ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

Otimização de Roteamento em Redes de Computadores utilizando Redes Neurais de Hopfield

Trabalho de Conclusão de Curso

Engenharia da Computação

Robson Alcântara Santana
Orientador: Adriano Lorena Inácio de Oliveira
Co-orientador: Carmelo José Albanez Bastos Filho

Recife, maio de 2007

UNIVERSIDADE
DE PERNAMBUCO

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Departamento de
Sistemas
Computacionais

Otimização de Roteamento em Redes de Computadores utilizando Redes Neurais de Hopfield

Trabalho de Conclusão de Curso

Engenharia da Computação

Este Projeto é apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia da Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Robson Alcântara Santana
Orientador: Adriano Lorena Inácio de Oliveira
Co-orientador: Carmelo José Albanez Bastos Filho

Recife, maio de 2007

UNIVERSIDADE
DE PERNAMBUCO

Robson Alcântara Santana

Otimização de Roteamento em Redes de Computadores utilizando Redes Neurais de Hopfield

Resumo

Em redes de computadores, particularmente nas redes de comutação de pacotes, o processo de roteamento tem um significante impacto no desempenho das redes. É desejável que um algoritmo de roteamento encontre uma rota entre a origem e o destino no menor tempo possível, para satisfazer a demanda dos usuários e fornecer um rápido serviço. O problema do roteamento de pacotes em redes de comutação de pacotes, sob certas circunstâncias, pode ser considerado como um problema de otimização, mais especificamente, o problema do menor caminho, onde Redes Neurais Artificiais são ótimas candidatas para serem aplicadas a este problema, mais especificamente a Rede Neural de Hopfield.

Trabalhos anteriores já resolveram este problema, porém este trabalho propõe uma nova abordagem para acelerar o algoritmo de roteamento baseado na Rede Neural de Hopfield. Outros trabalhos similares utilizam métodos sofisticados, como o de Runge Kutta, para resolver uma das equações do algoritmo. Nós mostramos que nosso método pode calcular a melhor rota em termos do custo em uma rede usando uma equação discreta ao invés da resolução da formulação diferencial utilizando o método de Runge Kutta.

Para implementar esta nova abordagem, foi desenvolvido um simulador em Java. Neste simulador foram implementados o algoritmo de Dijkstra, o algoritmo de Ali e Kamoun e nosso algoritmo. Foram realizadas diversas simulações e os resultados mostraram que o algoritmo proposto pode encontrar rotas ótimas mais rápido que o algoritmo de Ali e Kamoun. Como resultado um artigo científico completo foi publicado no simpósio “IEEE Foundations on Computational Intelligence”.

Abstract

In a computer network, particulary in packet-switched networks, the routing process has a significant impact in the performance of the nets. It is desirable that a router algorithm finds a route between the source and the destination in the lower possible time, to satisfy the demand of the users and to supply a fast service. The routing problem of packet-switched networks, under appropriate assumptions, can be considered as an optimization problem, more specifically, the problem of the shortest path, where Artificial Neural Networks are excellent candidates to be applied to this problem, more specifically the Hopfield Neural Network.

There are other solutions to this problem in previous works, however this work considers a new boarding to speed up the router algorithm based on the Hopfield Neural Network. Other similiar works uses sophisticated methods, as Runge Kutta's, to resolve one of the algorithm equations. We show that our method can calculate the best route in terms of the cost in a net using a discrete equation instead of the resolution of the distinguishing formularization using the Runge Kutta method.

To implement this new approach, was developed a simulator in Java. It was implemented, in this simulator, Dijkstra's altorithm, the Ali and Kamoun algorithn and our algorithm as well. Several simulations had been carried through and the results had shown that the considered algorithm can find excellent routes faster than the Ali and Kamoun algorithm. As result a complete scientific article was published in the symposium “IEEE Foundations on Computational Intelligence”.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

SP – *Shortest Path*
QoS – *Quality of Service*
CI – *Computational Inteligence*
RNA – Redes Neurais Artificiais
TSP – *Traveling Salesman Problem*
HNN – *Hopfield Neural Network*
HNN-RK – *Hopfield Neural Network – Runge Kutta*
HNN-DE – *Hopfield Neural Network – Equação Diferença*
IEEE – *Institute of Electrical and Electronics Engineers*
WAN – *Wide Area Network*
RWA – *Routing and Wavelenght Assignment*
MLP – *Multi-Layer Perceptron*
SOM – *Self Organizing Maps*
ART – *Adaptive Resonance Theory*
MCP – *McCulloch e Pitts*
ICANN – *Internet Corporation for Assigned Names and Numbers*
TCP – *Transmission Control Protocol*
IP – *Internet Protocol*
OSI – *Open Systems Interconnection*
ISO – *International Standards Organization*
CV – Circuito Virtual
LS – *link-state*
DV – *distance-vector*
OSPF – *Open Shortest Path First*
IGP – *Interior Gateway Protocol*
RIPv1 – *Routing Information Protocol*
IGRP – *Interior Gateway Routing Protocol*
EGP – *Exterior Gateway Protocol*
BGP – *Border Gateway Protocol*
SA – Sistema Autônomo
ADSL – *Asymmetric Digital Subscriber Line*

Agradecimentos

Agradeço principalmente a Deus.

Agradeço ao meu pai Roberto Santana por ter me apoiado em todos os momentos que precisei e por ter me incentivado a fazer este curso. A minha mãe Maria José Alcântara por estar de braços abertos sempre que necessário. Agradeço a meus tios por terem colaborado na minha formação, principalmente às minhas tias Zilma Santana e Zaíra Santana (*in memoriam*). Ao meu avô Espedito Santana e minha avó Ana Martins por me receberem sempre de braços abertos. À minhas irmãs Jackeline Santana, Roberta Santana, Ana Paula Santana e Vanessa Santana irmãos que sempre torceram por mim. E a todos meu familiares.

Agradeço também a todos os professores do Departamento de Sistemas Computacionais em especial aos professores Fernando Buarque, Carlos Alexandre e Ricardo Massa por terem sido importantes na minha formação academia. E principalmente aos meus orientadores neste trabalho Adriano Lorena e Carmelo Albanez por terem sido atenciosos em todos os momentos que precisei e sempre me motivaram.

Agradeço aos meus chefes Robson Costa e Sebastião Ferreira por terem sido flexíveis nos momentos mais difíceis para que eu pudesse desenvolver este projeto. Agradeço também a meus colegas de estágio.

Aos meus colegas de faculdade Tiago Fragoso, Júlio Dantas, George Silva e Wesnaida Holanda por terem colaborado diretamente na conclusão deste trabalho.

Aos meus amigos João Ferreira, Hamilton Estima, Washington Santana, Larissa Miranda, Aline Morais e Virgínia Santana que foram alguns dos poucos amigos que me apoiaram neste momento.

Por fim agradeço a meus amigos e colegas de faculdade Juliane Botelho, Tássia Lima, Emanuel Aragão, Frederico Álvares, Júlio Taveira, Vítor Moura, Renata Wanderley, Marcos André e principalmente a meu “irmão” Gabriel Falconieri.

Capítulo 1

Introdução

Nas redes de comunicação por comutação de pacotes, dados são roteados ao longo de enlaces de comunicação que compõe a rede. Existem diversas possibilidades para que um pacote de dados navegue do seu nó fonte, através de múltiplos nós, para seu(s) destino(s). Algoritmos de roteamento vêm sendo propostos e discutidos intensamente na comunidade científica, principalmente porque o processo de roteamento tem um drástico impacto no desempenho das redes de comunicações. O algoritmo de roteamento "ideal" visa encontrar o(s) caminho(s) "ótimo(s)" entre o nó fonte e o nó destino, permitindo uma alta qualidade de transmissão de dados, evitando as penalidades causadas por disparidades da camada física e minimizando a perda de pacotes. Uma vez que o tráfego na conexão moderna é caracterizado por uma imensa quantidade de pares de fonte e destino, alta variabilidade de carga distribuída na rede e não-linearidade, o processo de roteamento é uma tarefa muito complexa. Existem diversas formas de encontar o melhor caminho. Alguns algoritmos determinam as rotas baseados no menor caminho (SP – *Shortest Path*) [1], menor atraso [2], maior relação sinal-ruído (no caso de Redes Totalmente Ópticas) [3], balanceamento de carga [4], entre outros.

A maioria das redes de computadores de comutação de pacotes utiliza alguma forma de computar o menor caminho que é requerido pelo algoritmo de roteamento da camada de rede, onde cada enlace tem um parâmetro ou um custo, que representa o desejo em usar aquele enlace particular. Um algoritmo de roteamento menor caminho é usado para achar o custo do menor caminho entre um nó fonte e um nó destino.

Além disso, para manter a Qualidade do Serviço (QoS – *Quality of Service*) computações têm que ser realizada em tempo real e devem ser adaptativas. Para fornecer essa flexibilidade, muitas técnicas de Inteligência Computacional (CI – *Computational Inteligence*) foram aplicadas a este problema. As técnicas mais usadas são as Redes Neurais Artificiais (RNA) [2, 4-11]Otimização por Colônia de Formigas [11], [12], Algoritmos Genéticos [13], [14], e algoritmos Híbridos combinando estas técnicas [15], [16].

RNAs são ótimas candidatas para resolver o problema do roteamento em redes de computadores devido a sua elevada velocidade computacional e implementação natural em hardware, dado ao paralelismo inerente às RNA's. Hopfield e Tank descreveram uma RNA com configuração de *feedback* apropriada para resolver problemas relacionados a otimização, especialmente o Problema do Caixeiro Viajante (TSP – *Traveling Salesman Problem*) [17]. Conseqüentemente, esta configuração de RNA é chamada de Rede Neural de Hopfield (HNN – *Hopfield Neural Network*) [18].

O uso de HNN para achar o menor caminho entre dois nós em uma rede de comunicação foi iniciada por Rauch e por Winarske [19]. Entretanto, esta proposta requer um conhecimento prévio da topologia da rede de computadores, ou seja, a solução deve ser construída especificamente para cada topologia de rede formada. Para superar esta limitação, Ali e Kamoun [2] propuseram um novo algoritmo adaptável, onde a matriz de pesos carregasse apenas a informação da convergência.

1.1 Motivação

Algoritmos de roteamento são de grande importância no processo de roteamento de redes de computadores, principalmente nas redes de comutação de pacotes. Para satisfazer a demanda de usuários e manter um bom serviço em uma rede de computadores, é importante que o algoritmo de roteamento da rede seja rápido, sendo um desafio reduzir o tempo de resposta de algoritmos que já são eficientes.

Com a finalidade de acelerar a resposta de um algoritmo de roteamento baseado em RNAs, nós desenvolvemos um simulador e implementamos dois algoritmos já existentes e um novo algoritmo baseado no algoritmo de Ali e Kamoun.

1.2 Objetivos do Trabalho

Este trabalho tem como principal objetivo propor uma nova abordagem para aumentar a velocidade das HNN utilizando Equação diferença finita e discreta para aumentar a velocidade de convergência.

Os objetivos específicos do trabalho são:

- modificar o algoritmo de Ali e Kamoun. Uma das fórmulas deste algoritmo utiliza o método de Runge-Kutta (HNN-RK) [20] que é um método sofisticado. Substituindo este método por Equação diferença (HNN-DE) a convergência será acelerada.

- construir um simulador que encontre o menor caminho em redes de computadores. Esta aplicação terá alguns algoritmos de roteamento implementados;

- a partir do simulador implementado, realizar simulações e comparações entre os mesmos.

Além dos objetivos propostos, conseguimos submeter um artigo científico [21] para a conferência internacional FOCI 2007 do IEEE (*Institute of Electrical and Electronics Engineers*) [22]. Este artigo é um resumo desta monografia.

