de que a expressão

$$E = -\alpha_F M \beta_F^T \qquad (3.25)$$

é um mínimo local. Uma propriedade da rede, demonstrada por Wang et al. em [47], que é de muita importância em relação à recuperação do par de treinamento (Ai,Bi), é a seguinte: se a energia E avaliada, utilizando as coordenadas do par (A_i,B_i), isto é:

$$E = -A_i M B_i^T \qquad (3.26)$$

não constitui um mínimo local, o ponto não pode ser recuperado, ainda que a rede seja inicializada em $\alpha=A_i$. Segundo Wang, o método de Kosko não garante que os pares armazenados estejam localizados num mínimo local. Pelo contrário, se (3.26) não produz um mínimo local, quando se começa com $\alpha=A_i$, dado o inicial $\beta=B_i$, (α_F,β_F) será igual ao par (A_i,B_i). Todas estas propriedades foram demonstradas analiticamente por Wang e a simulação de uma proposta alternativa para melhorar o desempenho da rede, é apresentada em [47].

As redes neurais de tipo BAM não foram simuladas neste trabalho.

4. O Modelo Binário de Hopfield

4.1. Introdução

Neste capítulo é tratado particularmente do modelo binário da rede neural proposta por J.J. Hopfield [17] quando funciona como uma memória endereçada pelo conteúdo. O modelo de memória auto-associativa, proposto por Hopfield, possui características muito interessantes quando é utilizado também para resolver complexos problemas de otimização [48]. Embora o modelo seja afetado por algumas desvantagens, especialmente em termos da sua capacidade de armazenamento [40]-[42], as propriedades computacionais emergentes, produzidas pela maciça interação dos elementos de processamento, permitem que ele seja utilizado com muitas vantagens como uma CAM, especialmente devido à relativa simplicidade de operação. A sua arquitetura composta de 1-nível de células elementares tem chamado a atenção da área de microeletrônica, já que tal estrutura facilita a implementação em *hardware*. Utilizando os avanços nas tecnologias VLSI, o modelo de Hopfield tem sido construído tanto de forma digital quanto analógica [6],[49]. Por outro lado, aproveitando o potencial das tecnologias de computação óptica, a construção de memórias associativas, baseadas no modelo de Hopfield, também está sendo cada vez mais estimulada [50], [51].

Neste trabalho é visada especialmente a implementação virtual do modelo de Hopfield num sistema de multiprocessadores, utilizando as técnicas de processamento paralelo (vide Cap. 5). Nas secções seguintes serão descritos o funcionamento e as características do modelo binário síncrono e o seu comportamento dinâmico.

De um determinado ponto de vista, a rede de Hopfield pode ser considerada similar a uma memória associativa matricial. Quando um padrão é representado de forma vetorial, obtem-se uma memória autoassociativa fazendo o produto externo (*outerproduct*) do vetor com ele mesmo (vide Cap. 2 e 3), definindo assim os elementos da matriz. Numa rede, tal operação define os valores dos *pesos* que conectam os elementos de processamento (equivalentes aos nós). Embora similares, desde a perspectiva de uma rede existe uma diferença com a operação das memórias de tipo matriz, o que foi bem definido por Kohonen em [18] e deduzido do trabalho de J.J.Hopfield [17], [28], [48].

Na Fig. 4.1 é representada a estrutura geral da rede de Hopfield, para n elementos de processamento. O vetor $\mathbf{x}=(x_1,\ldots,x_n)$ representa um padrão de entrada desconhecido aplicado no tempo t=0. Após este instante a rede começa a iterar, até que é conseguida uma condição de estabilidade, representando o seu estado final, a ser explicado nas

secções seguintes. O padrão final ao qual a rede converge é representado pelo vetor de estado $\mathbf{x}^*=(x_1^*,....,x_n^*)$.

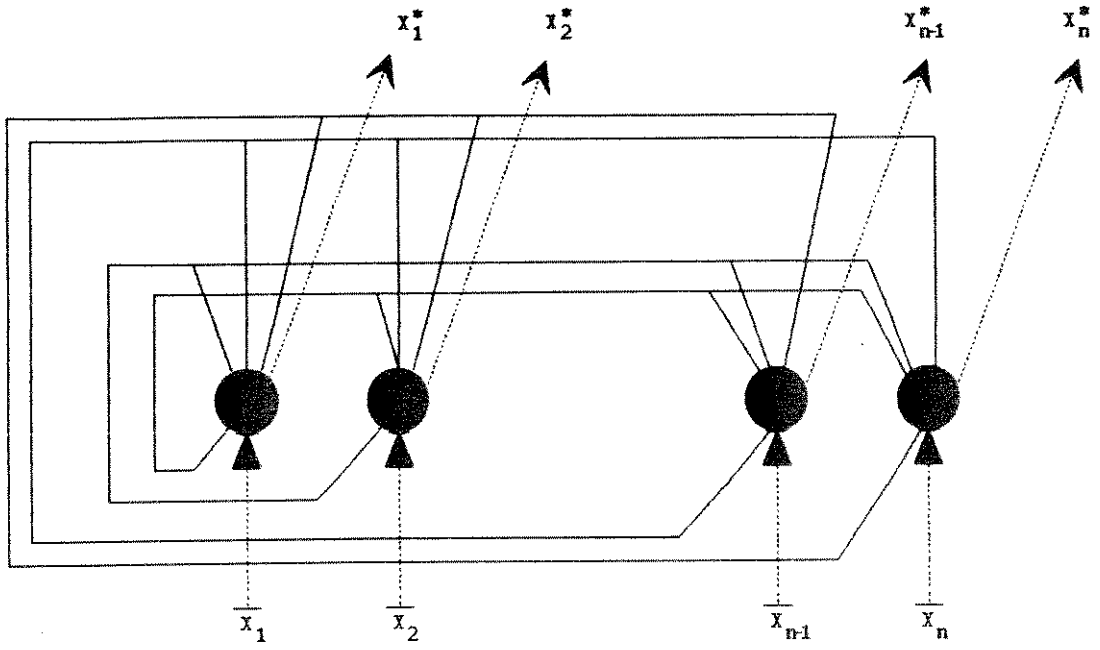

Fig. 4.1. Conectividade da Rede de Hopfield.
Os elementos x_i (i=1,..,n) representam as entradas ao neurônio i.

4.2. Características gerais da Rede Neural de Hopfield.

O modelo binário de Hopfield [17] constitui uma classe de sistema físico, cujo comportamento dinâmico pode ser utilizado como uma forma de memória endereçada pelo conteúdo (CAM) de tipo geral [30]. A evolução no tempo do sistema pode ser descrito por um conjunto de coordenadas $x_1,....,x_n$; que representam os componentes de um vetor $\mathbf{x}$. Um ponto no espaço de estados representa a condição instantânea do sistema, e tal espaço de estados pode ser tanto contínuo quanto discreto.

Supondo que o sistema possui pontos limites estáveis localmente $\mathbf{x}_a,\mathbf{x}_b,...$ (cf.4.4), se ele é inicializado suficientemente próximo de algum ponto $\mathbf{x}_a$, tal que $\mathbf{x}=\mathbf{x}_a+\Delta$ (no espaço de estados binário n-dimensional, Δ representa um deslocamento de $\mathbf{x}_a$), o sistema evoluirá no tempo até $\mathbf{x}\approx\mathbf{x}_a$. Neste sentido, o deslocamento Δ representa uma medida da distância entre dois padrões no sentido de Hamming, no espaço de estados sob consideração. Pode-se considerar que os vetores $\mathbf{x}_a$, $\mathbf{x}_b$, ... formam a informação armazenada no sistema e que o ponto de partida $\mathbf{x}=\mathbf{x}_a+\Delta$ representa o conhecimento parcial do ítem $\mathbf{x}_a$; logo, o sistema gera a informação total $\mathbf{x}_a$. Se tal sistema físico possui uma dinâmica no espaço de fase que é dominado por um número notório de estados estáveis locais aos quais é atraído, então o sistema pode ser considerado como uma memória endereçada pelo conteúdo (um tipo particular de memória associativa).

4.3. Operação do Modelo

Dado um conjunto de M vetores binários bipolares (1,-1) $\mathbf{x}^{(m)}$, m=1,2,3,...,M onde **x** é um vetor de *n* componentes tal que x_i representa o estado do i-ésimo neurônio (i=1,2,3,...,n), estes são armazenados numa *matriz sináptica* de acordo com uma regra de aprendizagem, representada pela expressão:

$$W_{ij} = \sum_{m}^{M} x_i^{(m)} x_j^{(m)} \; ; \qquad (4.1)$$

$$W_{ii} = 0, \qquad W_{ij} = W_{ji} \qquad i,j = 1,2,3,...,n; \quad m=1,..., M;$$

onde os vetores $x_i^{(m)}$ representam os vetores de estado nominais da memória.
Se a memória é endereçada pela multiplicação da matriz W_{ij} por um dos vetores de estado, por ex., $x_i^{(m0)}$,tem-se a estimativa:

$$u_i^{(m0)} = \sum_{j}^{N} W_{ij} x_j^{(m0)} \qquad (4.2)$$

$$= \sum_{i \neq j}^{N} \sum_{m}^{M} x_i^{(m)} x_j^{(m)} x_j^{(m0)}$$

$$= (n-1) x_i^{(m0)} + \sum_{m \neq m0} \alpha_{m,m0} x_i^{(m)} \qquad (4.3)$$

onde $\alpha_{m,m0} = \sum_{j} x_j^{(m0)} x_j^{(m)} \qquad 1 \leq j \leq N$

O segundo termo em (4.3) é uma combinação linear dos vetores armazenados remanescentes e representa um termo de ruído (*cross-talk*) não desejado, pelo que $u_i^{(m0)}$ representa a soma do vetor de entrada amplificado pelo fator (n-1) mais o ruído. Na versão discreta do modelo de Hopfield [40], as mudanças de estado no tempo, para cada neurônio i, se produzem de acordo com o seguinte algoritmo: para cada neurônio i existe um bias externo I_i, alimentado à entrada; logo, P_i é o potencial do elemento i dado por $P_i = u_i^{(m0)} + I_i$. Pode-se considerar que para cada elemento de processamento i, o *bias* externo $I_i = 0$. Portanto o potencial de cada elemento é u_i; logo, cada neurônio computa o seu estado de acordo com :

$$x_i^{(m0)} = \text{sgn}[u_i^{(m0)}] \qquad (4.4)$$

Quando a memória é endereçada por um vetor binário que não é um dos vetores armazenados, a multiplicação vetor-matriz mais a função de ativação sgn[.] produzem um vetor de saída binário que, de forma geral, é uma aproximação da palavra armazenada com a menor distância de Hamming do vetor de entrada.

Se o vetor de saída é realimentado e utilizado como entrada à memória, a nova saída geralmente é um vetor que representa uma versão corrigida mais precisa do vetor armazenado. Se o vetor de entrada é considerado uma versão ruidosa do vetor de memória original, é possível visualizar a operação da rede como uma forma de correção de erro. O reconhecimento de um vetor de entrada, que corresponde a um dos vetores de estado da memória (ou próximo a um deles no sentido de Hamming), é considerado o estado estável da rede. Na Fig. 4.2 é representado um esquema da matriz sináptica **W**, equivalente à memória do sistema, F_h representa a função de transferência de cada elemento de processamento e o vetor **x** é o estado da rede no instante t. x_i' representa o estado do neurônio i, no instante t+1.

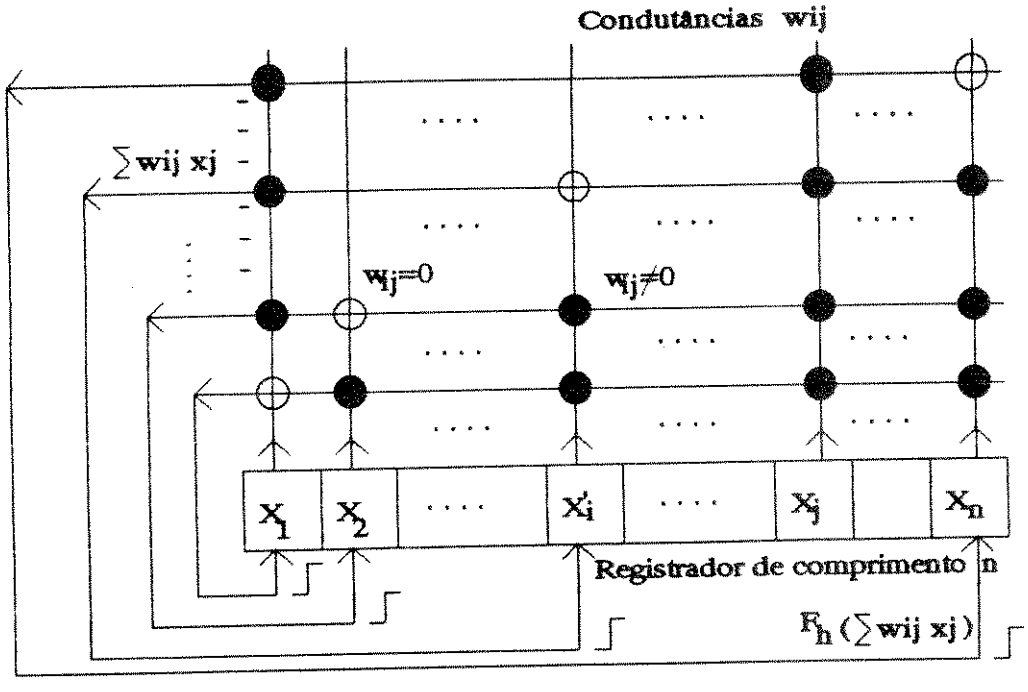

Fig. 4.2. Esquema da matriz sináptica.

Na Fig. 4.3 é esquematizada a operação de escrita ou inserção de um padrão na memória e a operação de leitura da rede. Na leitura pode-se notar que a estimativa $u_i^{(m0)}$ pode ser visualizada como uma operação de projeção de W_{ij} [50]. Os M vetores armazenados na memória também são conhecidos como *memórias fundamentais* [41], onde tais memórias fundamentais são consideradas como **atratores** do espaço de estados da rede neural, isto é, tais pontos exercem uma região de influência ao redor deles, pelo que a inicialização do sistema num ponto próximo a um destes atratores equivale a mapear tal estado inicial na memória, através das iterações sucessivas da rede.

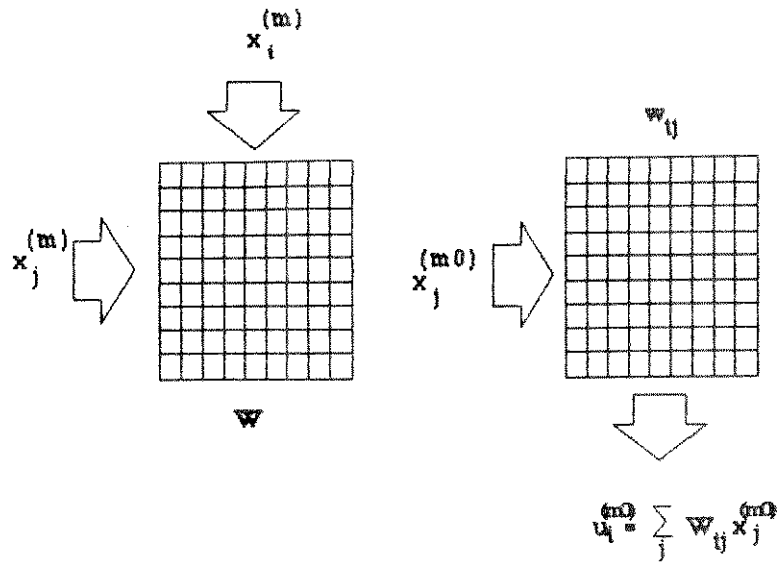

Fig. 4.3. Esquema da operação de escrita e leitura da memória.