1.3 Trabalhos Relacionados

No início dos anos 80 Hopfield publicou dois artigos científicos [18][35], que caracterizaram como o reinício pelo interesse na área de RNAs, área esta que estava desaquecida por mais de dez anos. O trabalho de Hopfield foi um grande avanço na fronteira do conhecimento das RNAs. Hopfield mostrou que modelos de sistemas físicos poderiam ser usados para resolver problemas computacionais. Tais sistemas poderiam ser implementados em *hardware*, combinando

componentes básicos como capacitores e resistores. Redes de Hopfield são tipicamente utilizadas para resolver problemas de classificação, como memória assossiativa e para resolver problemas de otimização. O modelo de Hopfield é utilizado no algoritmo de roteamento que este trabalho propõem.

Redes ópticas transparentes [23] são redes *Wide Area Network* (WAN) [46] que ainda estão em um estágio de desenvolvimento inicial. Essas redes têm a característica de não necessitar de conversão óptica-elétrica-óptica nos nós intermediários. Com essa característica não existe o inconveniente de converter o sinal luminoso em sinal elétrico e vice-versa, dando mais velocidade a rede óptica. Essa rede tem como uma de suas funções a definição do roteamento e seleção do comprimento de onda (RWA – *Routing and Wavelenght Assignment*) [24], antes de iniciar a comunicação, porém existem problemas com algoritmos de roteamento e algoritmos de alocação de comprimento de onda. Bastos-Filho *et al* [25] propuseram e demonstraram um algoritmo dinâmico de roteamento para as redes ópticas transparentes baseado nos impecilhos da camada física. O algoritmo proposto neste trabalho fará parte do simulador de Redes Ópticas do trabalho de iniciação científica de Bastos-Filho *et al* e será mais uma opção de algoritmo de roteamento.

1.4 Estrutura do Trabalho

Esta monografia está estruturada da seguinte forma

- **Capítulo 2** – Neste capítulo, iniciamos descrevendo as Redes Neurais Artificiais, falando sobre os tipos de redes e problemas que podem ser solucionados. Em seguida apresentamos a Rede Neural de Hopfield, detalhando seu histórico, seu funcionamento, suas aplicações e variações;

- **Capítulo 3** – Descrevemos os conceitos básicos de roteamento de redes, detalhando os tipos de algoritmos de roteamento. Neste capítulo temos ainda uma explicação sobre o algoritmo de Dijkstra, um breve comentário sobre algoritmos do tipo vetor-distância e uma comparação entre algoritmos estado de enlace e vetor distância;

- **Capítulo 4** – Mostramos como a Rede Neural de Hopfield é aplicada para o problema do menor caminho. Além disso, detalhamos a modificação feita no algoritmo de Ali e Kamoun, que é um dos objetivos deste trabalho;

- **Capítulo 5** – Neste capítulo, mostramos um conjunto de simulações executadas para comparar o método de roteamento proposto neste trabalho com o método proposto por Ali e Kamoun;

- **Capítulo 6** – Neste capítulo apresentamos as conclusões deste trabalho, as contribuições e os trabalhos que poderão ser desenvolvidos no futuro.

Além dos capítulos, esta monografia inclue um anexo:

- **Anexo 1** – Anexo destinado ao artigo científico baseado nesta monografia, publicado na conferência internacional FOCI 2007 do IEEE (*Institute of Electrical and Electronics Engineers*).

Capítulo 2

Redes de Hopfield

Uma Rede Neural Artificial é um sistema de computação que visa trabalhar no processamento de dados de maneira semelhante ao cérebro humano. Devido a essa similaridade com o cérebro humano, as RNA devem ser capazes de aprender e tomar decisões baseadas na aprendizagem. As Redes Neurais representam uma tentativa de superar as limitações da computação convencional, utilizando algumas vantagens do cérebro humano, como: alto grau de paralelismo, tolerância à falhas, robustez, capacidade de adaptação, auto-organização, etc. As semelhanças dessas redes ao cérebro estão no fato de que o conhecimento é obtido através de etapas de aprendizagem (treinamento) e pesos sinápticos são usados para armazenar o conhecimento. Sinapse Nervosa é o contato funcional entre membranas da célula nervosa com outra célula nervosa, com um efetor (músculo ou glândula) ou com uma célula receptora sensitiva [26]. Os pesos sinápticos são as "sinapses nervosas" nas RNAs; tais pesos são responsáveis pelo aprendizado e pelo armazenamento de informações.

Uma RNA é um sistema paralelo e distribuído composto por um conjunto de neurônios artificiais que são as unidades de processamento. Essa forma de computação é utilizada para aproximar funções matemáticas (lineares e não-lineares).

A quantidade de conexões entre os neurônios (pesos sinápticos) é importante no processo de aprendizado. Poucas conexões fazem com que a rede tenha um aprendizado insuficiente para o problema a ser aplicado (*underfitting*) [27]. Uma rede com grande número de conexões pode ter o processo de generalização prejudicado, devido ao fato que a rede se adapta a ruídos (*overfitting*) [27].

Um fator que influencia no processo de treinamento de uma RNA são os valores iniciais dos pesos sinápticos. É recomendado utilizar valores aleatórios para que a rede retorne diferentes resultados nos treinamentos.

Existem diversos tipos de RNAs tais como: *Perceptron* [28], *Adaline* [29], Redes Neurais de Hopfield, Redes *Perceptron* Multi-Camadas (MLP – *Multi-Layer Perceptron*) [30], Mapas auto-organizáveis (SOM – *Self Organizing Maps*) [31], ART (*Adaptive Resonance Theory*) [32], etc. A escolha da arquitetura é um fator importante na resolução do problema da qual a rede será aplicada. RNAs podem ser aplicadas a alguns tipos de problemas, entre eles: aproximação de funções, previsão de séries temporais, classificadores, reconhecimento de padrões, problemas de otimização, etc.

De acordo com o problema a ser solucionado é escolhido o tipo neural. Deve ser selecionada uma arquitetura, em que deve ser escolhido o número de camadas usadas, a quantidade de neurônios em cada camada, o tipo de sinapse, o processo de treinamento, etc.

O passo fundamental para o surgimento das RNAs foi o modelo matemático proposto por McCulloch e Pitts [33], em 1943, que é baseado no neurônio biológico, chamado de modelo MCP mostrado na Figura 1.

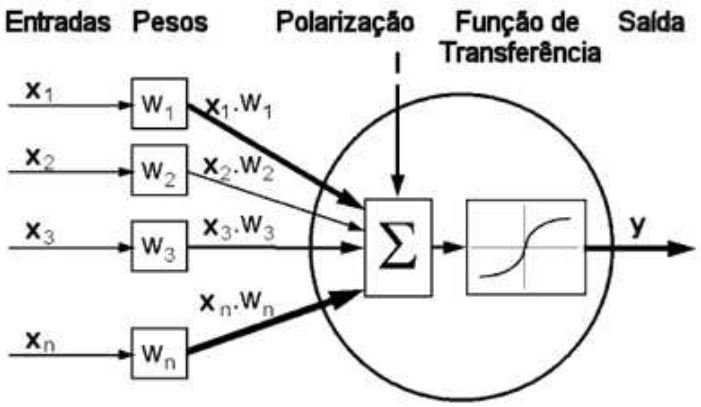

Figura 1. Neurônio artificial de McCulloch e Pitts.

Esse modelo é composto por um conjunto de n entradas x_1, x_2, ..., x_n, pela polarização (bias) I e por uma saída y. Os pesos sinápticos são w_1, w_2, ..., w_n. Valores positivos dos pesos modelam a excitação das sinapses e valores negativos modelam a inibição. O valor da saída do neurônio é obtido pela multiplicação dos pesos pelas entradas, em seguida, os resultados são somados e comparados a um limiar, conforme mostrado na Equação 1 [33].

$$y = f\left(\sum_{j=1}^{p} w_{ij} x_j - \theta_i\right) \qquad \textbf{(Equação 1)}$$

onde θ_i é o limiar do neurônio i, x_i o seu estado, e a função f do MCP original (o usado no *Perceptron*) é a função *sgn()*, definida pela Equação 2 [33].

$$\mathrm{sgn}(t) = \begin{cases} +1, & t \geq 0 \\ -1, & t \lhd 0 \end{cases} \qquad \textbf{(Equação 2)}$$

Outras funções de limiarização são utilizadas em RNAs, sendo uma delas a função sigmóide logística, conforme mostrado na Figura 2, e definida pela Equação 3. A função sigmóide logística é uma função matemática que produz uma curva em forma de “S”, e é utilizada nas RNAs para introduzir não-linearidade no modelo. Alguns modelos de RNAs utilizam esta função, onde podemos destacar o modelo de Hopfield utilizado neste trabalho.

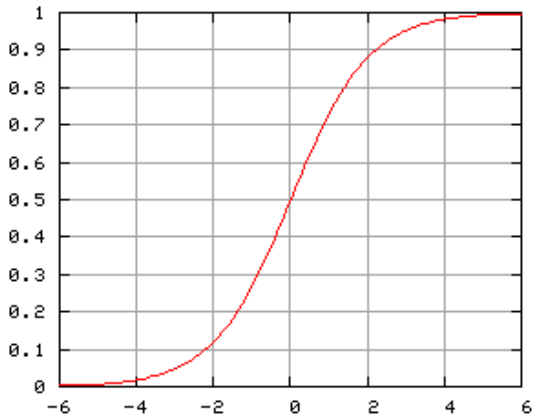

Figura 2. A curva da função sigmóide logística.

$$P(t) = \frac{1}{1 + e^{-t}} \qquad \textbf{(Equação 3)}$$

2.1 Estrutura Básica

A área de Redes Neurais passou por um período de crise de mais de 10 anos, devido a Minsky e Papert [34] exporem as limitações do modelo Perceptron. O ressurgimento pelo interesse na área foi graças a John Hopfield, um cientista norte-americano que propôs uma nova arquitetura para redes neurais em 1982, que hoje é mais conhecida como Rede Neural de Hopfield (HNN - *Hopfield Neural Network*) [18]. Até então havia pouco interesse no estudo de redes neurais realimentadas, porque havia pouco conhecimento no processo de treinamento, e existia dificuldade em analisá-las. Uma HNN é um modelo matricial não-linear recorrente, ou seja, para cada nó da rede existe uma conexão com todos os outros nós da rede. A Figura 3 mostra o modelo de Hopfield. A conexão entre um nodo e ele mesmo é inexistente na arquitetura tradicional de Hopfield. Possui uma única camada de neurônios. Os valores das entradas são os mesmos valores das saídas dos neurônios com um atraso no tempo.

Os neurônios da rede são similares aos da rede *Perceptron*, cada nó tem um limiar (*threshold*), um somatório da multiplicação dos pesos pelas entradas e uma função de limiarização.

No modelo original de Hopfield, os valores das saídas (e conseqüentemente das entradas) são binários (0 ou 1), ou bipolares (-1 e +1), ou seja, os neurônios são do tipo MCP [33]. Hopfield em seguida mostrou que é possível preservar as características do modelo original quando as saídas dão respostas contínuas [35].

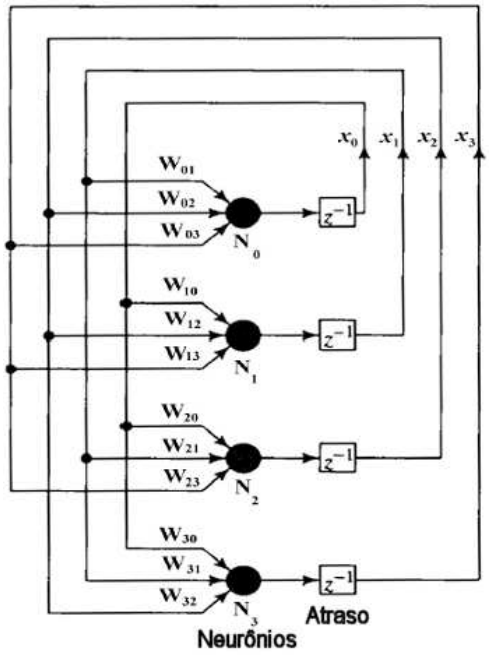

Figura 3. Exemplo de uma arquitetura de uma Rede Neural de Hopfield.

A atualização dos valores das saídas dos neurônios é feita de forma assíncrona e pode ser feita de forma aleatória, baseado na Equação 1. A saída da rede será o valor de todos os nós quando um estado estável for alcançado, ou seja, não haverá mais mudanças nos valores das saídas dos nodos.

O estado da rede é definido pelos valores das saídas. Um valor de energia é associado a cada estado da rede. Esse valor de energia para problema de memória associativa pode ser calculado a partir da Equação 4 [18]. Esses estados seguem para estados atratores que são estados de valor mínimo de energia. A criação dos estados atratores é feita no processo de treinamento da rede.

$$E = -\frac{1}{2}\left(\sum_i \sum_j w_{ij} x_i x_j \right)_{i \neq j} \qquad \textbf{(Equação 4)}$$

Onde *w* representa o peso, *x* o valor da entrada, *i* e *j* os índices das entradas e *E* representa o valor de energia.

2.2 Redes de Hopfield como Memória Associativa

Hopfield, investigando algumas propriedades do modelo de RNA por ele projetado, conseguiu demonstrar que a rede era capaz de “armazenar” informações baseado em alguns estados da rede. Ou seja, observa-se que a RNA de Hopfield possui a capacidade de memorizar. A forma de recuperação da informação era baseada no conteúdo das mesmas, como uma memória associativa, sendo essa uma característica do cérebro humano.

O armazenamento das informações é feito de forma distribuída na rede. Não existe o conceito de endereçamento que é utilizado nas memórias dos computadores digitais. A recuperação da informação em redes de Hopfield é feita através da execução da rede, Durante a execução a rede evolui para um dos estados atratores, diferentemente dos computadores digitais em que é feito por endereçamento.

O acesso às informações é feito com base no conteúdo das mesmas, ou seja, com fragmentos da informação a ser buscada, pode-se obter a informação completa.

De acordo com o modelo de Hopfield, a recuperação de uma informação armazenada na rede é feita atravez da inicialização da rede com um estado que representa uma porção conhecida da informação buscada. Em seguida com a evolução da execução da rede, os estados da rede convergem para um estado atrator (um dos estados finais), que representa um item armazenado na memória. Um certo nível de ruído é tolerado na recuperação da informação, esse ruído depende também da quantidade de informações armazenada na rede, devido ao fato que, quanto mais informações são adicionadas à rede, mais estados atratores vão existir e conseqüentemente mais bacias atratoras, fazendo com que a probabilidade da chave de busca ficar localizada em uma bacia que não é a bacia referente ao item de memória procurado aumente. A Figura 4 ilustra uma HNN com alguns estados atratores e suas bacias atratoras. Os pontos são estados atratores que são os itens armazenados, a seta representa o caminho que a rede segue do estado inicial (a chave de busca da informação) até o estado final. A região onde fica localizada o conjunto de setas que seguem para o mesmo ponto representa uma bacia atratora.

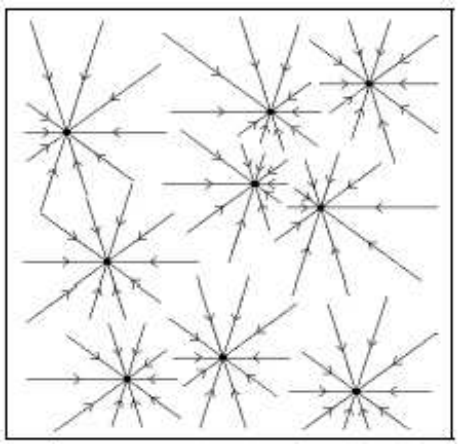

Figura 4. Exemplo de uma HNN com estados atratores armazenados.

Neste projeto nós desenvolvemos uma ferramenta de simulação em Java que armazena informações e recupera essas informações baseadas em parte da informação original. O simulador é baseado em redes de Hopfield e seu objetivo foi preparar o simulador do algoritmo de roteamento que é o principal objetivo deste trabalho. Estados atratores são adicionados à rede a partir da imagem desenhada no simulador da qual desejamos memorizar. Cada pixel é associado a um neurônio da HNN, um pixel preto tem o valor –1 e o pixel branco o valor +1.

O processo de treinamento consiste em modificar a matriz de pesos baseado nos valores dos pixels, fazendo assim a criação de um novo estado atrator.

O processo de recuperação consiste em fornecer parte da imagem a ser recuperada, ou a imagem original acrescida de ruído. A Figura 5 ilustra um exemplo de imagens originais acrescida de 10% de ruído, e a Figura 6 mostra as imagens após a execução do simulador.

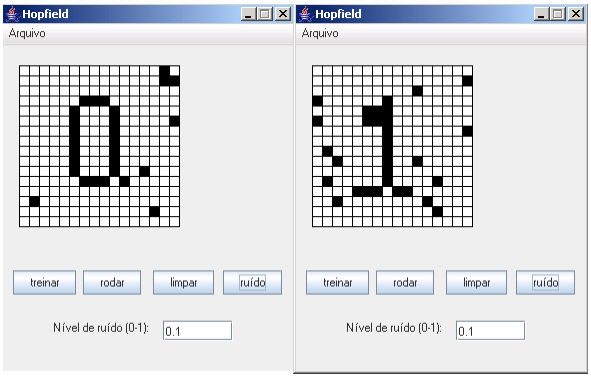

Figura 5. Simulador de Memória Associativa com exemplos de imagens com 10% de ruído a serem recuperadas.

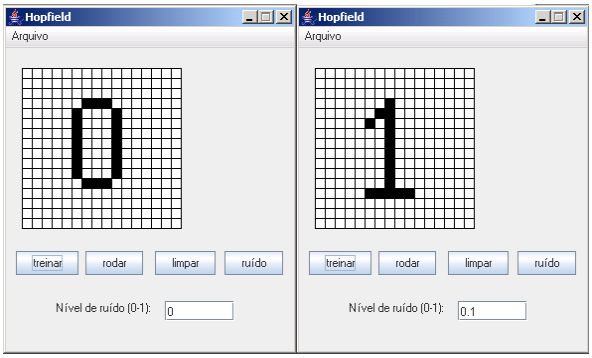

Figura 6. Simulador de Memória Associativa com as respectivas imagens recuperadas.

2.3 Redes de Hopfield para Otimização

As Redes Neurais são uma alternativa para solucionar problemas de otimização de sistemas. Embora a computação tradicional seja utilizada pra resolver esse tipo de problema, essa abordagem tem relativa ineficiência computacional (tempo computacional, memória) e existe uma necessidade crescente de utilizar métodos alternativos que utilizem arquiteturas de processamento paralelas e adaptativas. RNAs podem ser eficientes na aplicação desse tipo de problema, sendo suas vantagens: o paralelismo intríseco das Redes Neurais, a facilidade de implementação em *hardware*, a velocidade de computação utilizando elementos simples de processamento. Diversos problemas de otimização podem ser solucionados por Redes Neurais, dentre os quais se destacam:

- problema de otimização combinatorial ou Problema do Caixeiro Viajante (TSP - *Traveling Salesman Problem*) [5];
- problemas de programação linear [36];
- problemas de programação não-linear [37] ;
- problemas de roteamento em redes de comunicação [38];
- problemas de otimização irrestrita [40] ;

O primeiro modelo introduzido por Hopfield empregava neurônios de apenas dois estados. Este modelo foi usado para desenvolver memórias associativas. Posteriormente,

Hopfield introduziu uma versão modificada denominada modelo contínuo que empregava uma função de transferência não-linear para os neurônios, ou seja, os valores de saída do neurônio eram contínuos.

Hopfield mostrou que a partir do modelo contínuo [35] é possível resolver problemas de otimização [5].

Hopfield e Tank utilizaram o modelo contínuo para resolver o Problema do Caixeiro Viajante [5]. O objetivo é encontrar o estado que minimiza uma função-custo, a qual fornece uma medida de desempenho para o sistema.

As redes neurais utilizam parâmetros de penalidade para resolver problemas de otimização [37]. Os pontos de equilíbrio são as soluções do problema, eles são obtidos a partir dos parâmetros de penalidades que são responsáveis pela convergência da rede. Os valores destas constantes devem ser grandes, porém a escolha destes valores é difícil, já que é realizada através de técnicas de tentativa e erros, o que pode acarretar alto esforço computacional. A qualidade da solução final também depende da escolha destes parâmetros. Outro problema relacionado aos valores dos parâmetros é em relação ao processo de convergência para os pontos de equilíbrio. A partir da obtenção destes parâmetros a rede tende a convergir aos pontos de equilíbrio que representam as possíveis soluções para o problema.

Capítulo 3

Roteamento de Redes

Redes de Computadores são formados por um conjunto de *hosts* capazes de trocar informações e compartilhar recursos. A comunicação em uma rede é feita por interligações do meio físico e um conjunto de protocolos para trasmissão de dados, podendo ser organizada em topologias diversas.

O exemplo de rede mais conhecida atualmente é a Internet, que é um conglomerado de redes comerciais, governamentais, educacionais, e outras redes de computadores que compartilham o mesmo conjunto de protocolos, nomes, e endereços reservados controlados pelo ICANN (*Internet Corporation for Assigned Names and Numbers*) [41]. O modelo de referência utilizado pela Internet para comunicação dos dispositivos é o "TCP/IP". Este modelo é assim chamado porque os protocolos TCP (*Transmission Control Protocol*) [42] e o IP (*Internet Protocol*) [43] são dois dos seus fundamentais componentes.

Outro modelo de referência importante no universo das redes é o modelo OSI (*Open Systems Interconnection*) [44]. Este modelo se baseia em uma proposta desenvolvida pela ISO (*International Standards Organization*) [45]. O modelo OSI é dividido em sete camadas: a camada física, a camada de enlace de dados, a camada de rede, a camada de transporte, a camada de sessão, a camada de apresentação e a camada de aplicação.

A principal função da camada de rede é determinar uma rota ou caminho para que pacotes do *host* de origem cheguem ao *host* de destino, tanto para redes que forneçam o serviço de datagrama, quanto para redes que forneçam o serviço de Circuito Virtual (CV). A responsabilidade do processo de roteamento nas redes de computadores é do protocolo de roteamento da camada de rede.

O algoritmo de roteamento é o motor do protocolo de roteamento. Ele é de crucial importância no processo de roteamento e no projeto da camada de rede. O objetivo do algoritmo é simples: em uma rede composta de roteadores e enlaces, o algoritmo deve encontrar o mais adequado caminho entre a origem e o destino. Este caminho pode depender de diversos fatores, como por exemplo:

- capacidade do enlace;
- carga;
- distância física do enlace;

Algumas características são desejáveis para um algoritmo de roteamento, como:

- correção;

- simplicidade;
- robustez;
- estabilidade;
- eqüidade;
- otimização.

Os algoritmos podem ser classificados como globais ou descentralizados. E também podem ser classificados como estáticos ou dinâmicos.

- Algoritmos de roteamento global: a principal característica dessa classe é que o algoritmo deve ter informações completas sobre custos dos enlaces e topologias para que ele calcule as rotas, podendo ser concentrado em um local (centralizado) ou podendo ser executado em múltiplos locais. Na prática estes algoritmos são chamados de *algoritmos de estado de enlace* (LS – *link-state*) [46].