4.4. Comportamento Dinâmico.

A questão da estabilidade da rede é direcionada especificamente como a estabilidade do sistema no sentido de Liapunov [48], e o problema consiste em determinar se a partir de um estado inicial próximo a uma memória ou estado fundamental, o sistema evolui na direção de um ponto de equilíbrio local, ou se ele é deslocado de tal ponto devido a uma perturbação. Hopfield descreveu em [17] o comportamento dinâmico da rede em termos de uma função de energia E, do tipo de uma função de Liapunov, a que caracteriza, mediante uma quantidade escalar, o comportamento coletivo do sistema no espaço discreto n-dimensional constituído por 2^n estados [41]. A função E é expressa por:

$$E = -1/2 \sum_{i,j} W_{ij} x_i x_j - \sum_j I_j x_j \qquad (4.5)$$

Hopfield demonstrou em [28] que (4.5) é uma função monotonicamente decrescente, e que a rede converge a um dos estados armazenados com uma alta probabilidade. Tal convergência é conseguida ainda quando as células atualizam os eus estados aleatoriamente, em forma assíncrona, e assim que a atualização é feita na mesma taxa média de atividade para cada elemento de processamento.

Como já foi mencionado na secção 4.2., se se considerar a localização de um ponto estável particular no espaço de estados da memória, então os estados próximos àquele ponto estável particular, possuem informação parcial relativa àquela memória fundamental. Segundo as considerações para uma estabilidade global, deve ser atingido pelo menos um estado estável. Sob o ponto de vista da estabilidade local, discute-se o aspecto da recuperação completa da informação contida num padrão, ao inicializar a

rede num ponto inicial contendo informação parcial acerca de um dos padrões armazenados na memória (cf. secção 4.2). O algoritmo que permite armazenar um conjunto de estados estáveis $\mathbf{x}^m$ (onde m=1,...,M) é obtido a partir do procedimento descrito na secção 4.3, para o modelo com comportamento discreto. No modelo denominado assíncrono, cada neurônio reajusta o seu estado aleatoriamente, com a mesma taxa média no tempo.

Na secção seguinte será descrita a operação para o modelo binário conhecido como síncrono [41].

4.5. Condições de estabilidade.

A estabilidade do modelo proposto por Hopfield está ligada à questão da recuperabilidade das chamadas memórias fundamentais. Num primeiro momento, estabelece-se que tais memórias devem, pelo menos, representar pontos fixos no mapeamento $\mathbf{x} \rightarrow \mathbf{x}'=\text{sgn}(\mathbf{u})$. Ademais, deve-se destacar que tais pontos fixos possuem o mesmo sentido, mesmo em se tratando do caso de operação síncrona quanto da assíncrona, onde se entende como síncrona àquela operação onde os componentes do vetor de estado corrente $\mathbf{x}$ são avaliados simultaneamente, de acordo com as regras descritas na secção 4.3.

Na proposta de McEliece e outros [41], foi colocado que existe um domínio ou uma base de atração ao redor de cada memória fundamental, para onde há uma alta probabilidade de os componentes x_i do vetor de estado inicial (ou prova) serem atraídos. Na memória associativa de Hopfield, com $\mathbf{m}$ memórias fundamentais representadas por vetores de estado, onde todos os componentes dos vetores assumem com a mesma probabilidade de 1/2 o valor ± 1, um dos mais surpreendentes resultados obtidos por McEliece é que, ao se inicializar a rede com uma prova de n-componentes a uma distância de Hamming máxima de ρn de uma memória fundamental ($0 \leq \rho \leq 1/2$), os pontos estáveis são localizados essencialmente nas proximidades do perímetro de uma esfera de raio εn, ainda quando o ponto inicial encontra-se dentro da esfera. Inclusive ao se inicializar a operação no seu centro, equivalente à memória verdadeira, a rede converge para aqueles pontos (cf. Fig. 4.4). Tal resultado indica que são introduzidos erros na operação da rede ao se inicializar com a memória fundamental exata, embora McEliece considera que tais erros são desprezíveis [41]. A expresão $R2=\varepsilon n$ representa a fração errada permissível dos $\mathbf{n}$ componentes do vetor de estado, após ter alcançado o ponto de estabilidade.

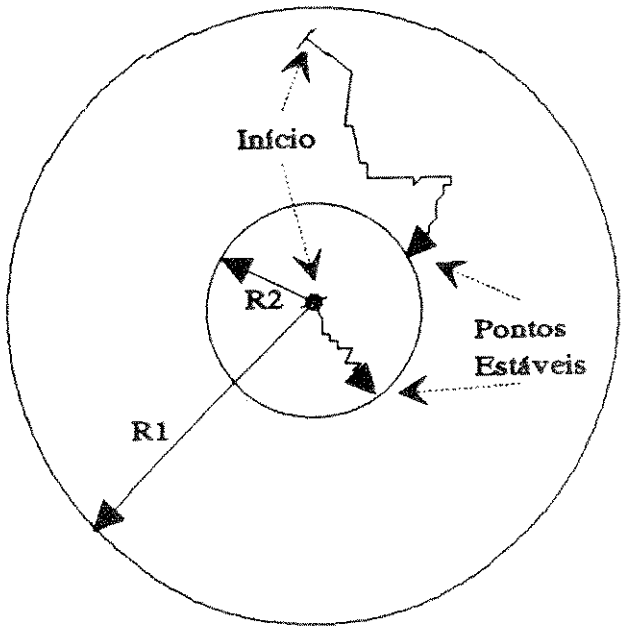

Fig.4.4. Pontos de estabilidade nas proximidades do raio de atração R1.

Na Fig. 4.4, o maior valor possível de R1=ρn representa o raio de atração de uma memória fundamental. Se o espaço de estados é representado por um conjunto de estados possíveis para os quais a rede pode convergir (por ex. o espaço de Hamming), ter-se-á m esferas disjuntas de raio <n/2 representando as memórias fundamentais, abrangendo uma pequena fração da probabilidade total, no caso de m pequenos. McEliece reporta no seu trabalho três posssibilidades principais de convergência, tanto para o caso da operação síncrona como assíncrona:

(i) A esfera de raio ρn pode ser atraída, diretamente (ou monotonicamente), ao centro de sua memória fundamental. Na versão assíncrona, uma transição de estado, representada pela mudança de um componente, é uma mudança na direção correta. A versão síncrona converge numa só iteração (Fig.4.5.a).

(ii) No caso assíncrono, um passo feito aleatoriamente representa um passo na direção correta, com uma probabilidade próxima a 1, mas não exatamente 1 (Fig.4.5.b). Na proximidade da memória fundamental, as mudanças consecutivas de estado são na direção correta. No caso síncrono, isto implica uma convergência em duas iterações.

(iii) Neste caso, os componentes do vetor de estado podem variar, produzindo a aproximação ou não do vetor corrente com o ponto fixo (estável) assim que a rede evolui no tempo através do espaço de estados, mas em média, é muito provável que o vetor esteja mais correto após a mudança, do que antes dela. Isto significa que, após um número finito de iterações, a rede converge a um ponto fixo, embora tal ponto

possa ser exatamente uma memória fundamental, como também um ponto de estabilidade próximo a uma delas a uma distância $\leq \varepsilon n$ (vide Figs. 4.4 e 4.5.c).

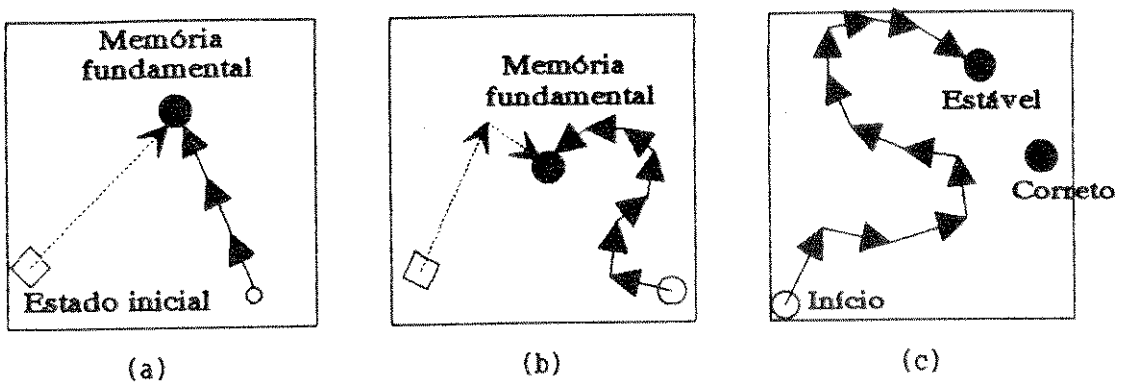

Fig. 4.5. Possibilidades de convergência do modelo de Hopfield
a)atração direta; --- : caso síncrono, ____ : caso assíncrono
b) atração na direção correta com probabilidade próxima a 1.
c) Convergência a um ponto fixo distinto da memória correta.

O anteriormente exposto indica que existem pontos de estabilidade aos quais a rede converge após um número finito de operações. Hopfield demonstrou em [17] que sempre é alcançada a estabilidade no modelo assíncrono, sempre que a matriz sináptica **W** seja simétrica e de diagonal principal 0 (o que implica $w_{ij}=w_{ji}$; $w_{ii}=0$), começando a operação da rede em qualquer ponto do espaço de estados. McEliece acrescenta em [41] que, se existe uma região de atração ao redor das memórias fundamentais, tanto no modelo síncrono quanto no assíncrono, então há uma alta probabilidade de existirem pontos de estabilidade, inclusive no modelo síncrono.

5. Simulação paralela do modelo de Hopfield
Uma abordagem

5.1. Introdução

Neste capítulo é tratado primordialmente o aspecto da simulação virtual do modelo binário de J.J. Hopfield operando em forma síncrona, i.e. com todos os seus elementos de processamento operando simultaneamente, de forma paralela. O interesse principal é utilizar esta rede como uma memória endereçada pelo conteúdo (CAM), visando a implementação da simulação do modelo num sistema de multi-processadores baseado no processador **Transputer T800** [23]. A topologia de interconexão considerada foi a de um *hipercubo* de dimensão 3 (2^3=8 elementos de processamento), embora os processos concorrentes associados à simulação das células elementares (equivalentes aos neurônios) tenham sido desenvolvidos com base num ambiente *mono-Transputer* (1 só processador Transputer T800), no qual construiram-se os processos concorrentes. Isto foi feito utilizando a linguagem de programação para processamento paralelo Occam-2 [22], rodando no ambiente TDS (*Transputer Development System*) versão D700d (1988). Tal ambiente, junto às características da linguagem Occam, possibilitam o desenvolvimento de programas concorrentes utilizando um só processador, permitindo desta forma emular o comportamento dos processos, como se estivessem numa rede de processadores Transputers. Neste capítulo será descrita uma abordagem para a simulação paralela de redes neurais, considerando particularmente o modelo binário de Hopfield implementado na arquitetura 3-cubo. Também descrever-se-á sucintamente o esquema de funcionamento dos processos Occam que constituem o simulador. Finalmente serão discutidos os algoritmos de comunicação dos processos concorrentes.

5.2. Metodologia proposta para a simulação de ANNs num multicomputador.

A metodologia para a simulação de ANNs utilizada neste trabalho baseia-se em modelos de mapeamento de redes neurais propostos em [27] por J. Ghosh e K. Hwang, onde são tratados modelos conexionistas de grande porte, simulados em multicomputadores de tamanho moderado, i.e. ANNs da ordem $N=O(10^6)$-(10^{10}) células de processamento mapeadas em sistemas com $M=O(10^2)$-(10^5) processadores. Embora este trabalho trata de um modelo de rede neural de 48 elementos de processamento, para ser simulado numa arquitetura composta por 8 processadores Transputers, os princípios propostos por Ghosh e Hwang são válidos também para sistemas de menor porte.

5.2.1. Características gerais dos processadores Transputers e Occam

O Transputer é um dispositivo VLSI que incorpora *on-chip*: um processador de 32 bits, uma unidade de ponto flutuante de 64 bits, uma memória local de 4 Kbytes de capacidade e 4 portas de ligação. Estes elementos permitem comunicar o dispositivo com outros 4 Transputers via canais bidirecionais, o que implica a possibilidade de comunicação simultânea através das 4 conexões *Full-duplex*. O Transputer foi originalmente projetado para ser programado em Occam, linguagem que é fundamentada num modelo de concorrência que deriva quase integralmente do conceito CSP (*Communicating Sequential Processes*) [52]. Occam tem sido utilizado tanto como uma linguagem para processamento paralelo, quanto como uma linguagem de simulação e descrição de *hardware* [53], sendo de fato empregada por INMOS no projeto do Transputer. Porém, Occam está estreitamente associada à arquitetura do Transputer, permitindo ademais, junto ao ambiente TDS, o desenvolvimento da programação de processos concorrentes, independentemente do número de processadores e de sua configuração particular, facilitando assim a programação paralela. A linguagem possui características que permitem expressar o paralelismo de forma simples fornecendo uma maneira natural e eficiente de simular o paralelismo existente em diferentes sistemas de *hardware*, e de forma especial na simulação e descrição do alto grau de paralelismo dos sistemas conexionistas.

5.2.2. Alguns Aspectos da Simulação Paralela PDES

A simulação PDES (*Parallel Discrete Event Simulation*) [26], algumas vezes referida como Simulação Distribuída, consiste na execução da simulação numa máquina de arquitetura paralela na qual são mapeados os processos que constituem o sistema. A simulação de eventos discretos é freqüentemente utilizada para estudar sistemas complexos, que são modelados como conjuntos de processos interagindo. Cada um desses processos representa um modelo de algum componente do sistema real, e as interações entre os processos no modelo (processos lógicos) simulam interações entre processos reais (processos físicos). A simulação é levada a cabo por um processo computacional que simula uma seqüência de eventos que ocorrem no sistema, atualizando o seu estado quando ele evolui através do tempo. A atividade de uma ANN pode ser considerada como um conjunto de eventos assíncronos onde não existe um relógio global. Pelo contrário, os eventos equivalentes à atividade de cada EP individual ocorrem a intervalos de tempo irregulares. Deste modo, o programa de simulação é decomposto num conjunto de processos sendo executado de forma

concorrente. Tal abordagem se ajusta perfeitamente ao modelo de programação de Occam, onde um programa é composto por vários processos concorrentes comunicando-se através de canais. No desenvolvimento do programa de simulação, as maiores dificuldades estão relacionadas com a solução dos problemas de *Deadlock*, *Starvation* e possíveis erros de causalidade [26]; portanto, é preciso manter certas restrições de seqüenciamento para assegurar uma computação correta. É de especial importância realizar um adequado mapeamento do modelo de N neurônios numa rede de M processadores (M < N), principalmente devido à necessidade de comunicações intensivas entre os elementos de processamento, causada pela alta conectividade do modelo neural virtual. Tal conectividade é fundamentalmente limitada pela capacidade de comunicação entre os processadores que conformam o sistema.

5.2.3. Mapeamento num sistema multicomputador

A *conectividade* de uma rede neural é definida como a relação do número real de conexões entre as células e o número de conexões se elas fossem completamente conectadas. Para abordar o problema do mapeamento, é fundamental distingüir o caso onde é possível particionar a rede neural em grupos de células, de tal modo que a conectividade dentro de um grupo seja consideravelmente maior que a conectividade da rede completa. Segundo a proposta de Ghosh & Hwang, uma ANN pode ser caracterizada segundo um modelo geral que particiona a rede neural virtual em regiões de **núcleos** disjuntos de tamanhos comparáveis. Um núcleo consiste de um grupo de células elementares (os neurônios) que são fortemente conectadas. Por sua vez, os núcleos podem ser agrupados em **regiões de influência**, o resto das células forma um conjunto chamado de **região remota**. Desta forma, a rede pode ir sendo hierarquicamente particionada em sub-redes distingüíveis pelas suas propriedades funcionais e pelo grau de conectividade entre elas.

Uma ANN pode ser implementada completamente em hardware, mediante circuitos VLSI [6], [9], [10], [54] e/ou dispositivos ópticos [50], [51] ou implementada virtualmente por *software*. Numa implementação virtual, o funcionamento da rede neural é simulado utilizando um número de processadores físicos menor que o número de neurônios, multiplexando no tempo várias células em cada processador físico. A simulação de N células num sistema de M processadores se considera virtual quando M<N. No mapeamento da rede de neurônios na rede de processadores, é preciso definir a topologia de interconexão dos processadores físicos, considerando a carga de tráfego nas linhas de comunicação entre os EP, balanceamento da carga computacional e restrições da memória local, já que a atividade de todas as células de um grupo é

simulada num só processador. O esquema proposto em [27] para um mapeamento generalizado é apresentado na Fig. 5.1.

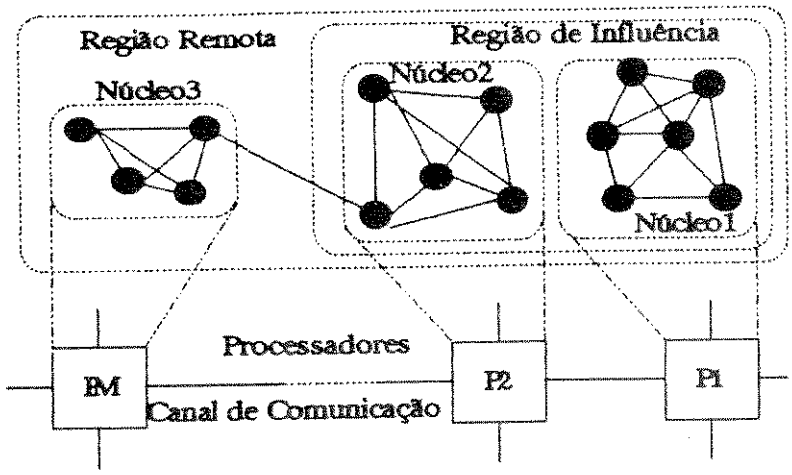

Fig. 5.1. Mapeamento de uma ANN num sistema de M processadores.

Numa rede neural completamente conectada, a tarefa do mapeamento é facilitada, e os conjuntos de células são mapeados nos processadores considerando somente a formação de núcleos de tamanhos comparáveis, a fim de balancear a carga de trabalho em cada processador.

5.2.4. Descrição do funcionamento das células e esquematização do simulador.

Uma célula neural genérica similar à representada na Fig 1.1.a recebe na sua entrada sinais correspondentes às saídas das células vizinhas, onde numa rede completamente conectada, cada célula recebe os sinais de todas as outras células da rede. Tais sinais são modulados pelos pesos de conexão correspondente à ligação entre as células. No instante de tempo t, a célula i está num estado interno $p_i(t)$, o que no modelo de Hopfield assume valores inteiros representando o seu potencial. A saída da célula é denotada por $x_i(t)$ e representa o seu nível de ativação, ±1 no modelo binário bipolar de Hopfield, o que pode ser representado por apenas 1 byte.

Cada célula atualiza o seu estado continuamente, assim que ele dispõe de todos os sinais de entrada das células às quais é conectada, transmitindo o seu novo estado corrente às mesmas células. A atualização é feita de acordo com as equações descritas na secção 4.3., considerando uma função de transferência f_i e os pesos na coluna i correspondente à matriz sináptica da rede (eq. (4.4), do Capítulo anterior). As funções de transferência podem ser visualizadas na Fig1.1.b-c. Num sistema neural adaptativo, tal como no modelo de Kohonen, cada célula pode modificar seus pesos de conexão w_{ij} $(1 \leq i,j \leq N)$, alterando assim dinamicamente o seu comportamento funcional. No

caso de modelos com aprendizagem não supervisionado, as mudanças nos valores dos pesos de conexão $w_{ij}(t)$ são atualizadas localmente considerando algoritmos do tipo de minimização de erro entre a saída corrente $x_i(t)$ e uma saída desejada $x_i'(t)$, onde $w_{ij}(t)$ representa o peso de interconexão entre a célula i e a célula j, no instante t .

Quando uma rede de N células é particionada em M conjuntos disjuntos, onde cada conjunto é alocado a um processador, pode-se definir como grupo local de um processador, o conjunto de células mapeado nele, que é chamado de processador local. A computação da ANN envolve uma contínua atualização dos estados correntes das células, a intervalos de tempo regulares, e o estado geral de rede deve ser avaliado continuamente para detetar a convergência ou a estabilidade do sistema. Tal computação é feita numa seqüência de ciclos de iterações, onde a cada ciclo os processadores operam concorrentemente atualizando os estados das células correspondentes ao seu grupo local, recebendo e transmitindo mensagens aos outros processadores do sistema. Utilizando processos concorrentes Occam, tal operação é feita segundo o esquema descrito na Fig. 5.2, onde um processo chamado de Monitor controla a operação de N processos célula, inicializando a rede, recebendo as mensagens correspondentes às saídas das células (ou a um conjunto delas) e enviando um sinal de parada quando é detetado o estado de convergência.

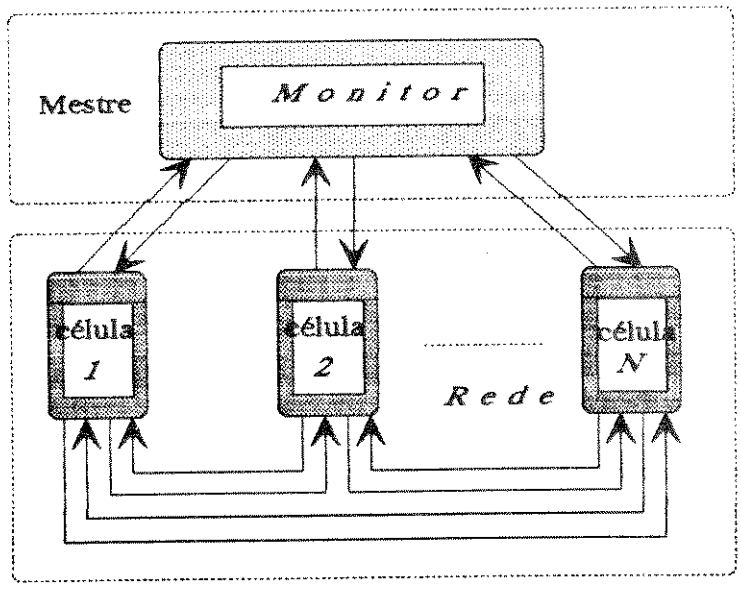

Fig.5.2. Arquitetura do simulador

Da Figura anterior pode se deduzir que a arquitetura do simulador proposto está estreitamente associada à arquitetura da rede neural, facilitando assim o mapeamento dos núcleos nos processadores, sujeito a futuras modificações.

5.2.5. Organização de uma Máquina MIMD de tipo *Message-Passing*

O sistema de computadores objetivado para a simulação é uma máquina de arquitetura MIMD [25] composta por uma rede de Transputers, organizada como um sistema de tipo *message-passing* [16], com os processadores distribuídos e interligados através de uma rede de interconexão (no caso, uma *crossbar-switch* programável). Isto permite configurar distintas topologias, tais como: *pipe-line*, árvore, *Mesh* com e sem *wrap-around* e hipercubo [55]. Neste trabalho, foi escolhido um sistema configurado como um hipercubo de dimensão d=3, o que equivale a 2^d=8 processadores mais um processador *Master* (MT) interligando a rede com o *Host*, destinado a fazer as operações de Entrada/Saída (vide Figs. 5.3 e 5.4).

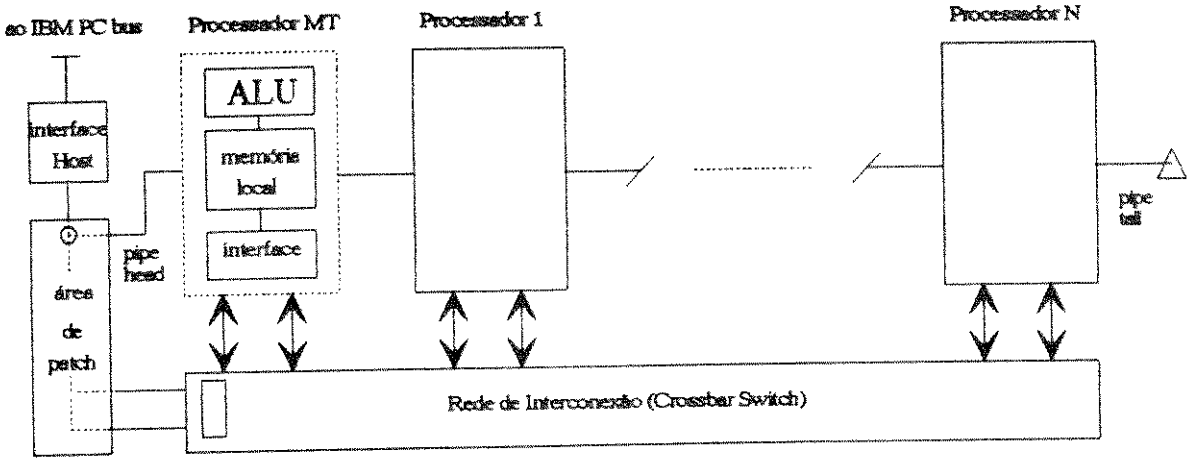

Fig.5.3. Arquitetura de um sistema multicomputador do tipo *Message-Passing*

Topologia Hipercúbica.

Um computador hipercubo d-dimensional é um multiprocessador de memória distribuída, composto por 2^d elementos de processamento individuais, chamados de nós, ligados numa rede de dimensão d (vide Fig. 5.4).

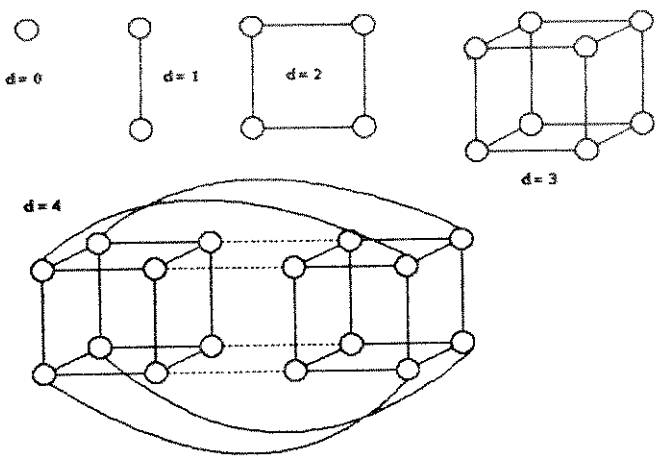

Fig.5.4. Topologias hipercúbicas para pequenas dimensões d.

Cada nó do hipercubo recebe um endereço ou número de identificação de d-bits (aqui chamado de n.id) e dois nós são ligados se e somente se os seus n.id diferem em exatamente uma posição de bit, a distância entre dois nós A e B, com n.ids:

$A = a_1 \ldots a_i\, a_{i+1}\, a_n$ e $B = b_1 \ldots b_i\, b_{i+1}\, b_n$

A distância entre os nós representa o comprimento do caminho mais curto, em número de ligações, e é igual à distância de Hamming dos seus endereços binários definida pela expresão :

$$H(A,B) = \sum_{i=1}^{n} h(a_i,b_i) \qquad (5.1)$$

onde $h(a_i,b_i) = 0$ se $a_i=b_i$

1 caso contrário

Dois nós A e B num hipercubo são chamados adjacentes ou vizinhos, se eles compartilham uma ligação.

A razão da utilização da arquitetura hipercúbica fundamenta-se nas suas atraentes propriedades topológicas, tal como diâmetro de comunicação logarítmico, regularidade e simetria [56]; e em recentes estudos baseados em simulação por computador onde se tem conferido o bom desempenho da arquitetura, quando se tem uma carga computacional balanceada [57].

5.2.6. Paralelismo com processos Occam.

5.2.6.1.Processos Occam

Na linguagem Occam, o elemento de construção principal é o processo. Um processo pode-se comunicar com outro processo através de canais físicos (*links*), quando ambos os processos são alocados em diferentes processadores, ou através de canais lógicos representados por uma palavra na memória, quando os processos são alocados no mesmo processador. A seguir, representa-se um conjunto de processos Occam conectados por canais virtuais (ou lógicos), que descrevem a implementação de uma ANN de n+1 neurônios (Fig. 5.5.). Os processos **cel** representam a atividade de uma célula neural e cada célula é comunicada a um processo consumidor **cons.**, que distribui a saída da célula nos canais de comunicação da rede.

5.2.6.2.Canais Occam

Os canais utilizados no exemplo da Fig 5.5. permitem a comunicação entre os processos concorrentes internos à rede (**cel** e **cons.**) utilizando um arranjo de canais internos **c.rede**. A rede comunica-se com o exterior através de 3 vetores de canais:

i) **cel.in.T.V**, onde são recebidos os vetores iniciais correspondentes à matriz sináptica e a o vetor de inicialização da operação;

ii) o canal **halt** recebe um sinal de um processo externo destinado a monitorar o estado geral da rede em cada ciclo de iteração. Ao conseguir a estabilidade o sinal recebido em ***halt*** pára a execução do sistema;

iii) a saida das células é enviada aos canais internos *c.rede* e ao canal **cel.out** pelo processo consumidor, a fim de ser processada por um processo externo Monitor.

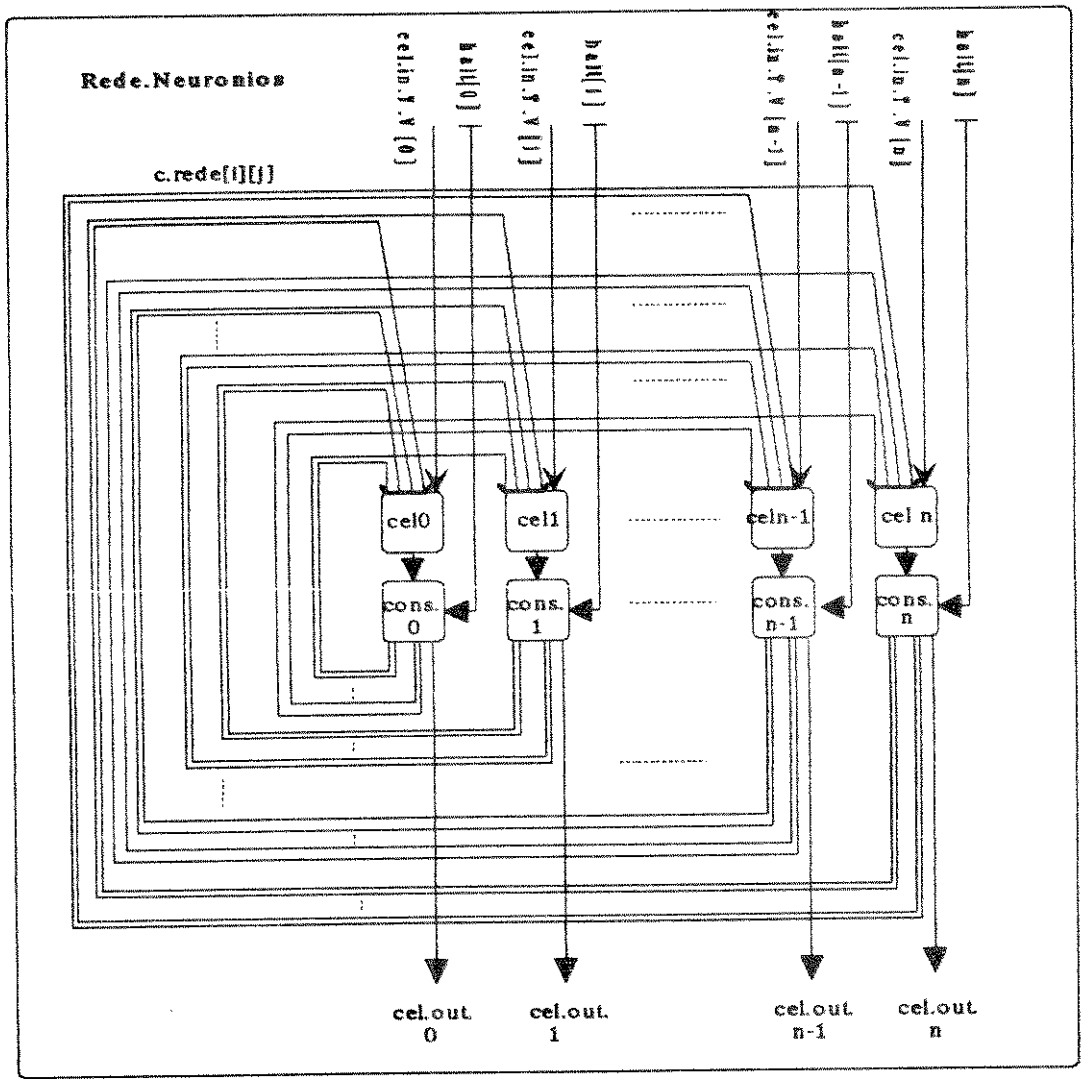

Fig. 5.5. Conjunto de processos concorrentes Occam representando uma ANN de n+1 neurônios.