- Algoritmo de roteamento descentralizado: estes algoritmos calculam as rotas de forma iterativa e distribuída. Os nós não têm informações completas sobre a rede. Inicialmente os nós possuem apenas informações referentes aos enlaces diretamente conectados. Um nó determina as rotas através de troca de informações com os nós vizinhos. Um algoritmo com essa característica é o *algoritmo vetor-distância* (DV – *distance-vector*) [46].

- Algoritmos de roteamento estático: nestes algoritmos os custos dos enlaces mudam de forma lenta, muitas vezes por intervenção humana.

- Algoritmos de roteamento dinâmico: nestes algoritmos os custos dos enlaces mudam conforme a carga da rede e a topologia da rede. Estes algoritmos sofrem de alguns problemas, como laços de roteamento e oscilações nas rotas.

3.1 Algoritmo(s) de roteamento *link-state*

Neste tipo de algoritmo os custos dos enlaces e a topologia da rede são conhecidos, ou seja, o algoritmo descobre as rotas a partir de todos os dados da rede disponíveis. Todos os nós enviam suas informações de custos para todos os outros nós. As informações são enviadas por *broadcast*. São enviadas as informações dos custos dos nós diretamente conectados a um determinado nó. Desta forma, com as recepções dessas informações, cada nó ficará a par da topologia de toda a rede. Com isso cada nó terá uma visão idêntica e completa da rede. Com estas informações é possível para cada nó rodar o algoritmo *link-state* e encontrar o caminho de menor custo para os outros nós.

Um dos principais algoritmos *link-state* é o algoritmo de Dijkstra [1], nome do seu inventor. Será utilizada a rede mostrada na Figura 7 para auxílio na explicação do algoritmo de Dijkstra.

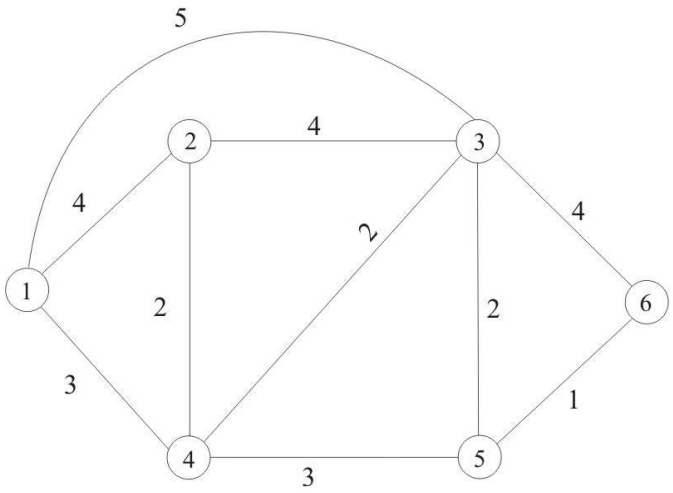

Figura 7. Uma rede hipotética. Os números representam os custos dos enlaces.

Neste exemplo chamaremos a origem de 1, o algoritmo calculará o menor caminho até os outros nós da rede. Após a *k*-ésima iteração, o algoritmo encontra os caminhos de menor custo para *k* nós de destino. O algoritmo *link-state* funciona da seguinte maneira:

- $c(i, j)$: custo do enlace do nó *i* ao nó *j*. $c(i, j) = \infty$ é dado quando o *link* é inexistente entre *i* e *j*.
- *D(v)*: custo do caminho entre o nó origem e o destino *v*. No fim da execução do algoritmo é o menor custo.
- *p(v)*: nó anterior (vizinho de *v*) ao longo do caminho de menor custo desde a fonte até *v*.
- *N*: conjunto de nós cujo caminho de menor custo a partir da fonte é positivamente conhecido.

O algoritmo *link-state* consiste em uma etapa de inicialização seguida de um loop. Na etapa de inicialização é atribuída a *N* o nó origem e então é atribuído a *D(v)* o valor do custo do enlace de cada nó *v*. Se o nó for adjacente, *D(v)* será o custo do enlace de 1 até *v*. Senão *D(v)* será ∞. A etapa de *loop* busca minimizar o valor de *D(v)* para cada nó. O número de vezes que o *loop* é executado é igual ao número de nós da rede. Ao terminar, o algoritmo terá calculado os caminhos mais curtos desde o nó fonte até todos os outros nós da rede.

Algoritmo *link-state:*

```
Inicialização
      N = {1}
      para todos os nós v
            se o v é adjacente a 1
                  então D(v) = C(1,v)
                  senão D(v) = ∞

Loop
      encontre w ∉ N, tal que D(w) é mínimo
      adicione w a N
      atualize D(v) para todo v adjacente a w e não em N:
            D(v) = min( D(v), D(w) + c(w,v) )
```

```
      /*o novo custo para v é o velho custo
      ou o custo do menor caminho para w mais o custo de w
      para v */
até todos os nós em N
```

Neste exemplo chamaremos a origem de 1, o algoritmo calculará o menor caminho até os outros nós da rede. Após a *k*-ésima iteração, o algoritmo encontra os caminhos de menor custo para *k* nós de destino.

Tabela 1. Execução do algoritmo *link-state* no exemplo citado.

Estágio	N	D(2), p(2)	D(3), p(3)	D(4), p(4)	D(5), p(5)	D(6), p(6)
0	1	4, *1*	5, *1*	3, *1*	∞	∞
1	14	4, *1*	5, *4*		6, *4*	∞
2	142		5, *4*		6, *4*	∞
3	1423				6, *4*	9, *3*
4	14235					7, *5*
5	145236					

A Tabela 1 resume o funcionamento do algoritmo. As linhas da tabela representam os estágios de execução do algoritmo. A primeira coluna enumera os estágios, a segunda coluna mostra a evolução do conjunto *N*, e as colunas seguintes representam a evolução das informações referentes ao menor caminho para cada nó.

No início, a partir do nó 1, os menores caminhos aos vizinhos (2, 3, 4) diretamente conectados são 4, 5 e 3 respectivamente. Os custos para 5 e 6 são infinitos porque não existe conexão direta com *1*. No estágio 0, descobre-se que o menor custo foi o do nó 4 que tem o valor 3, então ele é adicionado ao conjunto *N*. Na segunda iteração, o nó 5 muda de valor, porque existe uma conexão com 4, o valor do custo é 6 porque do nó 1 ao nó 4 o custo é 3 e do nó 4 ao nó 3, também é 3. Na próxima iteração 2 é adicionado ao conjunto *N* e o ciclo se repete. O processo é encerrado quando todos os nós são adicionados ao conjunto *N*. No final teremos para cada nó, seu custo total do menor caminho e qual é o nó predecessor. O nó predecessor também terá um outro nó predecessor, dessa forma é possível determinar o caminho.

Na primeira iteração, pesquisa-se todos os nós *n* para encontrar o nó *w*, que não está em *N*. Na segunda iteração verifica-se *n-1* para encontrar outro nó *w*. Na terceira iteração, *n-2* nós. Conclui-se que o número total de nós que são pesquisados é $n(n + 1)/2$, então a complexidade deste algoritmo, no pior caso, é da ordem de *n* ao quadrado: $O(n^2)$. Porém, existem otimizações que reduzem esta complexidade [46].

Protocolos de roteamento por estado de enlace são amplamente utilizados em redes de computadores, sendo dois deles o IS-IS (*Intermediate System – Intermediate System*) [47] e o OSPF (*Open Shortest Path First*) [48].

O OSPF é um protocolo de roteamento de hierarquia do tipo IGP (*Interior Gateway Protocol*), baseado no algoritmo de *Dijkstra*, O OSPF foi descrito inicialmente em 1989 com o objetivo de substituir o protocolo RIPv1 (*Routing Information Protocol*) [49] para uso na Internet. É um dos protocolos de roteamento mais empregado, sendo suportado pela maioria dos roteadores e servidores.

3.2 Algoritmo(s) de roteamento *distance-vector*

Estes algoritmos têm como propriedade serem iterativos, assíncronos e distribuídos. Cada nó da rede depende das informações dos nós vizinhos para realizar seus cálculos e em seguida enviar os

resultados para os nós vizinhos, por isso são distribuídos. O processo deixa de ser executado quando não há mais informação a ser trocada entre nós vizinhos, ou seja, são algoritmos com *auto-extinção*, dando a característica de iteratividade. Os nós não precisam operar em sincronia, ou seja, são assíncronos. Um algoritmo amplamente utilizado é o de Bellman-Ford [50][51]. Porém esse algortimo tem alguns problemas como: baixa velocidade de convergência, *loops* de roteamento e contagem ao infinito.

O algoritmo vetor-distância cria uma convergência vagarosa ou contagem ao infinito, problema em que as inconsistências surgem devido ao roteamento atualizar as mensagens propagadas vagarosamente através da rede.

A Figura 8 ilustra um exemplo de uma rede onde a rede *1* torna-se inoperante. As informações sobre a rede *1* ficam circulando na rede infinitamente enquanto o custo do enlace aumenta.

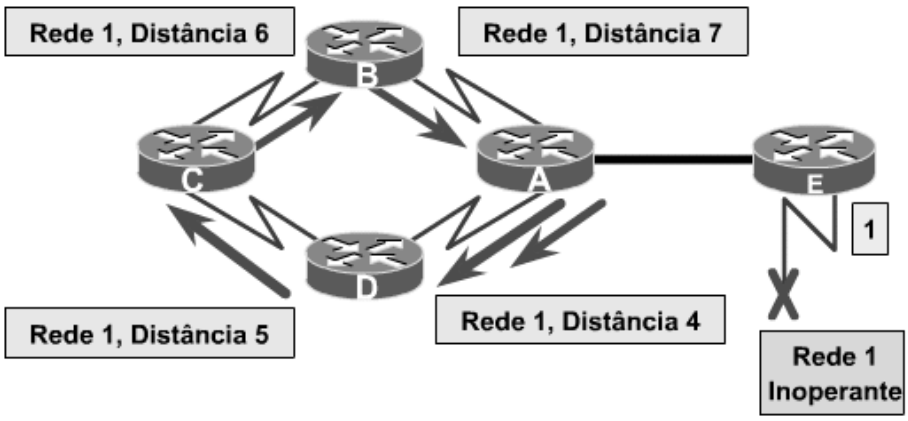

Figura 8. Exemplo de uma rede com o problema da contagem ao infinito.

Exemplos de protocolos de roteamento que utilizam estes algoritmos são: RIPv1, RIPv2 [52] e IGRP (*Interior Gateway Routing Protocol*) [53]. O EGP (*Exterior Gateway Protocol*) [54] e o BGP (*Border Gateway Protocol*) [55]. Esses algoritmos do tipo vetor-caminho (*path-vector*) que por às vezes utilizam o algoritmo *distance-vector* para solucionar o problema da contagem ao infinito. O RIPv1, o RIPv2 e o IGRP são utilizados dentro de um Sistema Autônomo (SA). O BGP, e o EGP (já obsoleto), são protocolos de roteamento interdomínios, utilizados para comunicação entre Sistemas Autônomos. Foram criados para uso nos roteadores principais da Internet.

3.3 Comparação entre os algoritmos de roteamento *link-state* e *distance-vector*

Nesta seção iremos comparar as duas classes de algoritmos explicados anteriormente, baseado em alguns de seus atributos.

- **Complexidade da mensagem**: Algoritmos *link-state* requer em que cada nó saiba os custos do enlaces dos outros nós. Fazendo com que sejam enviadas da ordem de *O(nE)* mensagens, onde *n* é o número de nós da rede e *E* o número de enlaces. Algoritmos

distance-vector requer em troca de mensagens entre vizinhos diretamente conectados a cada iteração. As mensagem para estas duas classes de algoritmos são muito diferentes, dificultando uma comparação.