O código fonte em Occam equivalente à Fig. anterior é:

```
[n]CHAN OF Protocol c.aux:
PAR i=0 FOR N
  PAR
    cel(cel.in.V.T[i],c.rede[i],c.aux[i])
    consumidor(c.aux[i],c.rede[i],cel.out[i],halt[i])
```

A construção `PAR i=0 FOR N` permite a criação de N processos concorrentes. A comunicação entre os processos `cel` e `consumidor`, em execução paralela, é feita através de um vetor de N canais chamado de `c.aux`. Um protocolo `Protocol` permite definir o tipo dos dados a serem transmitidos no canal.

5.2.7. Alocação de processos Occam em diferentes processadores.

Os processos Occam podem ser facilmente atribuídos a distintos processadores, implementando a comunicação entre os processos mediante as conexões físicas entre dois processadores. Um exemplo de alocação é representado na Fig. 5.6.

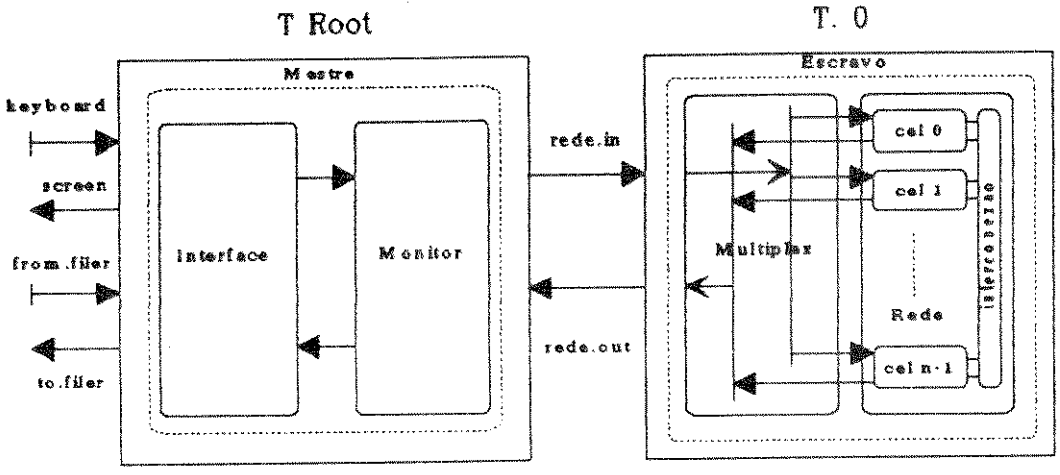

Fig. 5.6. Alocação de processos concorrentes Occam em dois processadores.

Na Figura anterior, um processo **Mestre** e um processo **Escravo** são alocados em dois processadores Transputers: **T.Root** e **T.0**, para execução paralela. A comunicação entre os processos é implementada pelos canais `rede.in` e `rede.out`, os quais são atribuídos a um canal físico *full-duplex*, correspondente a um *link* físico do Transputer. As operações de E/S são realizadas pelo processo *Mestre* no *T.Root*, sendo que o *Mestre* é constituído por dois processos concorrentes: *Interface* e *Monitor*. Os canais `keyboard`, `screen`, `from.filer` e `to.filer` realizam a transferência de dados com o sistema de arquivos e com os outros dispositivos do *Host*.

No processador T.0, o processo **Escravo** é formado pelos processos concorrentes *Multiplex* e *Rede* (proc. *Rede* semelhante ao processo da Fig. 5.5). Multiplex recebe as informações desde o Monitor e as distribui nos canais de entrada da Rede, ao mesmo tempo que recebe as saídas das células para enviá-las ao Monitor.

O código fonte em Occam, correspondente aos processos descritos na Fig.5.6., pode ser escrito como:

```
CHAN OF Protocol rede.out,rede.in:
PLACE PAR
  PROCESSOR 0 T8          -- Transputer T.Root
    PLACE rede.out AT 6:     -- link 2 in
    PLACE rede.in  AT 2:     -- link 2 out
    Mestre(keyboard,screen,from.filer,to.filer,rede.out,rede.in)
  PROCESSOR 1 T8          -- Transputer T.0
    PLACE rede.out AT 1:     -- link 1 out
    PLACE rede.in  AT 5:     -- link 1 in
    Escravo(rede.out,rede.in)
```

No trecho de código descrito acima, o Transputer T.Root utiliza o seu canal físico **link2** para comunicar-se com o Transputer T.0, este último comunica-se pelo **link1**. Cada Transputer dispõe de 4 canais físicos ou **links**, (link0...link3), onde cada canal é endereçado por um valor inteiro entre 0 e 7.

Uma outra opção é pensar o processo *Rede* da Fig.5.6. como um conjunto de processos a serem executados numa configuração particular, como por ex. numa rede de processadores configurado como um hipercubo. Os diferentes processos podem ainda serem desenvolvidos num ambiente constituído por 2 processadores ou até num só processador, aproveitando a característica de Occam que permite a decomposição hierárquica de processos. A modificação do sistema anterior é representada na Fig. 5.7.

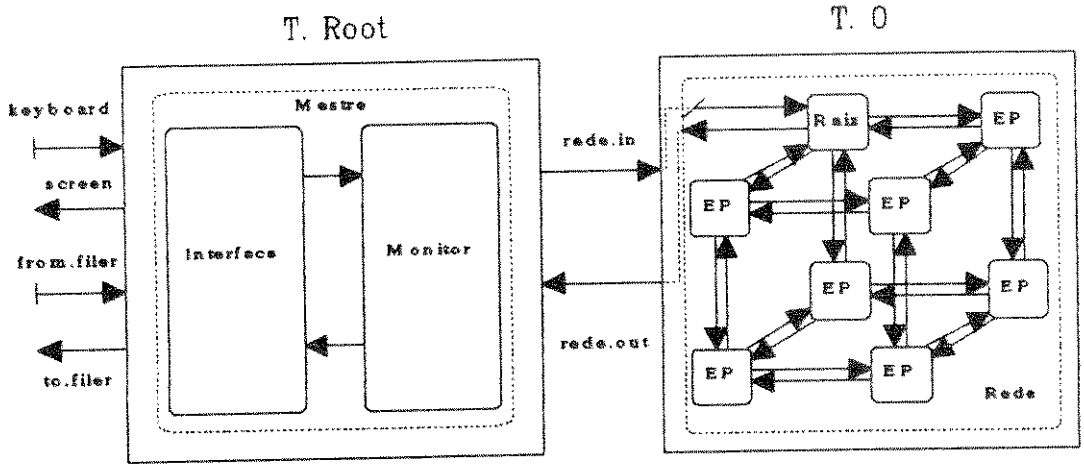

Fig. 5.7. Alocação de processos Occam em dois processadores.
O Processo Rede, no T.0, simula uma configuração hipercúbica.

5.3.Mapeamento e Implementação no 3-cubo.

Nesta secção é tratada a implementação de processos Occam no hipercubo de dimensão 3. Os processos a serem executados em cada nó foram decompostos hierarquicamente em processos concorrentes rodando no processador local e comunicando-se por meio de canais virtuais. Os programas Occam foram desenvolvidos num ambiente Mono-Transputer.

O ambiente objeto para a execução paralela das simulações inclui processadores IMS T800, hospedados numa placa INMOS B008 e rodando em TDS. A configuração da rede de processadores é feita através de uma *Crossbar-Switch* IMS-C004, programável pelo processador IMS-T212, através do uso do *Software* MMS2 (de INMOS). O cubo é conectado ao exterior através de um processador Mestre (MT).

Na Fig. 5.8 o número associado a cada processador é a identificação dada pelo *Software* MMS2. A linha contínua representa uma conexão de *hardware* já definida pelo fabricante. Os pontos *EdgeLink*, possibilitam a conexão a uma área de *patching*. O número em cada linha de comunicação identifica o *link* de cada processador.

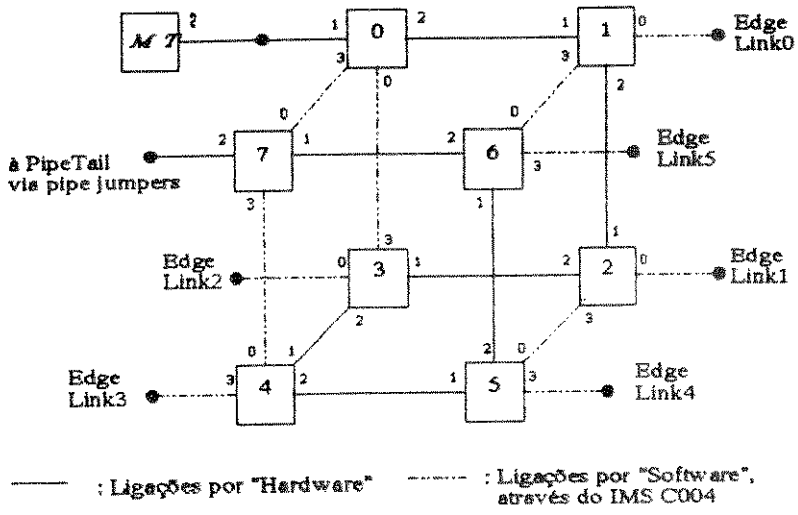

Fig. 5.8. Ambiente objeto para utilização do Simulador.

5.3.1. Mapeando Processos Occam no 3-cubo.

Considerando o ambiente da Fig. 5.8, um conjunto de processos Occam é mapeado no cubo para serem executados em paralelo em cada nó, os processos são descritos a seguir e o mapeamento é representado pela Fig. 5.9. O número acima de cada processador indica o n.id, número de identificação de cada nó (ref. à secção 5.2.5). Tal como nos casos anteriores, existe um processo *Monitor* (rodando dentro do Mestre, M), com as mesmas funções já explicadas, mais oito processos similares, sendo executados em paralelo, cada um destinado a simular um núcleo de células. Os processos EP comunicam-se com 3 nós vizinhos simultaneamente, o processo R (Raiz) além de se comunicar com os 3 vizinhos deve trocar informações com o Monitor.

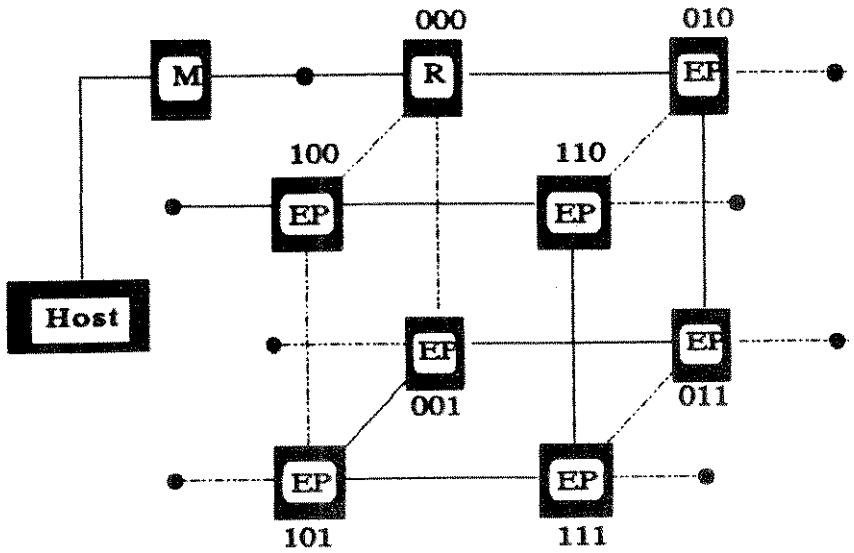

Fig. 5.9. Mapeamento dos processos Occam no 3-cubo.

A decomposição hierárquica dos processos na Fig.5.9 é representada na Fig.5.10. Neste esquema, cada um dos processos EP se comunica utilizando um total de 2*d canais (onde d é a dimensão do cubo). O processo Raiz utiliza 2*(d+1) canais, já que deve estabelecer comunicações com o Monitor, que por sua vez está conectado com o Host. Os algoritmos de comunicação dos nós são implementados pelos procesos *Router.R* e *Router.EP* e serão explicados na secção 5.4., a *Rede de interconexão* no *Nucleo*, representa o conjunto de canais de comunicação internos descritos em 5.2.6.

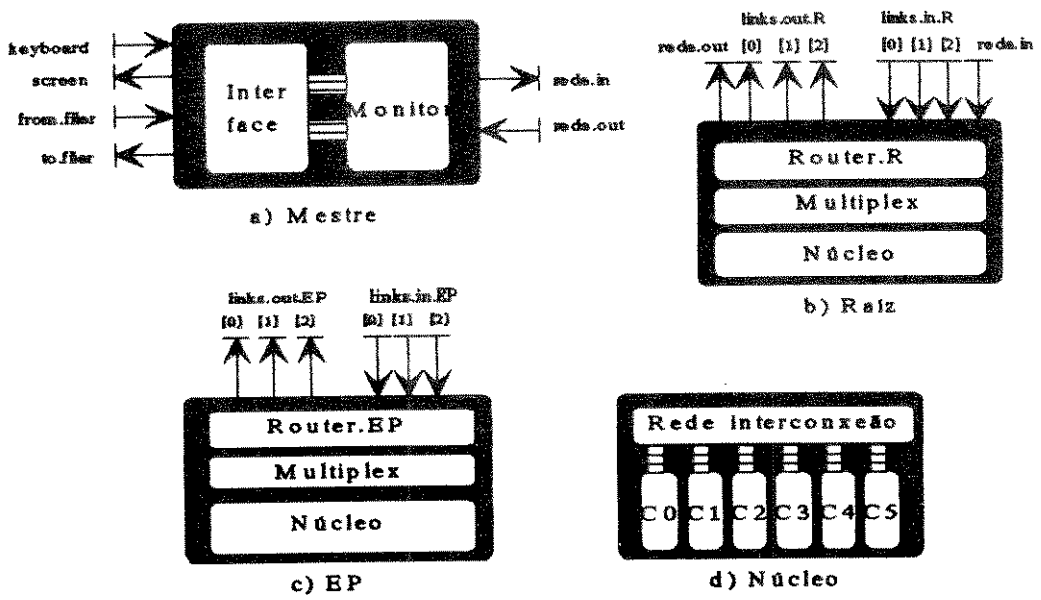

Fig. 5.10. Esquema geral dos processos mapeados em cada nó do 3-cubo.
O processo Raiz (R) comunica-se com o Monitor para E/S dos dados da rede.

O código fonte em Occam, correspondente à operação concorrente dos processos da Fig.5.10., pode ser escrito como:

```
VAL INT d IS 3:     -- Dimensao do Hipercubo
VAL INT M IS 8:     -- Numero de Processadores
[d+1]CHAN OF Protocol links.out.R, links.in.R:  -- canais em R
[d]CHAN OF Protocol links.out.EP,links.in.EP:   -- canais em EP
CHAN OF Protocol rede.out,rede.in:              -- canais ao Monitor
PAR
  Mestre(keyboard,screen,from.filer,to.filer,rede.in,rede.out)
  PAR
    Escravo.R(rede.out,rede.in,links.out.R,links.in.R)
    PAR i=1 FOR M-1
      Escravo.EP(links.out.EP,links.in.EP)
```

Os processos Escravo.R e Escravo.EP representam os processos R (Raiz) e EP da Fig.5.9, os canais keyboard, screen, from.filer, to.filer permitem a comunicação com o Host.

A comunicação entre os nós é realizada através de canais físicos, nos quais são alocados os vetores de canais lógicos links.out.R, links.in.R, links.out.EP, links.in.EP, dos processos R e EP, respectivamente. Na operação, *Multiplex* recebe as saídas do processo *Núcleo,* para assim formar um pacote que é transmitido pelo *Router* (EP e R) a todos os nós.