- **Velocidade de convergência**: Algoritmos *link-state* têm a complexidade da ordem de $O(n^2)$. A velocidade de convergência depende diretamente da complexidade. Algoritmos *distance-vector* sofre de problemas de convergência por causa de *loops* de roteamento, também pelo problema de contagem ao infinito. Algoritmos *link-state* geralmente convergem mais rápido que algoritmos *distance-vector*.

- **Robustez**: Nos algoritmos *link-state*, caso um nó propague pela rede informações incorretas, os outros nós que receberem estas informações ou descartarão as mesmas ou aceitarão e farão os cálculos das rotas de forma incorreta. Porém não propagarão estas informações, fornecendo um certo grau de robustez. Já nos algoritmos *distance-vector* caso um nó anuncie incorretamente informações sobre custos de enlaces para seus vizinhos a informação será propagada por toda a rede. Assim os algoritmos *link-state* são mais robustos.

De forma geral, os algoritmos *link-state* e *distance-vector* as duas classes de algoritmos possuem suas vantagens e desvantagens, tanto que eles são utilizados hoje em dia tanto na Internet como nas outras redes.

Capítulo 4

Aplicações de Redes de Hopfield para roteamento de Redes de Computadores

Este capítulo destina-se a explicar o uso de HNN para resolver o problema do menor caminho usando o algoritmo de Ali e Kamoun [2]. Também neste capítulo será apresentada uma breve descrição do simulador desenvolvido, um dos resultados deste trabalho. A nova abordagem para o algoritmo de Ali e Kamoun que é outro objetivo desta monografia será explicada.

Rauch e Winarske [38] iniciaram o uso de HNN para resolver o problema do menor caminho, porém esta proposta exigia um conhecimento prévio da topologia da rede, e esta topologia não poderia ser modificada. Para superar esta limitação, Ali e Kamoun propuseram um novo algoritmo adaptável, onde a matriz de pesos carregasse apenas a informação da convergência. A informação sobre o custo do enlace e a topologia é adicionada na polarização (*bias*) como mostrado na Figura 9. Artigos científicos adicionais relatam técnicas para evitar laços e problemas envolvidos na convergência do algoritmo [7][8]. Entretanto estes algoritmos são baseados na mesma equação diferencial proposta por Ali e por Kamoun [2].

4.1 Redes Neurais de Hopfield para Roteamento em Redes de Comunicação

O diagrama de bloco da Rede Neural de Hopfield **(**HNN**)** está mostrado na Figura 9. Os elementos de processamento são os neurônios. Cada saída de cada neurônio é conectada às entradas de todos os neurônios restantes através das sinapses de peso. Para resolver problemas de roteamento em redes de comunicação, Ali e Kamoun propuseram que cada enlace na rede de comunicação entre dois nós adjacentes deve ter um neurônio associado. Pro exemplo, um enlace do nó *x* ao nó *i* é associado um neurônio que tem a descrição *xi*. A saída do neurônio V_{xi} depende da entrada U_{xi}, de acordo com função sigmóide expressa na Equação 5. Observe que a entrada de cada neurônio corresponde à soma de todas as saídas dos neurônios da HNN multiplicado por um fator representado pela matriz $T_{xi,yj}$ adicionada a uma polarização externa I_{xi}. O elemento $T_{xi,yj}$

representa os pesos sinápticos que conectam a saída do neurônio *yj* ao ponto da soma na entrada do neurônio *xi*.

$$V_{xi} = \frac{1}{1+e^{-\lambda_{xi}U_{xi}}}, \quad \forall(x,i) \in \overline{N} \times \overline{N},\ x \neq i \qquad \textbf{(Equação 5)}$$

O parâmetro λ_{xi} determina o tempo de computação para convergência e exatidão do algoritmo. Nas simulações foi usado o mesmo λ_{xi} para todos os neurônios na HNN. Conseqüentemente, nós chamaremos este parâmetro de λ no resto do trabalho, ao invés de λ_{xi}. O comportamento da função sigmóide logística é mostrado na Figura 10 para diferentes valores de λ. Com um valor alto de λ a função tende para uma função degrau.

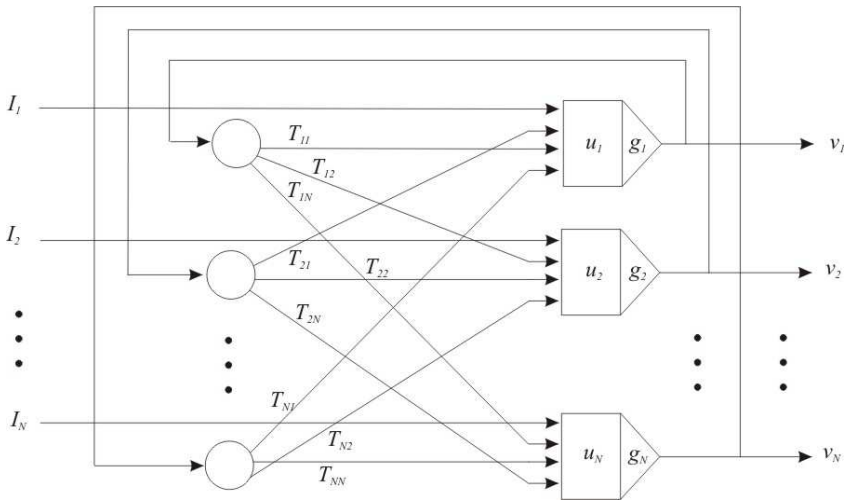

Figura 9. Configuração da Rede Neural de Hopfield.

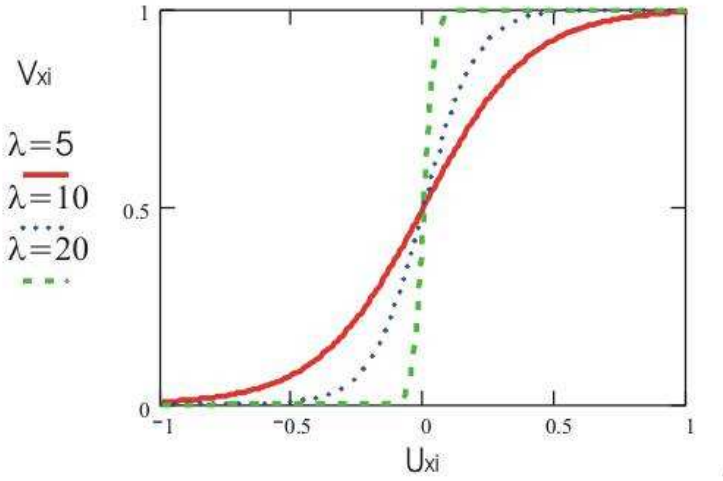

Figura 10. Comportamento de uma função Sigmóide Logística para diferente valores λ.

Para cada enlace da rede existe um valor não-negativo associado C_{xi}. O objetivo da HNN é achar caminho que minimiza o custo de um nó fonte *s* para um nó destino *d* através da rede de comunicação. Assim, a HNN deve indicar um caminho dirigido em uma seqüência de nós ordenados conectando *s* a *d*, assim a soma de todos os custos que conectam estes nós fornece o menor custo possível. O caminho que fornece o menor caminho definido como L_{sd}. Em muitos casos, o custo da matriz é simétrico ($C_{xi} = C_{ix}$), porém existem alguns trabalhos que consideram assimétrico o custo da matriz ($C_{xi} \neq C_{ix}$) [4], por exemplo enlaces ADSL (*Asymmetric Digital Subscriber Line*) [39]. Observe que os elementos C_{ii} são nulos porque um nó não pode ser conectado a ele mesmo.

A matriz ρ_{xi} define se o arco *xi* existe na topologia da rede de comunicação usada na simulação. Se o arco *xi* existe então $\rho_{xi} = 0$, senão $\rho_{xi} = 1$. Quando a simulação converge, isto é, quando a mudança de valores de cada saída V_{xi} em uma interação é menor que um critério predefinido, um ajuste é feito em cada saída. Se uma saída tiver um valor maior que 0.5 o valor é ajustado para "1", senão é ajustado para "0", como mostrado na Equação 6. O valor final de V_{xi} define se o arco relacionado ao neurônio *xi* pertence ou não ao menor caminho L_{sd}.

$$V_{xi} = \begin{cases} 1, arc_{xi} \in L_{sd} \\ 0, otherwise \end{cases} \qquad \textbf{(Equação 6)}$$

Ao lado disto, cada célula pode ser externamente excitada por uma polarização de entrada I_{xi}. Este sistema de entradas pode ser usado para ajustar o nível geral de excitação da rede inteira e representar os dados reais fornecidos pelo usuário à rede neural. Foi utilizada a seguinte expressão para a polarização [2]:

$$I_{xi} = -\frac{\mu_1}{2} C_{xi}\left(1 - \delta_{xd}\,\delta_{is}\right) - \frac{\mu_2}{2}\rho_{xi}\left(1 - \delta_{xd}\,\delta_{is}\right)$$
$$-\frac{\mu_4}{2} + \frac{\mu_5}{2}\delta_{xd}\,\delta_{is} \qquad \textbf{(Equação 7)}$$
$$\forall\left(x \neq i\right), \forall\left(y \neq i\right)$$

onde δ é a função de Kronecker e μ_1, μ_2, μ_4 e μ_5 são constantes.

A matriz de sinapses $T_{xi,yj}$ e a função de energia da HNN é descrita em [2]:

$$T_{xi,yj} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy} \qquad \textbf{(Equação 8)}$$

$$E = \frac{\mu_1}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} C_{xi} V_{xi} + \frac{\mu_2}{2} \sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} \rho_{xi} V_{xi}$$
$$+ \frac{\mu_3}{2} \sum_{x=1}^{n} \left\{ \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi} - \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{ix} \right\}^2 \qquad \textbf{(Equação 9)}$$
$$+ \frac{\mu_4}{2} \sum_{x=1}^{n} \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi}\left(1 - V_{xi}\right) + \frac{\mu_5}{2}\left(1 - V_{ds}\right)$$

onde *E* é a energia da HNN e μ_3 é uma constante.

Estes parâmetros têm função específica na função de energia: μ_1 minimiza o custo total; μ_2 impede que enlaces inexistentes estejam incluídos no trajeto escolhido; μ_3 é zero para cada nó no caminho válido, μ_4 força a HNN para convergir a um estado estável e é introduzido para assegurar-se de que a fonte e os nós de destino perteçam à solução. Ahn *et al* propôs outros termos para evitar os laços [7].

Conseqüentemente, se o sistema for estável no sentido de Liapunov, então as iterações conduzirão para uma menor mudança nas saídas. Como conseqüência, após um número conveniente de iterações, a HNN alcança alguns mínimos de energia do sistema. Ali e Kamoun resolvem o sistema usando a equação diferencial descrita abaixo.

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \sum_{y=1}^{n} \sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj} V_{yj} + I_{xi} \qquad \textbf{(Equaçao 10)}$$

Como a soma do segundo e do terceiro termo representa a variação de energia da HNN, pode-se reescrever como segue [2]:

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{dE}{dV_{xi}} \qquad \textbf{(Equaçao 11)}$$

Ali e Kamoun usam o método de Runge-Kutta para resolver a equação. Para simplificar a acelerar a resolução, foi proposto neste trabalho um nova abordagem baseada na equação diferença discreta [21].