5.3.2. Operação do processo Núcleo

A principal operação do simulador consiste na execução concorrente de um número de células, agrupadas dentro de um processo chamado de Núcleo, que é executado simultaneamente em todos os processadores. As saídas do Núcleo são conectadas a um processo *Multiplex* que forma um pacote entregue ao *Router* . O pacote é distribuído na forma de uma mensagem a toda a rede, através da execução de um algoritmo de BROADCAST (a ser descrito nas próximas secções), simultaneamente cada nó recebe mensagens de todos os òutros nós, definindo desta forma uma operação de MULTI-BROADCAST em toda a rede. Na atual implementação, um núcleo é composto por 6 células, o que define uma rede neural completa de 48 elementos. O processo *Núcleo* recebe desde o *Router* o sinal de parada que indica que o estado de convergência do sistema já foi conseguido, pelo que a simulação é detida.

O código fonte Occam equivalente ao processo *Núcleo* foi descrito brevemente na secção 5.2.6. (caso generalizado). A recepção de um sinal de parada pode ser incluída através da primitiva ALT, que é um processo que usa os chamados ***comandos guardados*** definidos no conceito de **Processos Sequenciais Comunicantes** (CSP) [52]. A guarda do comando é um processo de leitura de um canal do seguinte tipo:

```
INT x:
CHAN OF INT canal:
SEQ
  canal ? x
  ...
```

No processo *consumidor*, na Fig. 5.11., um processo de leitura contínua de canais, incluindo comandos guardados, pode ser escrito como:

```
INT any,Y:
BOOL sinal:
CHAN OF INT cel.Halt,cel.out:
SEQ
  sinal:=TRUE
  WHILE sinal
    ALT
      cel.Halt ? any
        sinal:=FALSE
      cel.out ? Y
        ... continue
```

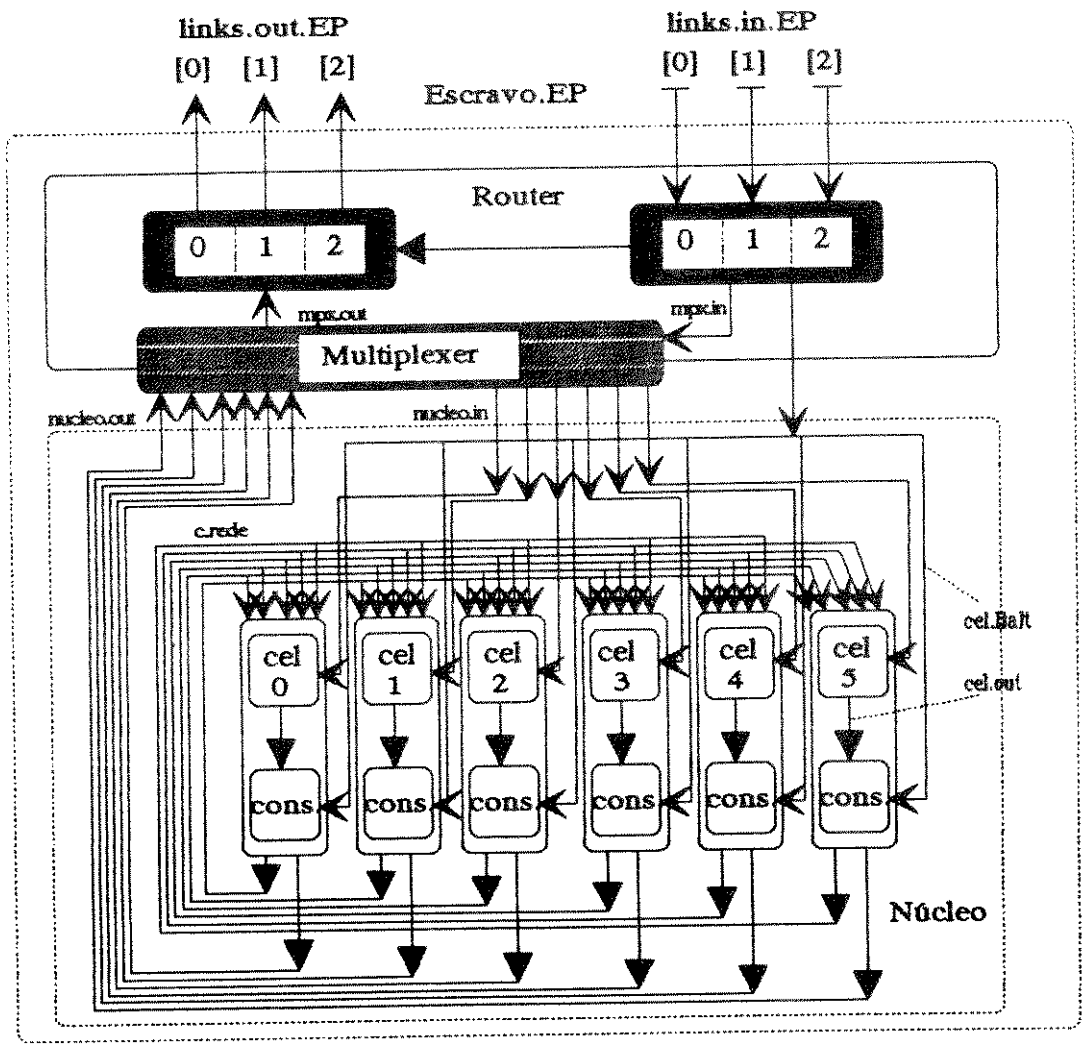

Fig. 5.11. Representação esquemática do processo Escravo.EP e a sua decomposição hierárquica.

No diagrama da Figura anterior, é representada a decomposição do processo *Escravo.EP* em três processos concorrentes: Router, Multiplexer e Núcleo. O processo Núcleo é, por sua vez, decomposto hierarquicamente em 6 processos, onde cada um representa a operação de um neurônio. Cada neurônio é formado por uma célula e um processo consumidor. As comunicações entre todos os processos internos são feitas através de canais virtuais implementados na memória de cada processador.

O código fonte, em Occam, correspondente à operação do processo Escravo.EP e esquematizado na Fig. 5.11, pode ser escrito da seguinte forma:

```
VAL INT d IS 3:      -- dimensao do hipercubo
PROC Escravo.EP([d]CHAN OF Protocol links.out.EP, links.in.EP)
   ... declaracoes de ctes & variaveis
  ...  definicoes das PROCedures
  SEQ
    ...Inicializacao
    CHAN OF INT c.Halt:                  -- canal sinal Halt
    CHAN OF Protocol mpx.out,mpx.in:  -- canais ao multiplex
    PAR
      Router(links.out.EP,links.in.EP,mpx.out,mpx.in,c.Halt)
      Multiplex(nucleo.out,nucleo.in,mpx.out,mpx.in)
      Nucleo(nucleo.out,nucleo.in,c.Halt)
```

5.4. Algoritmos de Comunicação no Hipercubo.

No simulador, as operações de comunicação entre os nós do cubo de dimensão d=3 baseiam-se, principalmente, em algoritmos de comunicação propostos por Q. Stout & B. Wagar [24]. Já que o cubo precisa realizar transferência de informações com um computador *Host* (hospedeiro), o nó identificado com o n.id=000 se transforma num nó especial ao ter que realizar simultaneamente a tarefa correspondente a todos os nós e as operações de E/S com um processador interface (MT na Fig.5.8). Foram implementadas as seguintes operações:

i)BROADCAST: transmissão de uma mesma mensagem desde um nó (por ex. 000) a todos os outros nós.

ii)MULTI-BROADCAST: conhecida também como *Gossiping* [58], consiste na operação de transmissão simultânea (BROADCAST) de todos os nós.

iii)O.C.SEND: é a transferência de uma mensagem entre o nó e outro situado na esquina oposta do cubo (ex. 000->111, para d=3). A operação pode ser realizada dividindo o d-cubo em sub-cubos (de dimensao <d), contidos no cubo principal.

iv)DISTRIBUTE: Operação conhecida também como *scattering* [59] ou *comunicação personalizada* . Nela, um nó envia uma mensagem diferente a cada um dos nós restantes do sistema.

5.4.1. Definições.

Para explicar o esquema de roteamento dos algoritmos serão definidas duas operações chamadas de *Flipping* e *Left Rotation*, que envolvem os números de identificação dos nós (n.id). As funções são denotadas pelos símbolos seguintes:
$\oplus_k(x)$ - *Flipping* Denota o n.id formado fazendo uma operação bit-a-bit OR-Exclusiva entre um nó identificado por x e 2^k.
$\backsim_i(x)$ - *Left Rotation* Denota um deslocamento à esquerda com rotação (*circular-shift*) em i bits do número x representado por d-bits.

Assume-se que a transmissão de uma mensagem de comprimento m, desde qualquer nó a um vizinho, leva um tempo $\tau m + \beta$, onde
τ é a taxa de transferência elementar,por ex. o tempo para transmitir 1 byte.
β é o tempo de partida (*start-up*) ou latência.

Nas Figuras descritas a seguir, assume-se que o nó identificado com o n.id=000, encontra-se conectado a um processador MT (mestre), de onde recebe as informações a serem distribuídas ao cubo.

5.4.2. Operação de distribuição de Mensagens.

Na inicialização da operação da rede, cada uma das células (u_i) deve receber a coluna i da matriz de pesos **W**. O processo Mestre deve dividir W em **M** sub-matrizes de **c** colunas cada uma, onde **M** é o número de processadores e **c** corresponde ao número de células por núcleo. O Mestre distribui M pacotes de mensagens, cada uma de tamanho c*N, onde N corresponde ao número total de neurônios da rede (no nosso caso M=8, c=6 e N=48 elementos). Antes de descrever o algoritmo de distribuição, será descrito o algoritmo *o.c.send* (*opposite corner send*), que é utilizado na distribuição.

5.4.2.1.Algoritmo O.C. SEND

O algoritmo é composto por d estágios, designados por 0,1,...,d-1. A mensagem é dividida em d pacotes: $P_0, P_1,...,P_{d-1}$. Em cada estágio k, o nó $\backsim_i(2^k-1)$ envia o pacote P_i ao nó $\oplus_k(\backsim_i(2^k-1))$, para cada i em {0,1,...,d-1}.

A divisão da mensagem em pacotes permite que o algoritmo leve um tempo de apenas $\tau m + d\beta$ (d a dimensão do cubo).

5.4.2.2.Algoritmo DISTRIBUTE

A *distribuição* é levada a cabo em d estágios, designados por 0,1,...,d-1. A cada um dos nós é enviada uma mensagem através de O.C.SENDs separados, aplicados a um sub-cubo contendo o nó 000 numa esquina e o nó destino na esquina oposta. No

algoritmo DISTRIBUTE original [24], os 2^d-1 O.C.SENDs separados são executados de forma concorrente, uma modificação é descrita em 6.2.

Para o caso de 3-cubo, na Fig.5.12 representa-se graficamente a operação do algoritmo O.C.SEND entre os nós opostos 000 e 111.

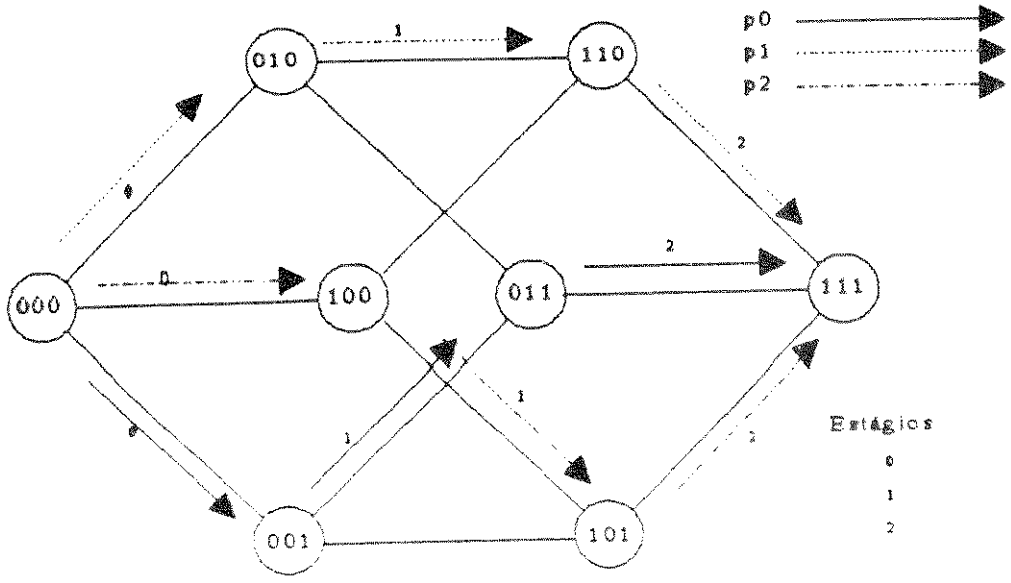

Fig. 5.12. Representação gráfica do algoritmo O.C.Send.

Na Figura anterior, os pacotes são representados por 3 setas diferentes, o número associado a cada uma delas indica o estágio em que se encontra o algoritmo.

A operação de O.C.SEND é necessária para a transmissão da cada sub-matriz de W desde o nó 000 a cada um dos outros nós. A operação de distribuição completa é levada a cabo numa seqüência de O.C.SENDs: dividindo o 3-cubo em 3 sub-cubos de tamanhos d=2 são feitas 3 sequências de transmissões O.C.SENDs desde 000 aos nós situados a uma distância de Hamming de 2. Nas últimas 3 operações o nó 000 transmite uma sub-matriz diferente a cada um dos seus nós vizinhos, situados a uma distância de Hamming de 1.

A seguir serão descritos os algoritmos considerados fundamentais para a comunicação da rede: dois tipos de BROADCAST e um algoritmo onde todos os nós participam simultaneamente, chamado de MULTI-BROADCAST.

5.4.3. Operações de *Broadcast* e *MultiBroadcast*

5.4.3.1.Algoritmo BROADCAST1

Na sua versão mais simples, a operação de BROADCAST é feita em d estágios, e as mensagens a serem transmitidas não são divididas em pacotes. Esta operação é considerada convenientemente eficiente para mensagens curtas [24], [58].

Operação: Existem d estágios designados por 0,1,...,d-1. No estágio k, os nós 0,1,...,2^k-1 enviam em forma concorrente a sua mensagem aos nós $\Phi_k(0)$, $\Phi_k(1)$, ..., $\Phi_k(2^k-1)$, respectivamente.

Na Figura seguinte, é representada em forma esquemática a operação deste algoritmo para o hipercubo de dimensão d. Da mesma forma que na Fig. 5.12, os números associados à mensagem, representada por uma seta, indicam o estágio em que se encontra o algoritmo.

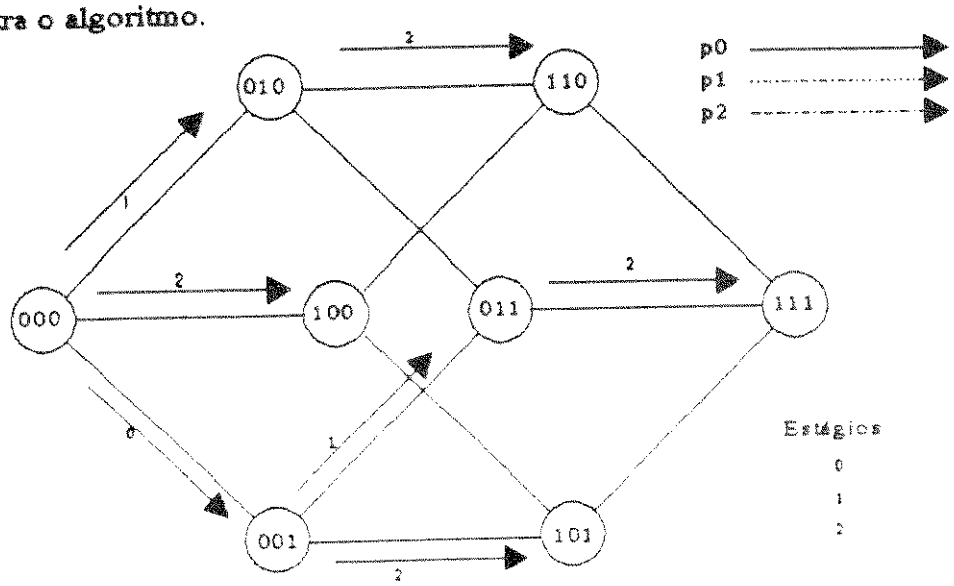

Fig. 5.13. Representação gráfica do algoritmo BROADCAST1.