4.2 Abordagem de HNN baseada na Equação Diferença de Energia Finita

Alguns autores propuseram resolver problemas através de HNN's usando as Equações discretas de energia no tempo [56][57]. Entretanto, nenhum destes autores a aplicou ao problema de roteamento em redes de comunicações. Assim, foi adaptada a formulação discreta do tempo a HNN modelada para resolver o problema do roteamento. Acredita-se que é possível acelerar o cálculo de rotas. Consequentemente, em vez de resolver sistemas usando uma equação diferencial que requer elevado custo computacional (como o método de Runge Kutta, isto é, Equação 11), neste trabalho nós propomos uma aproximação simples baseada na referência discreta do tempo [21]. A equação usada para calcular o valor da nova entrada dos neurônios é mostrada abaixo.

$$U_{xi}[n+1] = U_{xi}[n] - AU_{xi}[n-1] - BU_{xi}[n-2] + C\left(\sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + I_{xi}[n]\right) \quad \textbf{(Equação 12)}$$

onde $U_{xi}[n+1]$ é a próxima entrada do neurônio *xi* calculada com base nos valores de saída de todos os neurônios da rede $V_{yj}[n]$, na polarização externa $I_{xi}[n]$ e suas próprias entradas em instantes precedente $U_{xi}[n]$, $U_{xi}[n-1]$ e $U_{xi}[n-2]$. *A*, *B* e *C* são constantes.

4.3 Descrição do Algoritmo e Ferramenta de Simulação

Neste trabalho foi desenvolvida uma ferramenta de simulação em Java [58] para calcular a rota baseada em HNN. A tela principal do simulador desenvolvido está mostrada na Figura 11.

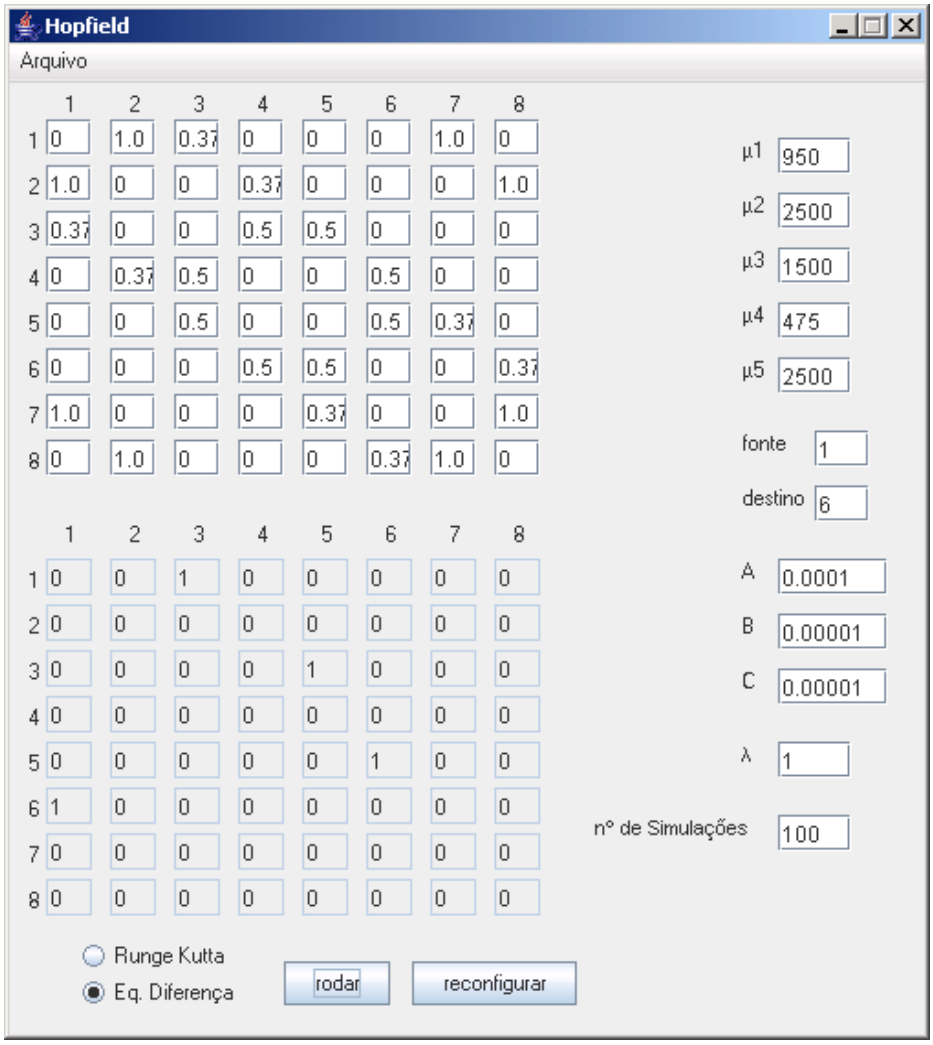

Figura 11. Simulador de Redes Neurais de Hopfield para problema do menor caminho.

Na Figura 11 a matriz superior de campos de entradas representa os custos do enlace de um dado nó *x* até um nó *i*; os campos μ_1, μ_2, μ_3, μ_4, μ_5, *A, B, C* e λ são usados para inserir os valores dos respectivos parâmetros; o botão *reconfigurar* tem a função de recalcular a matriz de pesos $T_{xi,yj}$ caso os valores dos parâmetros $\mu's$ sejam alterados; os *radio buttons*: *Runge Kutta* e *Eq. Diferença* são usados para a seleção do método que a Rede de Hopfield irá encontrar o menor caminho.

O simulador funciona de duas formas: encontrar um caminho dado um nó fonte e um nó destino, e a outra é para simular uma certa quantidade de execuções com nós fontes e destinos aleatórios. Na execução padrão, os campos que foram citados anteriormente devem estar configurados e devem ser escolhidos o nó fonte e o nó destino (configurados nos campos *fonte* e *destino* respectivamente), e então o botão *rodar* fará com que o simulador ache o menor caminho, que é mostrado na matriz inferior. Valores *1* na matriz inferior indicam que do nó *x* ao nó *i* o enlace da rede de comunicação fará parte do menor caminho. No modo de simulação, o campo *nº de Simulações* deve conter a quantidade de execuções da simulação, então no menu *Arquivo -> Simulação* fará as simulações e os resultados serão mostrados na tela. Dentre os resultados mostrados temos:

- tempo de simulação por Dijkstra e pelos métodos baseado em HNN;
- número de iterações para os métodos e percentual de acerto para os métodos baseados em HNN, entre outras informações.

No fluxograma mostrado na Figura 12 podemos ver o processo de funcionamento do simulador. Na primeira etapa os parâmetros μ_1, μ_2, μ_3, μ_4 e μ_5 são usados para determinar a matriz de pesos. Após isto, os parâmetros das simulações são requeridos. Usando estes dados o *software* calcula a topologia e a matriz de polarização. Em seguida, os neurônios são inicializados: as entradas dos neurônios são ajustadas com um pouco de ruído para acelerar a convergência.

Tabela 2. Parâmetros usados nas simulações e seus valores padrões.

PARAMETROS	VALORES (PADRÕES)
μ_1	950
μ_2	2500
μ_3	1500
μ_4	475
μ_5	2500
A	0.0001
B	0.00001
C	0.00001
λ	1

Valores padrões dos parâmetros da simulação são apresentados na Tabela 2. Valores elevados para o parâmetro da função logística conduzem a uma convergência mais rápida do algoritmo, porém tais valores podem conduzir a erros na determinação na rota.

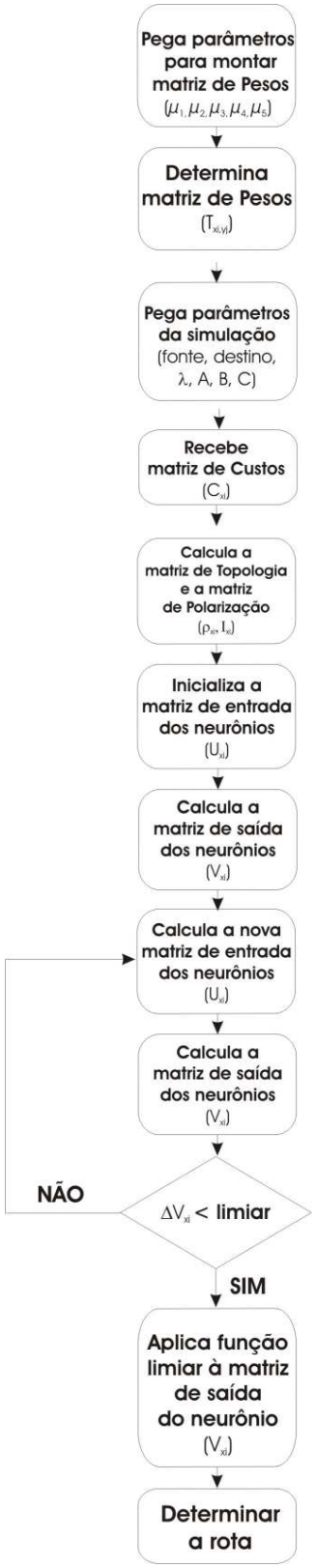

Figura 12. Fluxograma do algoritmo de tempo discreto para Redes Neurais de Hopfield para roteamento em redes de comunicação.

Capítulo 5

Experimentos

Este capítulo se destina a relatar um conjunto de simulações executadas para comparar o método de roteamento proposto neste trabalho com o método proposto por Ali e Kamoun. Os algoritmos de roteamento baseados na rede neural de Hopfield são comparados com o algoritmo de Dijkstra. Foram consideradas três topologias de redes de computadores em nossas simulações. As redes são mostradas na Figura 13, Figura 14 e Figura 15 respectivamente.

Para cada conjunto de simulações está-se interessado em comparar os três métodos de roteamento. Como parâmetros de desempenho utilizamos:

- número de rotas ótimas encontradas;
- tempo de simulação.

Para os métodos baseados em HNN compara-se o número de iterações necessárias para a convergência.

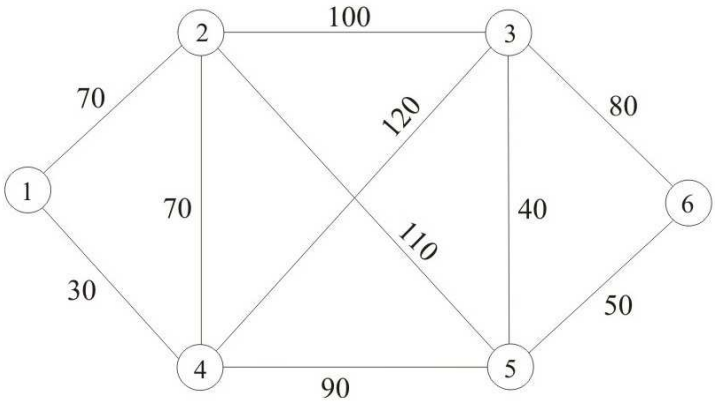

Figura 13. Rede 1: uma das topologias de redes de computadores usadas nas simulações.

Um conjunto de simulações para uma dada topologia de rede de computadores consiste em um número de simulações que consideram os três algoritmos de roteamento. Para cada simulação, os nós de fonte e destino são selecionados aleatoriamente. Em seguida, o algoritmo de

Dijkstra é usado para obter o menor caminho entre os nós fonte e destino. Finalmente, tanto a HNN-RK quanto a HNN-DE proposta neste trabalho são executadas independentemente para obter o menor caminho. Os caminhos obtidos pelos métodos de HNN são comparados com os caminhos computados pelo algoritmo de Dijkstra**.**

Um conjunto de simulações é executado considerando os mesmo valores de parâmetros das HNNs. Após o conjunto de simulação ser executado, obtêm-se o tempo total de simulação para cada método de roteamento usando HNN. Foram obtidos também a média e o desvio padrão do número de iterações necessárias para a HNN-DE e a HNN-RK. Finalmente, nós computamos, para as HNN's, a porcentagem de simulações em que este método produziu uma rota ótima, isto é, uma rota idêntica àquela produzida pelo algoritmo de Dijkstra para a mesma simulação (isto é, uma simulação usando os mesmos nós da fonte e de destino).

A Tabela 3, 4 e 5 relatam alguns resultados das simulações para a rede de computadores *1*, descritas na Figura 13. Cada linha das Tabelas 3, 4 e 5 apresenta os resultados obtidos para um dado conjunto de simulações. Cada conjunto de simulações compreende 100 simulações com nós da fonte e destino selecionados aleatoriamente. Em todas as simulações, os métodos de roteamento usando HNN utilizaram os valores padrões dos parâmetros μ_1, μ_2, μ_3, μ_4 e μ_5 descritos na Tabela 2. Os valores dos parâmetros *A, B, C* e λ foram variados para analisar sua influência do desempenho.

A Tabela 3 mostra a precisão dos métodos de HNN em comparação aos resultados fornecidos pelo algoritmo de Dijkstra. Os resultados da Tabela 3 mostram que para a maioria de valores de parâmetros os algoritmos de HNN obtêm as mesmas rotas obtidas pelo algoritmo de Dijkstra (isto é, as HNNs obtiveram 100% de precisão).

Tabela 3. Comparação dos métodos em função da precisão, para a Rede 1.

PARÂMETROS HNN	**Exatidão**	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	82.00%

A média e o desvio padrão do número de iterações para a convergência de cada método baseado em HNN para a rede *1* são apresentados na Tabela 4. Pode-se observar que o número das iterações depende significativamente dos valores dos parâmetros. Comparando os melhores resultados obtidos por cada método (464,3 e 332,73) conclui-se que a HNN-DE pode convergir usando menos iterações a HNN-RK.

Tabela 4. Comparação dos métodos em função da convergência, para a Rede 1.

PARÂMETROS HNN	Iterações	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	4163.79 ± 25%	4277.45 ± 23%
A=1e-4, B=1e-5, C=1e-5, λ=2	2569.23 ± 29%	2575.49 ± 25%
A=1e-4, B=1e-5, C=1e-5, λ=5	1265.14 ± 32%	1267.40 ± 31%
A=1e-4, B=1e-5, C=1e-5, λ=10	713.69 ± 33%	721.45 ± 31%
A=1e-4, B=1e-5, C=1e-5, λ=20	**464.93** ± 41%	460.52 ± 39%
A=1e-4, B=1e-5, C=1e-5, λ=50	10157.82 ± 196%	8321.70 ± 219%
A=2e-4, B=2e-5, C=2e-5, λ=10	835.0 ± 40%	487.72 ± 41%
A=3e-4, B=3e-5, C=3e-5, λ=10	778.57 ± 34%	**332.73** ± 38%
A=4e-4, B=4e-5, C=4e-5, λ=10	765.52 ± 34%	2751.17 ± 7%
A=5e-2, B=5e-3, C=5e-3, λ=10	4371.56 ± 26%	29557.71 ± 83%

A Tabela 5 compara os métodos de roteamento em termos do tempo de simulação. Os resultados mostram que, para a maioria de combinações dos parâmetros, a HNN-DE é muito mais rápida do que a HNN-RK. Além disso, o menor tempo da simulação para o método proposto é muito menor que o tempo de simulação obtido através da HNN-RK. Para a rede *1*, o menor tempo de simulação obtido por Runge-Kutta foi 8250ms, sendo que pela HNN-DE foi 2021ms.

Tabela 5. Comparação dos métodos em função do tempo de simulação, para a Rede 1.

PARÂMETROS HNN	Tempo de Simulação (ms)	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	73735	26687
A=1e-4, B=1e-5, C=1e-5, λ=2	45922	16078
A=1e-4, B=1e-5, C=1e-5, λ=5	22782	7921
A=1e-4, B=1e-5, C=1e-5, λ=10	12719	4562
A=1e-4, B=1e-5, C=1e-5, λ=20	**8250**	2875
A=1e-4, B=1e-5, C=1e-5, λ=50	172890	47125
A=2e-4, B=2e-5, C=2e-5, λ=10	14641	3015
A=3e-4, B=3e-5, C=3e-5, λ=10	13797	**2031**
A=4e-4, B=4e-5, C=4e-5, λ=10	13703	16828
A=5e-2, B=5e-3, C=5e-3, λ=10	81422	192015

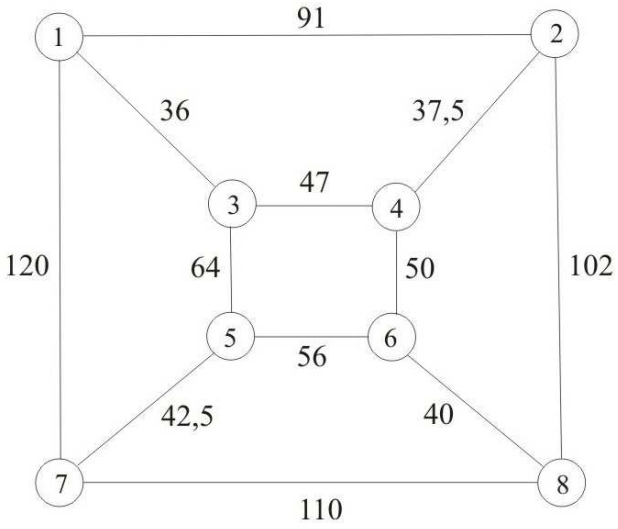

Figura 14. Rede 2: uma das topologias de redes de computadores usadas nas simulações.

Os resultados das simulações para a Rede 2 (Figura 14) são mostrados na Tabela 6, 7 e 8, que seguem a mesma estrutura das Tabelas 3, 4 e 5, respectivamente. Os resultados da simulação para a Rede 3 (Figura 15) são apresentados nas Tabelas 9, 10 e 11. As mesmas observações feitas para a Rede 1 podem ser feitas para as Redes 2 e 3, ou seja, a HNN-DE pode produzir rotas ótimas com tempos muito menores de simulação do que a HNN-RK.

Tabela 6. Comparação dos métodos em função da exatidão (caminhos encontrados), para a Rede 2.

PARÂMETROS HNN	**Exatidão**	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=100	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-4, B=5e-5, C=5e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-3, B=5e-4, C=5e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	95.00%

Tabela 7. Comparação dos métodos em função da convergência, para a Rede 2.

PARÂMETROS HNN	Iterações	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	4433.95 ± 30%	4623.90 ± 31%
A=1e-4, B=1e-5, C=1e-5, λ=2	2705.53 ± 30%	2802.07 ± 31%
A=1e-4, B=1e-5, C=1e-5, λ=5	1420.08 ± 27%	1463.95 ± 27%
A=1e-4, B=1e-5, C=1e-5, λ=10	834.88 ± 29%	858.07 ± 29%
A=1e-4, B=1e-5, C=1e-5, λ=20	495.75 ± 29%	502.75 ± 30%
A=1e-4, B=1e-5, C=1e-5, λ=50	**253.49** ± 30%	260.12 ± 28%
A=1e-4, B=1e-5, C=1e-5, λ=100	4634.77 ± 307%	192.06 ± 34%
A=2e-4, B=2e-5, C=2e-5, λ=10	786.28 ± 31%	490.88 ± 32%
A=3e-4, B=3e-5, C=3e-5, λ=10	830.34 ± 29%	391.53 ± 31%
A=4e-4, B=4e-5, C=4e-5, λ=10	832.49 ± 28%	327.94 ± 28%
A=5e-4, B=5e-5, C=5e-5, λ=10	822.15 ± 29%	267.20 ± 33%
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 ± 25%	**162.08** ± 30%
A=5e-3, B=5e-4, C=5e-4, λ=10	805.86 ± 29%	552.26 ± 900%
A=5e-2, B=5e-3, C=5e-3, λ=10	869.77 ± 28%	4030.47 ± 336%

Tabela 8. Comparação dos métodos em função do tempo de simulação, para a Rede 2.

PARÂMETROS HNN	Tempo de Simulação (ms)	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	259093	77844
A=1e-4, B=1e-5, C=1e-5, λ=2	149906	46266
A=1e-4, B=1e-5, C=1e-5, λ=5	81859	25188
A=1e-4, B=1e-5, C=1e-5, λ=10	46156	15375
A=1e-4, B=1e-5, C=1e-5, λ=20	27360	9000
A=1e-4, B=1e-5, C=1e-5, λ=50	**14359**	4672
A=1e-4, B=1e-5, C=1e-5, λ=100	257406	**2579**
A=2e-4, B=2e-5, C=2e-5, λ=10	45828	8672
A=3e-4, B=3e-5, C=3e-5, λ=10	48500	6766
A=4e-4, B=4e-5, C=4e-5, λ=10	48500	5703
A=5e-4, B=5e-5, C=5e-5, λ=10	48078	4532
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	2703
A=5e-3, B=5e-4, C=5e-4, λ=10	46969	9250

A=5e-2, B=5e-3, C=5e-3, λ=10	51672	69203

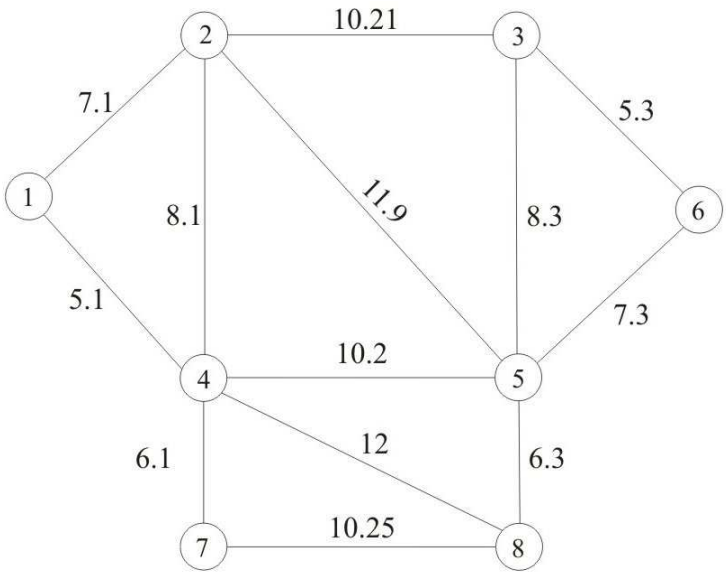

Figura 15. Rede 3: uma das topologias de redes de computadores usadas nas simulações.

Tabela 9. Comparação dos métodos em função da exatidão (caminhos encontrados), para a Rede 3.

PARÂMETROS HNN	**Exatidão**	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	86.00%

Tabela 10. Comparação dos métodos em função da convergência, para a Rede 3.

PARÂMETROS HNN	Iterações	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	4674.93 ± 52%	4742.56 ± 54%

A=1e-4, B=1e-5, C=1e-5, λ=2	2663.18 ± 44%	2692.09 ± 45%
A=1e-4, B=1e-5, C=1e-5, λ=5	1241.41 ± 33%	1246.74 ± 33%
A=1e-4, B=1e-5, C=1e-5, λ=10	752.1 ± 41%	748.62 ± 41%
A=1e-4, B=1e-5, C=1e-5, λ=20	3884.67 ± 325%	472.11 ± 32%
A=1e-4, B=1e-5, C=1e-5, λ=50	23597.75 ± 105%	10874.62 ± 186%
A=2e-4, B=2e-5, C=2e-5, λ=10	764.47 ± 41%	4870.60 ± 291%
A=3e-4, B=3e-5, C=3e-5, λ=10	**738.99** ± 36%	6753.91 ± 247%
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 ± 25%	**162.08** ± 30%
A=5e-2, B=5e-3, C=5e-3, λ=10	751.8 ± 39%	529.62 ± 938%

Tabela 11. Comparação dos métodos em função do tempo de simulação, para a Rede 3.