5.4.3.2. Algoritmo BROADCAST2

Este algoritmo é considerado como uma simetrização do BROADCAST1 [24], i.e., a mensagem é dividida em d pacotes: p_0, p_1,...,p_{d-1}, onde cada um dos pacotes é de tamanho m/d e m é o tamanho da mensagem original. No estágio k, os nós $\leftarrow_i(0)$, $\leftarrow_i(1)$, ..., $\leftarrow_i(2^k-1)$ enviam o pacote p_i aos nós $\Phi_k(\leftarrow_i(0))$, $\Phi_k(\leftarrow_i(1))$, ..., $\Phi_k(\leftarrow_i(2^k-1))$, respectivamente, para cada i pertencente ao conjunto {0,1,...d-1}.

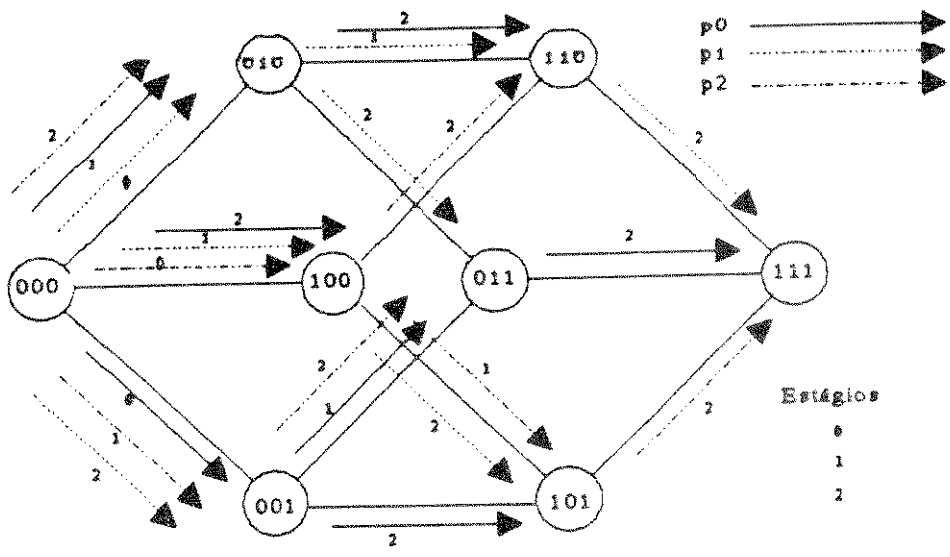

Fig. 5.14. Representação gráfica do algoritmo BROADCAST2.

A operação representada na Fig.5.14 é levada a cabo para transmitir o vetor de inicialização (v_{inic}) da rede neural, i.e., o padrão de entradas desconhecido que inicia o funcionamento do modelo de Hopfield. O nó 000 recebe v_{inic} desde o MT, e depois de dividi-lo em 3 pacotes é transmitido a cada um dos nós em forma concorrente.

Em cada um dos passos de BROADCAST2, todos os nós calculam o seu nó destino de acordo com as operações sobre as funções descritas no algoritmo. Na Figura anterior, os pacotes p_0, p_1, e p_2 são representados por setas e o número associado a cada uma delas indica o estágio k do algoritmo.

5.4.3.3.Algoritmo MULTI-BROADCAST

Nesta operação, todos os nós trabalham simultaneamante. O algoritmo baseia-se em tomar uma operação de *Broadcast* centrada no nó 000, e aplicá-la simultaneamente a todos os nós. Devido à simetria do hipercubo, isto é possível fazer convertendo o algoritmo escrito para o nó 000 em outro algoritmo idêntico, para cada um dos outros nós α pertencentes ao conjunto $\{1,2,\ldots,2^d-1\}$, através da operação de OR-EXclusivo entre cada nó referenciado no algoritmo original (centrado em 0) e o nó α.

A seguir, é descrito um algoritmo sem simetrização (não há divisão das mensagens em pacotes), onde cada nó opera de forma concorrente produzindo padrões de comunicação altamente simétricos.

Operação: Há d estágios designados como 0,1,...,d-1. O algoritmo BROADCAST1 é aplicado simultaneamente aos 2^d nós. No estágio k, cada nó compartilha o trabalho correspondente com os outros nós vizinhos.

Na Fig.5.15, cada um dos estágios é representado por um símbolo quadrado contendo um número k, onde k indica o estágio corrente do algoritmo. Uma seta, ou

um conjunto delas, representa a mensagem completa (um *pack*) entre nós vizinhos e o número associado a cada seta identifica o nó originário da mensagem. Em cada estágio k, cada um dos nós troca mensagens com o nó vizinho diferenciado no k-ésimo bit (onde k=0 representa o intercâmbio com o nó vizinho diferenciado no bit menos significativo).

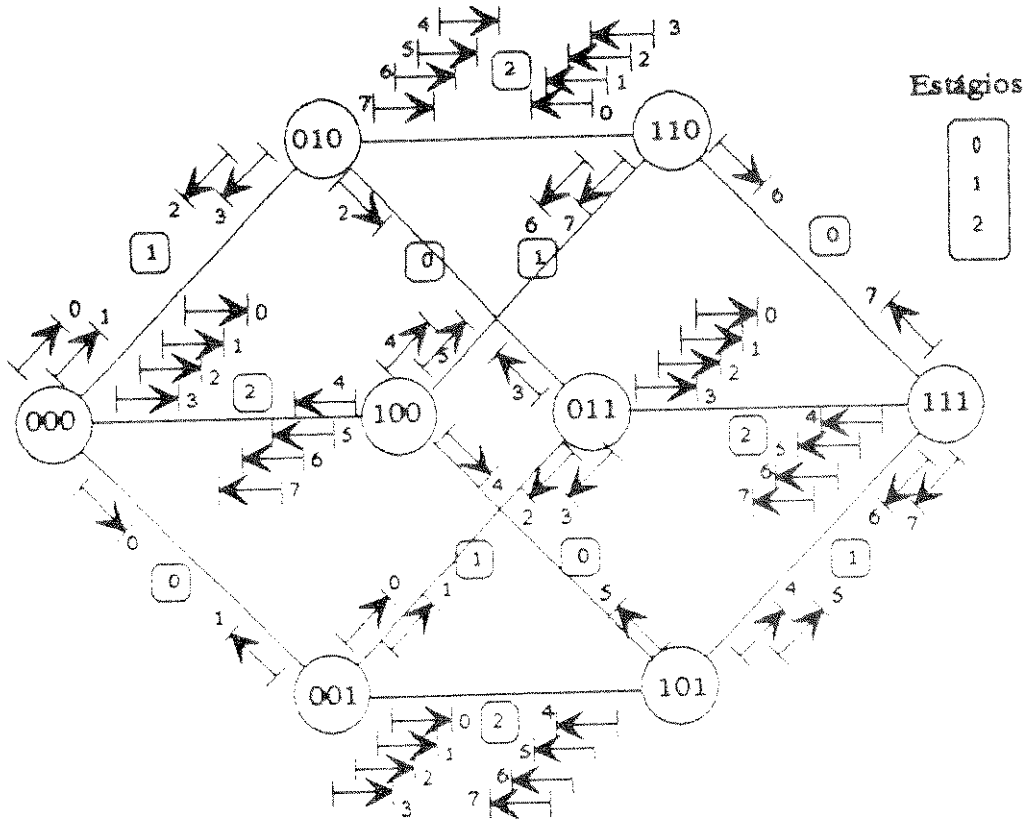

Fig. 5.15. Representação gráfica do algoritmo BROADCAST2.

No estágio k, o tamanho m das mensagens é de $p2^k$, considerando p o tamanho do pacote pertencente a um só nó, e que equivale à saída de um núcleo composto por 6 células. A operação de *Multi-Broadcast* é realizada continuamente até ser recebido, no nó 000, um sinal de parada que é transmitido como um BROADCAST1 a todos os nós, o que detem a execução da simulação.

No capítulo seguinte será feita uma análise correspondente aos tempos de execução de cada um dos algoritmos anteriormente descritos.

6. Análise do Desempenho

6.1. Introdução

Para analisar o desempenho do simulador, deu-se ênfase principalmente aos algoritmos de comunicação descritos no Capítulo 5, já que os *overheads*, devido aos tempos de comunicação, foram relevantes no desempenho global da simulação (ref. à Fig. 6.2). Serão estimados, primeiramente, os tempos de inicialização da rede ao executar os algoritmos DISTRIBUTE e BROADCAST2, onde são comunicadas as mensagens contendo as sub-matrizes de W (matriz sináptica da rede), correspondente a cada grupo de células (núcleo) alocado a um processador, e o vetor de início da operação, v_{init}, que é transmitido a todos os nós. Em segundo lugar será analisado o caso do MULTI-BROADCAST, onde todos os nós operam simultaneamente.

6.2. Análise para o Algoritmo DISTRIBUTE.

Na sua versão original [24], DISTRIBUTE é formado por 2^d-1 operações de O.C.SENDs concorrentes, utilizando *batching*, i.e., formando pacotes contendo mensagens distintas dirigidas a vários nós e enviados como um pacote por vez (vide Fig. 6.1). Este algoritmo foi modificado, já que a técnica de *batching* requer memória adicional, a fim de armazenar pacotes temporários que são passados aos outros nós, quando o nó corrente não é o destino.

Na modificação aqui proposta, o algoritmo é formado por uma seqüência de 2^d-1 operações de O.C.SENDs aplicados ao 3-cubo e aos dois tipos de sub-cubos contidos nele (dimensão d=2 e d=1). Obviamente, o algoritmo tem pior desempenho que o da versão original, embora permita aproveitar melhor a memória local disponível em cada processador. Num O.C.SEND, a transmissão de uma mensagem de comprimento m leva um tempo de: $\tau m + d\beta$. Para o algoritmo realizar 2^d-1 operações seqüenciais, a execução do algoritmo leva um tempo total de:

$$\sum_{k=0}^{2^d-2} (\tau m + d_k \beta) \qquad (6.1)$$

onde d_k, corresponde à dimensão do sub-cubo no estágio k, $d_0=3$.

Tempos de execução: assume-se que o envio de uma mensagem de comprimento m, leva um tempo de $\tau m + \beta$. O tamanho m da mensagem é constante e igual a 48*6 elementos (1152 bytes). Utilizando as medidas de desempenho em máquinas baseadas em Transputers [58]: β=1049,3 μs e τ=1,5 μs, obtêm-se os tempos de propagação de:

- 4858 μs, para k=0 (000→111),

- 3814,6 μs (para cada k=1,2,3; no sub-cubo de d=2) e

-2771.3 μs (para cada k=4,5,6; no sub-cubo de d=1).

O tempo total de DISTRIBUTE leva 24615,6 ≈ 25000 μs.

Na Fig.6.1, representam-se graficamente 3 operações O.C.SEND para sub-cubos de dimensão d=2, onde todas as mensagens são emitidas pelo nó 000.

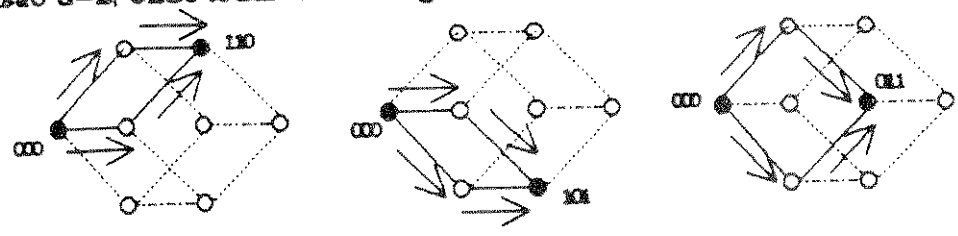

Fig.6.1. Representação gráfica de O.C.SEND para comunicações com sub-cubos de d=2.

6.3.Análise para o algoritmo BROADCAST2.

Na execução de BROADCAST2, uma mensagem de tamanho m, a ser comunicada a todos os nós desde 000, foi dividida em 3 pacotes, cada um de tamanho m/d. A mensagem é formada pelo vetor v_{inic} (ref. à secção 5.4.3.) de 48 elementos (192 bytes).

tempo de execução: O algoritmo leva um tempo total de $\tau m + d\beta$ [24], pelo que, considerando os valores para τ e β da secção anterior e o tamanho do pacote m/d, obtem-se um tempo de 3417,9≈3418 μs.

6.4. Inicialização da rede.

Com uma simples observação da Fig. 6.1. e com a análise de DISTRIBUTE, pode-se ver que a operação de realizar O.C.SEND concorrentes, utilizando *batching* requer que o nó 000 tenha a capacidade de armazenamento de pelo menos 3 mensagens diferentes para o caso de comunicação simultânea com os 3 sub-cubos da Fig. 6.1. A operação de mais 3 O.C.SENDs comunicando mensagens diferentes desde o nó 000 aos vizinhos e um O.C.SEND (no estágio k=0) dedicado a comunicar uma outra mensagem ao nó 111, aumenta consideravelmente os requerimentos de memória para cada processador, especialmente para o processador com n.id=000. Deve-se levar em conta que a matriz de pesos W deve ser transferida desde um processador MT (secção 5.3.) ao nó 000, para posteriormente ser distribuída à rede de processadores. A disponibilidade de módulos de processadores com pouca memória local reservada à área de dados levou a considerar a operação de DISTRIBUTE como 2^d-1 operações seqüenciais de O.C.SENDs, sacrificando a velocidade na inicialização da rede, mas tendo a opção de ter mais memória local disponível em cada processador.

tempo de execução: Com a soma dos valores obtidos em 6.2 e 6.3, obtem-se uma estimativa do tempo de comunicação total que leva a inicializar toda a rede com os vetores v_{inic} e as sub-matrizes de W, obtendo-se o valor de 28033,5≈28000 μs.

6.3. Análise para o algoritmo MULTI-BROADCAST.

Utilizando o algoritmo BROADCAST1 descrito na secção 5.4.3, estimar-se-á o tempo que leva para a rede toda realizar uma comunicação completa, onde todos os nós receberam a mensagem de todos os outros nós, permitindo desta forma que a rede neural complete um ciclo de iteração.

O algoritmo BROADCAST1, apesar da implementação relativamente simples e da necessidade de pouca memória para cada processador, sub-utiliza a capacidade de uso simultâneo de d canais por cada nó. Ao realizar 2^d operações de BROADCAST1 simultâneas na rede, implementando assim o MULTI-BROADCAST, no estágio k cada nó ocupa somente um canal de comunicação (em ambos os sentidos). O tamanho da mensagem é duplicado em cada um dos estágios.

tempo de execução: O tempo total que leva para realizar 2^d operações de BROADCAST simultaneamente é:

$$T = d\beta + \sum_{k=0}^{d-1} 2^k \tau m = d\beta + \tau m(2^d - 1) \qquad (6.2)$$

O pacote formado por cada núcleo contem 12 elementos (os valores correspondentes às saídas das células, mais um número de identificação), o que produz uma mensagem de apenas 48 bytes. Usando os mesmos valores anteriores para β e τ, na expressão (6.2), obtem-se o tempo total de execução para o algoritmo MULTI-BROADCAST igual a 3651,9≈3652 μs., o que equivale a um ciclo de iteração da rede neural. A diminuição do tamanho do pacote num fator de 0,5, eliminando o número de identificação de cada célula, leva a um tempo total de 3399,9≈3400 μs.

Numa versão algorítmica para o *Multi-Broadcast* que é a simetrização do algoritmo descrito acima (*Complete Broadcast* em [24]), as mensagens no d-cubo são divididas em d pacotes. Cada pacote é enviado de forma concorrente, utilizando todos os canais físicos disponíveis. O ganho de tempo de comunicação na implementação deste algoritmo, chamado em [58] de algoritmo de d portas, depende muito do tamanho das mensagens a serem transmitidas e da dimensão do cubo utilizado. Nas estimativas de tempo utilizou-se a expressão obtida por Stout & Wagar em [24]:

$$\{(2^d - 1)\,\tau m\}/d + d\beta \qquad (6.3)$$

Os tempos estimados, considerando a versão "simetrizada" (algoritmo para d portas utilizadas simultaneamente) e a versão para uma porta (algoritmo MULTI-BROADCAST), são representados nas tabelas 6.1-6.4). Utilizando (6.2) e (6.3), para distintas dimensões d do hipercubo, obtiveram-se as seguintes estimativas de tempo teóricas para cada um dos algoritmos.

Tabela 6.1 Tempos de comunicação para MULTI-BROADCAST (1 canal)
comprimento das mensagens = 24 bytes

dimensão	tempo (μs)
3	3399,9
4	4737,2
5	6362,5

Na tabela seguinte representa-se o Ganho de tempo (Redução em %), do algoritmo COMPLETE-BROADCAST [24], para d canais em uso simultâneo, relativo ao algoritmo MULTI-BROADCAST da secção 5.4.3.3.

Tabela 6.2 Tempos de comunicação para COMPLETE-BROADCAST (d canais)
Comprimento das mensagens = 24 bytes

dimensão	tempo(μs)	Redução
3	3231,9	-4,95 %
4	4332,2	-8,55 %
5	5469,7	-14,0 %

No nosso caso, as mensagens transmitidas entre os nós do 3-cubo são formadas por pacotes de 48 bytes. A seguir apresentam-se os tempos de comunicação estimados para os dois algoritmos.

Tabela 6.3 Tempos de comunicação para MULTI-BROADCAST (1 canal)
comprimento das mensagens = 48 bytes

dimensão d	tempo (μs)
3	3651,9
4	5277,2
5	7478,5

Posteriormente, representa-se o Ganho de tempo (Redução em %) do algoritmo COMPLETE-BROADCAST [24], para d canais em uso simultâneo, relativo ao algoritmo MULTI-BROADCAST da secção 5.4.3.3.

Tabela 6.4 Tempos de comunicação para COMPLETE-BROADCAST (d canais)
Comprimento das mensagens = 48 bytes

dimensão	tempo(μs)	Redução
3	3231,9	-9,2 %
4	4332,2	-15,35 %
5	5469,7	-23,9 %

6.6. Medidas de tempos de execução de processos Occam.

Nesta secção apresentam-se algumas medidas tomadas na execução de distintos processos Occam, considerados relevantes no desempenho global do simulador. As medidas foram feitas num ambiente Mono-Transputer, no processador Transputer T800 rodando TDS (Sistema de Desenvolvimento do Transputer), o que implica no pior caso de desempenho, considerando que o TDS deve rodar junto com a aplicação no mesmo processador. O desempenho de processos independentes rodando num processador, sem incluir o sistema TDS, deveria melhorar, num fator que é função da configuração particular e do *hardware* empregado.

Desta forma, num ciclo de iteração, a medida do desempenho dos processos **Nucleo + Multiplex**, incluindo um conjunto de 6 processos células, levou um tempo de execução de 114 *ticks* equivalentes, no Transputer, a 7296 μs. Considerando este valor e a estimativa para o tempo de comunicação, utilizando *Multi-Broadcast*, tem-se a seguinte distribuição de tempos de execução (Fig. 6.2).

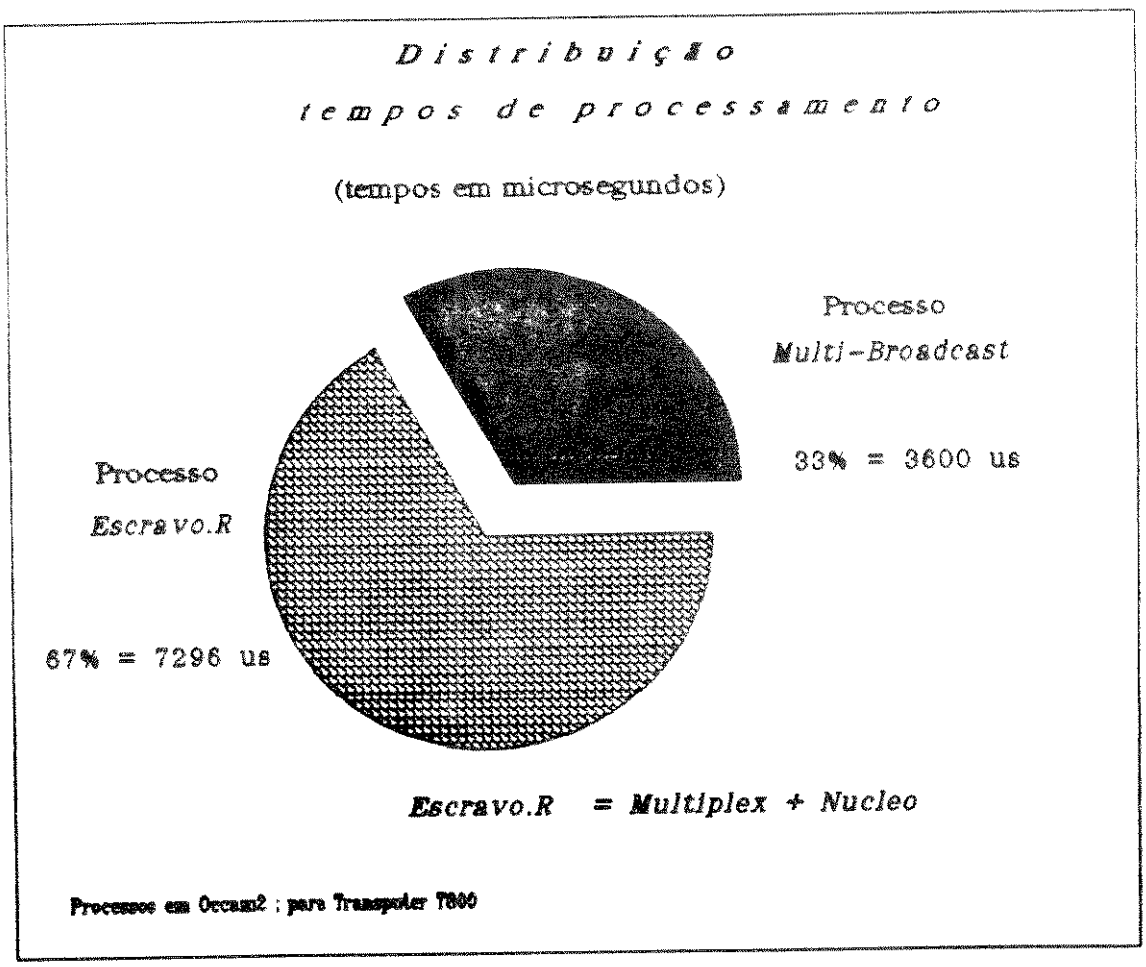

Fig.6.2. Representação gráfica da distribuição do tempo de execução para os processos *Núcleo* e *Multi-Broadcast*

Um dos parâmetros de medida importante é a velocidade de comunicação entre processos concorrentes sendo executados no mesmo processador. Tal comunicação é feita através de canais lógicos "internos", implementados como uma palavra na memória local de cada processador. A seguir, na Fig. 6.3, apresentam-se algumas medidas obtidas de processos seqüenciais, comunicando-se através de canais lógicos, para distintos tamanhos de mensagens de tipo escalar e em função do número de canais implementados entre os processos. Foi implementado um protocolo distinto para cada comunicação, do tipo:

PROTOCOL INT [,n INT], onde $1 \leq n \leq 10$.

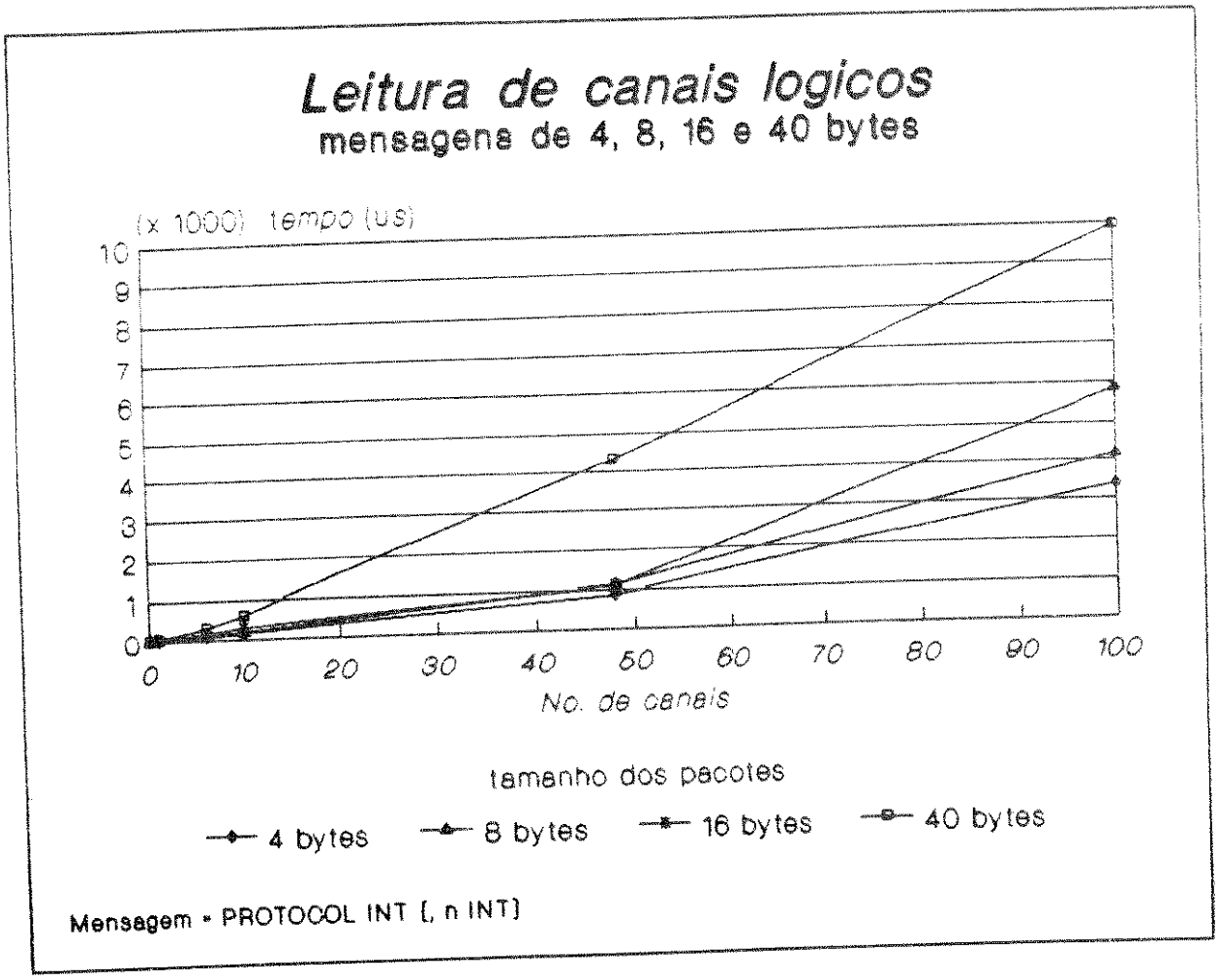

Fig. 6.3. Tempo de comunicação entre dois Processos Seqüenciais Comunicantes Occam v/s no. de canais lógicos. Comunicação utilizando pacotes de mensagens de distintos comprimentos : 4-bytes, 8-bytes, 16-bytes e 40 bytes.

Na Fig. 6.4 representam-se as medidas obtidas para a comunicação entre dois processos Occam, sendo executados no mesmo processador e utilizando pacotes de mensagens do tipo vetores (*arrays*). Os canais de comunicação foram também implementados *internamente* como palavras na memória do processador. O protocolo de comunicação utilizado foi do tipo:

PROTOCOL [n]INT , onde n é o tamanho do vetor; $1 \leq n \leq 100$.

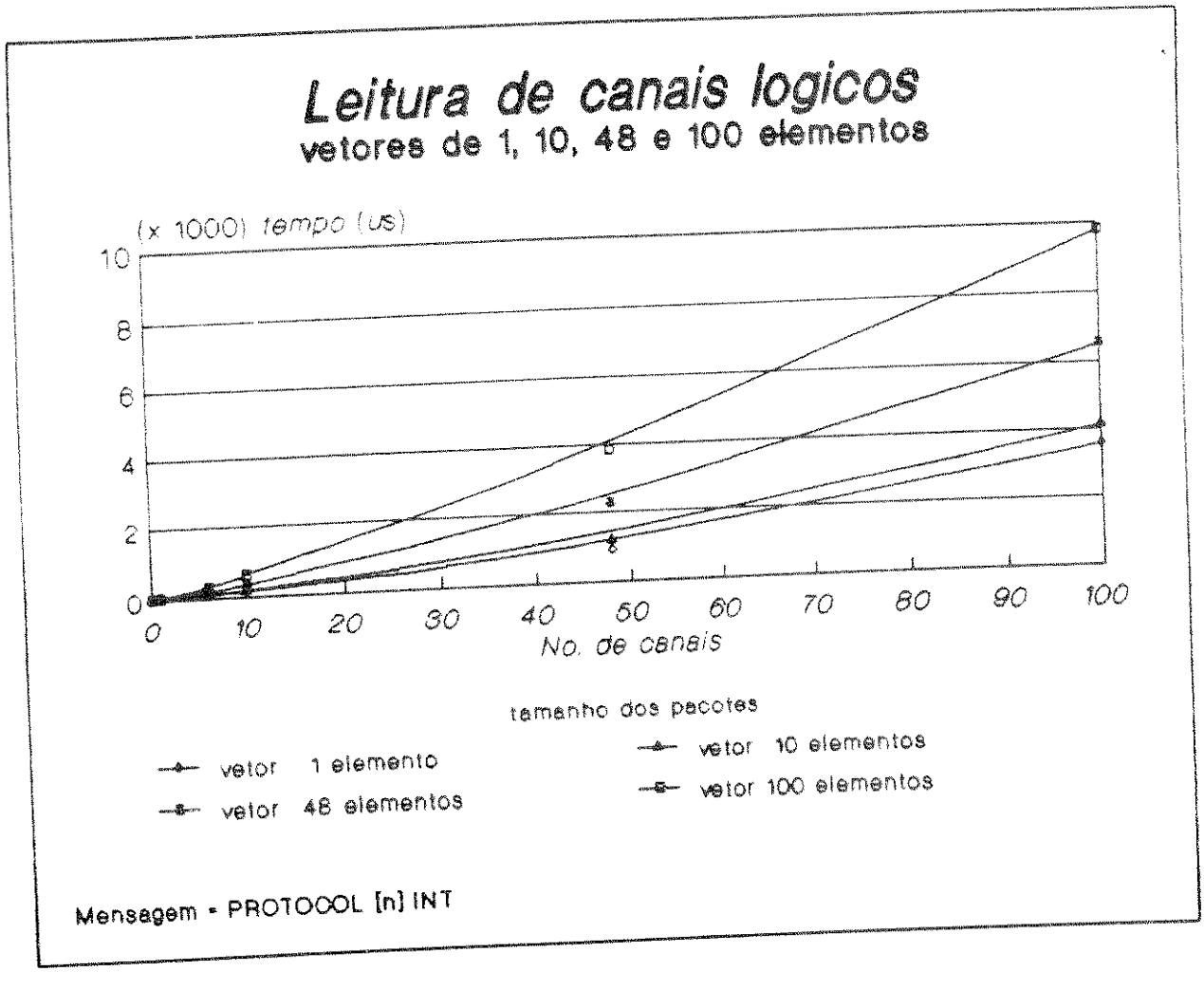

Fig. 6.4. Tempo de comunicação entre dois Processos Seqüenciais Comunicantes Occam v/s no. de canais lógicos. Comunicação utilizando pacotes de mensagens do tipo Vetores de n elementos. $1 \leq n \leq 100$. O Protocolo de comunicação é do tipo PROTOCOL [n]INT ; n : tamanho do vetor.

6.7.Pacotes de Mensagens entre nós.

O desempenho do simulador é influenciado principalmente pela capacidade de comunicação entre os processadores físicos. Das estimativas descritas nas secções anteriores, deriva-se que, o tempo de comunicação entre os processos em execução paralela representa um fator importante no tempo de processamento global (Fig 6.2).

Os critérios de decisão relacionados com a comunicação entre processos paralelos foram influenciados principalmente pelas estimativas do tempo que leva as mensagens a serem transmitidas. Buscou-se, principalmente, reduzir o tempo de comunicação (roteamento das mensagens e transmissão). A fim de minimizar o tempo de transmissão, decidiu-se implementar os pacotes como vetores de tamanho variável, devido ao melhor desempenho obtido com processos concorrentes Occam (Fig. 6.4.). Um pacote elementar é formado por um vetor contendo o número de identificação de cada célula + o seu valor de saída respectivo, totalizando um núcleo de 6 células. Em cada estágio do algoritmo *Multi-Broadcast* o tamanho do pacote é duplicado. O formato do pacote elementar é representado na Fig. 6.5.

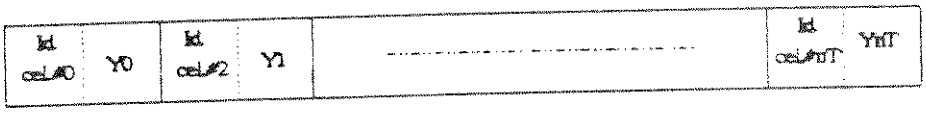

Fig. 6.5.Formato do pacote utilizado na transmissão de uma mensagem entre os nós .

Na Fig. 6.5, cel.#i representa a identificação do neurônio i , Yi é o seu valor de saída respectivo e nT é o número de células por processador. Os algoritmos de comunicação de Stout & Wagar, que foram utilizados neste trabalho, permitem eliminar um cabeçalho identificando os n.id (endereços) dos nós emissor e receptor, reduzindo o tamanho das mensagens e, conseqüentemente, o tempo que leva um pacote para ser transmitido desde o nó corrente ao vizinho.

7. Conclusão

A simulação de Redes Neurais apresenta-se mais interessante quando são utilizadas técnicas de processamento paralelo e sistemas de computadores de arquiteturas MIMD, do tipo de passagem de mensagens. O interesse surge devido principalmente às perspectivas de melhoras substanciais no desempenho das simulações. No entanto, nem sempre é obtido um bom desempenho ao utilizar máquinas paralelas. Isto, muitas vezes, é devido ao fato dos algoritmos (ou um modelo neural particular) não serem adequados à topologia da máquina. Note-se, por ex., que a simulação de um modelo neural completamente conectado, numa máquina MIMD configurada como uma topologia árvore -onde cada nó representa um processador- produziria uma sub-utilização do sistema. Isto é devido principalmente ao volume de comunicação excessivo, necessário nos nós dos níveis superiores da árvore, em contraposição ao dos nós nos níveis inferiores.

Em contrapartida, a topologia hipercúbica revela-se como uma configuração eficiente quando se tem uma carga de trabalho balanceada no conjunto de processadores, embora atrasos produzidos no roteamento das mensagens entre os nós constituam o fator limitante no desempenho do sistema.

Neste trabalho abordamos uma técnica de mapeamento de ANNs em sistemas de multicomputadores, que tem produzido resultados satisfatórios em máquinas com um grande número de processadores, da ordem de $O(10^2)$-(10^5). O mapeamento e a simulação paralela do modelo neural binário de J. J. Hopfield, funcionando como uma memória associativa (ou uma CAM), foram abordados objetivando um sistema de multicomputadores configurado como um 3-cubo, enfatizando os aspectos relativos à comunicação de mensagens entre os nós.

Uma primeira contribuição foi a de adequar os eficientes algoritmos de Stout & Wagar à nossa aplicação, viabilizando assim a implementação virtual da Rede de Hopfield numa máquina de arquitetura paralela, com um custo computacional mínimo.

Um outro aspecto é ter aproveitado as características da linguagem Occam, utilizando o esquema de processos seqüenciais comunicantes concorrentes (CSP). Isto permite representar e descrever, de forma mais "natural", a estrutura altamente paralela da rede neural. Tal metodologia facilita bastante o desenvolvimento da programação dos processos a serem executados em paralelo.

Embora o desenvolvimento do simulador tenha sido feito num ambiente Mono-Transputer, a metodologia utilizada, junto às características do ambiente, permite transportar o *software* a um sistema formado por uma rede de processadores Transputers, sem maiores dificuldades. As modificações deverão ser feitas mudando somente a configuração relativa às alocações dos canais que implementam a comunicação entre os processos *Escravos* e o processo *Mestre*.

Ao respeito do desempenho do simulador, as únicas medidas que nos permitiram ter uma estimativa do desempenho global foram as dos tempos de execução medidos isoladamente em cada processo *Escravo*, as dos tempos medidos na leitura de pacotes de distintos tamanhos e as das estimativas teóricas para a execução dos algoritmos de comunicação. Uma medida suficientemente confiável, tanto do *Speed-up* quanto da Eficiência [25] de nossas simulações, só será possível com a execução de todos os processos num sistema Multi-Transputer. Isto permitiria, ademais, "afinar" a execução (no sentido de minimizar o tempo de execução) dos processos concorrentes no sistema objeto. A implementação final num hipercubo deve permitir adequar o nosso *software* à arquitetura da máquina, visando detetar os pontos onde a execução se revele mais ineficiente.

Um dos resultados que estimamos importante foi quantificar o custo computacional e portanto, a influência das comunicações e o roteamento das mensagens entre os nós.

Por outro lado, a falta de alocação dinâmica da memória do Transputer, utilizando Occam, impede otimizar o uso da memória local disponível em cada processador, pelo que deve-se tratar cuidadosamente o aspecto relacionado com o tamanho dos pacotes. Em algumas circunstâncias, foi necessário sacrificar a velocidade de processamento, evitando o uso de técnicas de *batching* na transmissão de pacotes, devido principalmente às limitações de memória local.

A utilização de técnicas de transmissão de pacotes e o uso de algoritmos de comunicação eficientes entre os nós (em termos de velocidade), em vez de mensagens personalizadas, permitiram reduzir o custo produzido pela comunicação intensiva.

A implementação de redes neurais artificiais, para simulação altamente concorrente, revela-se ainda mais rápida em sistemas de *hardware*, tanto utilizando tecnologias VLSI quanto sistemas ópticos. Em contrapartida, o seu alto custo de implementação e a dificuldade, até hoje, de construção de ANNs de grande porte (da ordem de $O(10^8)$ neurônios), fazem com que as máquinas de arquitetura paralela do tipo de passagem de mensagens tornem-se bastante viáveis.

Referências Bibliográficas

[1] Richard Lippmann; "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, April 1987.

[2] Stephen Gallant; "Connectionist Expert Systems", Communications of the ACM, pp. 152-169, V.31, N.2, February 1988.

[3] J.A. Feldman et al.; "Computing with Structured Connectionist Networks", Communications of the ACM, pp. 170-187, V.31, N.2, February 1988.

[4] W. Hilberg; "Neural Network and Conditional Association Networks : Common properties and differences", IEE Proc., V.136, Part E, No.5, Sept. 1989.

[5] Jiro Naganuma et al.; "High-Speed CAM-Based Architecture for a Prolog Machine (ASCA), IEEE Trans. on Comp., V.37, No.11, Nov. 1988.

[6] M.Verleysen, B. Sirletti, A. Vandemeulebroecke & Paul Jespers; "A High-Storage Capacity Content-Addressable Memory and its Learning Algorithm", IEEE Trans. on Circuits and Systems, V.36, No.5, May 1989.

[7] S. Jones; "Design, Selection and Implementation of a Content Addressable Memory for a VLSI CMOS Chip Architecture", IEE Proc., Part E, No. 3, May 1988.

[8] Takeshi Ogura et al.; "A 20-Kbit Associative Memory LSI for Artificial Intelligence Machines", IEEE Journal of Solid State Circuits, V.24, No. 4, August 1989.

[9] M.A.C. Mahler, S.P. DeWeerth, M.A. Mahawold and Carver Mead; "Implementing Neural Arquitectures using VLSI Analog Circuits", IEEE Transactions on Circuits and Systems, V.36, No.5, May 1989.

[10] A.F. Murray; "Silicon Implementations of neural networks", IEE Proceedings-F, Radar and Signal Processing, V. 138, No. 1, pp. 3-12, February 1991.

[11] James D. Robert; "A CAM-Based Search Accelerator", IEEE Computing Futures, Inaugural Issue, Winter 1989-1990.

[12] H. Ch. Zeidler; "Content Addressable Mass Memories", IEE Proc., V.136, Part E, No. 5, Sept. 1989.

[13] K. E. Grosspietsch; "Architectures for Testability and Fault Tolerance in Content-Addressable Systems" IEE Proc., V.136, Part E, No.5, Sept. 1989.

[14] A. W. G. Duller et al.; "Design of an Associative Processor Array", IEE Proc., V.136, Part E, No.5, Sept. 1989.

[15] M. Hassoun & P. Watta; "Exact Associative Neural Memory Dynamics Utilizing Boolean Matrices", IEEE Transactions on Neural Networks, V. 2, No. 4, July 1991.

[16] G.W. Stewart; "Communication and matrix computations on large message passing systems", Parallel Computing, Vol. 16, pp. 27-40, February 1990.

[17] J.J. Hopfield; "Neural Networks and Physical Systems with Emergent Collective Computational Abilities ", Proc. Natl. Acad. Sci. U.S.A., Vol. 79, pp. 2554-2558, April 1982.

[18] Kohonen, Teuvo, "Self-Organization and Associative Memory", Springer-Verlag, Berlin, Germany, 1984.

[19] Lawrence Chisvin; "Content Addressable and Associative Memory : Alternatives to the Ubiquitous RAM", IEEE Computer Magazine, July 1989.

[20] J.A.G. Nijhuis; "Fault Tolerance of Neural Associative Memories", IEE Proc., V.136, Part E, No.5, Sept. 1989.

[21] David May; "Occam", SIGPLAN Notices, V.18, No.4, April 1983.

[22] Dick Pountain & David May; "A Tutorial Introduction to Occam-2 Programming", Inmos BSP Professional Books, March 1988.

[23] David May et al. ; "The IMS T800 Transputer ", IEEE Micro Magazine, October 1987.

[24] Q. Stout & B. Wagar; "Intensive Hypercube Communication", Journal of Parallel and Distributed Computing", V. 10, pp. 167-181, 1990.

[25] Michael Quinn; "Designing Efficient Algorithms for Parallel Computers", McGraw-Hill International Ed.,1988.

[26] R.M. Fujimoto; "Parallel Discrete Event Simulation", Communications of the ACM, Vol. 33, No. 10, pp. 31-53, October 1990.

[27] J.Gosh & K. Hwang; "Mapping Neural networks onto message-passing Multicomputers", Journal of Parallel and Distributed Computing, Vol. 6, pp. 291-330, 1989.

[28] J.J. Hopfield and D.W. Tank; "Computing with Neural Circuits: A Model", SCIENCE, Vol. 233, pp. 625-633, August 1986.

[29] Da Silva, J.G.D. and Watson, I, "Pseudo-Associative Store with Hardware Hashing", IEE Proc., V.130, Part E, No. 1, January 1983.

[30] Teuvo Kohonen; "Content Addressable Memories", Springer-Verlag, Berlin, Germany, 1980.

[31] A. G. Hanlon; "Content Addressable and Associative Memory Systems", IEEE Trans. Electronic Computers, V. EC-15, No. 4, Aug. 1966.

[32] G. M. Blair; "Content-Addressability : An Exercise in the Semantic Matching of Hardware and Software Design", IEE Proc., V.136, Part E, No.1, Jan. 1989.

[33] B. Parhami; "Associative Memories and Processors : An Overview and Selected Bibliography", Proc. IEEE, Vol. 61, No. 6, June 1973.

[34] Yoh-Han Pao; "Adaptive Pattern Recognition and Neural Networks", Addison-Wesley, 1989.

[35] Shun-Ichi Amari; "Characteristics of Random Nets of Analog Neuron-Like Elements", IEEE Transactions on Systems, Man, & Cybernetics, Vol. SMC-2, No. 5, November 1972.

[36]Shun-Ichi Amari; "Mathematical Foundations of Neurocomputing", Proceedings of the IEEE, V. 78, No. 9, September 1990.

[37] Shun-Ichi Amari;in "Dynamics Interactions of Neural Networks",Springer-Verlag, New York, 1989.

[38] Teuvo Kohonen; "Correlation Matrix Memories", IEEE Transactions on Computers, Vol. C-21, No. 4, April 1972.

[39] Kaoru Nakano; "Associatron-A Model of Associative Memory", IEEE Transactions on Systems, Man & Cybernetics, SMC-2, No.3, July 1972.

[40] L. Tarassenko, J. Tombs & J. Reynolds; "Neural Network Architectures for Content-Addressable Memory", IEE Proceedings-F,Radar and Signal Processing, Vol. 138, No. 1, pp. 33-39, February 1991.

[41]R. Mc Eliece et al.; "The Capacity of the Hopfield Associative Memory", IEEE Transactions on Information Theory, Vol. IT-33, No. 4, July 1987.

[42]A. Kuh & B. Dickinson; "Information Capacity of Associative Memories", IEEE Transactions on Information Theory", Vol. 35, No. 1, January 1989.

[43] Teuvo Kohonen; " Self-Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics Vol. 43, No. 69, 1982.

[44] J. Kangas, T. Kohonen e J. Laaksonen; "Variants of Self-Organizing Maps"; IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990.

[45] Bernard Widrow and Michael A. Lehr; "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation", Proceedings of the IEEE, V. 78, No. 9, September 1990.

[46]B. Kosko; "Adaptive Bidirectional Associative Memories"; Applied Optics, Vol.26, No. 23, December 1987.

[47] Yeou-Fang Wang et al.; "Two Coding Strategies for Bidirectional Associative Memory", IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990.

[48] D.W. Tank, and J.J. Hopfield; "Neural Computation by Concentrating Information in Time", Proc. Natl. Acad. Sci. U.S.A., Vol. 84, pp. 1896-1900, April 1987.

[49] D. E. Van den Bout & T. K. Miller III; "A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets", IEEE Trans. on Circuits and Systems, V.36, No.5, May 1989.

[50] Nabil Farhat et al.; "Optical Implementation of the Hopfield Model", Applied Optics, Vol. 24, No. 10, May 1985.

[51] A.L.Mikaelian, B.S.Kiselyov et al.;"Optical Implementation of High-Order Associative Memory", International Journal of Optical Computing, Vol. 1, pp. 89-92, 1990.

[52] C.A.R. Hoare; "Communicating Sequential Processes", Communications of the ACM, Vol. 21, No. 8, pp. 666-677, August 1978.

[53] Ian Gorton, J. Kerridge & B. Jervis; "Simulating Microprocessor Systems using Occam and a Network of Transputers", IEE Proceedings, Vol. 136, part E, No.1, January 1989.

[54] J. Van der Spiegel et al. ;"Artificial Neural Networks : Principles and VLSI Implementation", Anais V Congresso SBMicro, Campinas, SP, Julio 1990.

[55] C. Seitz; "The Cosmic Cube", Communications of the ACM, Vol. 28, No. 1, pp. 22-33, January 1985.

[56] R. Ginosar & D.Egozi; "Topological Comparison of Perfect Shuffle and Hypercube", International Journal of Parallel Programming, Vol. 18, No. 1, pp. 37-69, 1989.

[57] C. Lamana & M. Shaw; "A Performance Study of the Hypercube Parallel Processor Architecture", Simulation, Vol.56, No.3, March 1991.

[58] Pierre Fraigniaud; "Complexity Analysis of Broadcasting in Hypercubes with Restricted Communication Capabilities", Technical Report 90-16, Mai 1990, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon.

[59] Y. Saad & M.Schultz; "Data Communication in Hypercubes", Journal of Parallel and Distributed Computing, Vol. 6, pp. 115-135, 1989.