PARÂMETROS HNN	Tempo de Simulação (ms)	
	HNN (Runge-Kutta)	HNN (tempo discreto)
A=1e-4, B=1e-5, C=1e-5, λ=1	253860	78687
A=1e-4, B=1e-5, C=1e-5, λ=2	145579	45125
A=1e-4, B=1e-5, C=1e-5, λ=5	68235	20922
A=1e-4, B=1e-5, C=1e-5, λ=10	41063	12484
A=1e-4, B=1e-5, C=1e-5, λ=20	209032	7281
A=1e-4, B=1e-5, C=1e-5, λ=50	1258469	166000
A=2e-4, B=2e-5, C=2e-5, λ=10	41765	78953
A=3e-4, B=3e-5, C=3e-5, λ=10	**40297**	109797
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	**2703**
A=5e-2, B=5e-3, C=5e-3, λ=10	40953	8406

Finalmente, a Tabela 12 apresenta os melhores resultados da simulação em termos dos tempos da simulação obtidos pela HNN-DE e pela HNN-RK para cada topologia de rede. Estes resultados da Tabela 12 são o melhores resultados das Tabelas 5, 8 e 11. Observe que a complexidade da rede, dada pelo número de nós e o grau do nó, aumenta o tempo de simulação pelo método baseado em HNN-RK aumenta num ritmo significante. Inversamente, para o método proposto neste trabalho, os resultados na Tabela 12 mostram que o aumento no tempo de simulação com o aumento da complexidade da rede de computador é muito menor. Conseqüentemente, estas simulações mostram que a HNN-DE é mais eficiente do que a HNN-RK.

Tabela 12. Comparação de métodos de roteamento em termos dos menores tempos de simulação obtidos para cada topologia da rede de computador.

Rede	**Número de nós**	**Tempo de Simulação (ms)**	
		HNN (Runge-Kutta)	**HNN (discrete time)**

Rede 1 (Fig. 13)	6	8250	**2031**
Rede 2 (Fig. 14)	8	14359	**2579**
Rede 3 (Fig.1 5)	8	40297	**2703**

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho, foi proposto um algoritmo de roteamento baseado na HNN-DE. A HNN-DE é baseada na HNN-RK.

Simulações foram feitas usando três topologias de redes de computadores para avaliar o método proposto. Os resultados mostraram que a HNN-DE pode encontrar trajetos ótimos bem mais rápido do que a HNN-RK. Além disso**,** as simulações mostraram que o tempo da simulação do método baseado na versão contínua do HNN aumenta com a complexidade das redes de computadores muito mais rapidamente do que no exemplo do método proposto neste trabalho.

6.1 Contribuições

O trabalho resultou nas seguintes contribuições:

- Um novo método para resolver o problema do menor caminho modificando o algoritmo de Ali e Kamoun;
- Um simulador implementado em Java que inclui o algoritmo de Dijkstra, o algoritmo de Ali e Kamoun e o algoritmo proposto;
- Um artigo científico [21] publicado numa conferência internacional FOCI 2007 do IEEE (*Institute of Electrical and Electronics Engineers*).

6.2 Trabalho Futuros

Neste trabalho, foi apresentada uma nova abordagem para resolução de uma equação da HNN-RK, caracterizando a HNN-DE (Equação12). Porém os valores dos parâmetros *A*, *B*, e *C* da Equação 12 foram escolhidos por tentativa e erro, dando margem para otimizações destes valores.

Um trabalho futuro seria encontrar valores ótimos para estes parâmetros fazendo com que reduzisse ainda mais o tempo de resposta do algoritmo.

Outro trabalho futuro seria incluir esta nova proposta no processo de roteamento das Redes Ópticas Transparentes a partir do simulador de Bastos-Filho *et al* [25].

Bibliografia

[1] E. W. Dijkstra, “A Note on Two Problems in Connection with Graphs”, *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[2] M. K. Ali, and F. Kamoun, “Neural Networks for Shortest Path Computation and Routing in Computer Networks”, *IEEE Trans. on Neural Networks*, vol. 4, no. 6, pp. 941-953, 1993.

[3] J. F. Martins-Filho, C. J. A. Bastos-Filho, E. A. J. Arantes, S. C. Oliveira, L. D. Coelho, J. P. G. Oliveira, R. G. Dante, E. Fontana and F. D. Nunes, “Novel Routing Algorithm for Transparent Optical Networks Based on Noise Figure and Amplifier Saturation”, *In Proc. Of SBMO/IEEE MTT-S IMOC 2003*, vol.2, pp. 919-923, 2003.

[4] N. Kojic, I. Reljin, and B. Reljin, “Neural Network for finding Optimal Path in Packet-Switched Network”, *In: Proc. of IEEE 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL 2004*, Serbia and Montenegro, pp. 91-96. September 2004.

[5] J. J. Hopfield, and D. W Tank, “’Neural’ computations of decision in optimization problems”, *Biol. Cybern.*, vol. 52, pp. 141- 152, 1985.

[6] H. E. Rauch, and T. Winarske, “Neural Networks for Routing Communication Traffic”, *IEEE Cont. Syst. Mag.*, pp. 26-30, April 1988.

[7] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, “Shortest Path Routing Algorithm using Hopfield Neural Networks”, *IEE Electronics Letters*, vol. 37, no. 19, pp. 1176-1178, 2001.

[8] D. C. Park, and S. E. Choi, “A neural network based multi-destination routing algorithm for communication network”, *In Proc. Joint. Conf. Neural Networks*, Anchorage, pp. 1673-1678, USA, 1998.

[9] L. Zhang and S. C. A. Thomopoulos, “Neural network implementation of the shortest path algorithm for traffic routing in communication networks”, *In Proc. Int. Joint Con& Neural Networks*, pp. II. 591, June 1989.

[10] N. Shaikh-Husin, M. K. Hani, and T. G. Seng, “Implementation of Recurrent Neural Network Algorithm for Shortest Path Calculation in Network Routing”, *In: Proc. of IEEE International Symposium on Parallel Architectures, Algorithm and Networks, ISPAN 2002*, pp. 91-96, 2004.

[11] K. M. Sim and W. H. Sun, “Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions”, *IEEE Trans. On Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 5, pp. 560-572, September 2003.

[12] G. D. Caro and M. Dorigo, “AntNet: Distributed stigmergetic control for communications networks”, *J. Artif. Intell. Res.*, vol. 9, pp. 317–365, 1998.

[13] S.H. Ngo et al., "Adaptive routing and wavelength assignment using antbased algorithm", *In Proc. of 12th IEEE ICON*, vol. 2, pp 482-486, Singapore, November, 2004.

[14] D. Bisbal et al., "Dynamic Routing and Wavelength Assignment in Optical Networks by Means of Genetic Algorithms", *Photonic Network Communications*, vol. 7, no. 1, pp. 43-58, 2004.

[15] Y.-W. Yuan, H.-H. Zhan, and L.-M. Yan, "An Adaptative QoS Route Selection Algorithm Based on Genetic Approach in Combination with Neural Netwok", *In. Proc. of the Second Int. Conf. on Machine Learning and Cybernetics*, Xi´an, 2003.

[16] V. T. Le, X. Jiang, S. H. Ngo, S. Horiguchi, "Dynamic RWA Based on the Combination of Mobile Agents Technique and Genetic Algorithms in WDM Networks with Sparse Wavelength Conversion", *in Proc. of 19th IEEE IPDPS*, 2005.

[17] G. B. Dantzig, R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling salesman problem", *Operations Research 2 (1954)*, pp. 393-410.

[18] J. J. Hopfield, "Neural Networks And Physical Systems With Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences*, Vol. 79, Pp. 2554-2558, janeiro, 1982.

[19] H. Rauch, T. Winaske, "Neural networks for routing communication traffic", *IEEE Cont. Sysl. Ma., pp. 26-30*, April 1988.

[20] G. E. Forsythe, M. A. Malcolm, C. B. Moler. "Computer Methods for Mathematical Computations", *Englewood Cliffs*, NJ: Prentice-Hall, 1977.

[21] C. J. A. Bastos-Filho, R. A. Santana, A. L. I. Oliveira, "A Novel Approach for a Routing Algorithm Based on a Discrete Time Hopfield Neural Network", *To appear in: Proc. of IEEE Symposium on Foundations of Computational Intelligence*, FOCI'07, Hawaii, 2007.

[22] IEEE – *Institute of Electrical and Electronics Engineers*, http:// www.ieee.org, Acesso em 24 de maio de 2007.

[23] G. N. Rouskas, "Optical Network Engineering, Ip Over Wdm: Building The Next Generation Optical Internet", *Sudhir Dixit*, Editor, John Wiley & Sons, Pp. 299-327, 2003.

[24] H. Zang, J. P. Jue, E B. Mukherjee. "A Review of Routing and Wavelength Assignment Approaches for Wavelength routed Optical WDM Networks". *Optical Networks*, 1(1):47–60, janeiro, 2000.

[25] J. F. Martins-Filho, C. J. A. Bastos-Filho, E. A. J. Arantes, S. C. Oliveira, L. D. Coelho, J. P. G. Oliveira, R. G. Dante, E. Fontana and F. D. Nunes, "Novel Routing Algorithm for Transparent Optical Networks Based on Noise Figure and Amplifier Saturation", *In: SBMO/IEEE MTT-S IMOC 2003*, 2003, vol. 2, pp. 919-923.

[26] STEDMAN. "Dicionário Médico". Ed. Guanabara Koogan, 2003.

[27] I. V. Tetko, D. J. Livingstone, A. I. Luik, "Neural network studies. 1. Comparison of Overfitting and Overtraining", *J. Chem. Inf. Comput. Sci*., 35, 826-833, 1995.

[28] F. Rosenblat, "The Perceptron: A probabilistic model for information storage and organization in the brain", *P`sycol. Rev*.. 65:386-408. 1958.

[29] B. Widrow, "Generalization and Information Storage in Networks of Adeline Neurons", *Self-Organizing Systems, Yovitz, M.C., Jacobi, G. T., and Goldstein, G. D. editors*, pp. 435 - 461, Chicago, 1962.

[30] D. E. Rumelhart, G. E. Hinton, R. J. Williams, (1986): "Learning internal representations by error propagation", *In: D.E. RUMELHART and J.L. MC-CLELLAND (Eds.): Parallel Distributed Processing Vol 1. MIT Press*, Cambridge, 318-362.

[31] T. Honkela, "Self-Organizing Maps in Natural Language Processing", *Espoo* 1997.

[32] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance", *Cognitive Science (Publication), 11*, 23-63, 1987.

[33] W. S. McCulloch, W. H. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bulletin of Mathematical Biophysics*, 5:115-133.

[34] M. Minsky, S. A. Papert. “Perceptrons: An Introduction to Computational Geometry.” *MIT Press*, Cambridge, MA, expanded edition, 1988/1969.

[35] J. J. Hopfield, “Neurons with graded response have collective computational properties like those of two-state neurons,” *P m . Nut. Acad Sei.*,vol. 81, pp. 3088-3092, 1984.

[36] E. K. P. Chong, S. H. Zak, “An Introduction to Optimization”, *John Wiley & Sons*, 1996.

[37] M. P. Kennedy, L. O. Chua. Neural networks for nonlinear programming. *IEEE Transactions on Circuits and Systems, 35 (5)*, 554-562, 1988

[38] H. E. Rauch, T. Winarske, “Neural networks for routing communication traffic”, *IEEE Cont. Syst. Mag.*, pp. 26-30, Apr. 1988.

[39] ADSL – *Asymmetric Digital Subscriber Line*, http://www.thinkbroadband.com/. Acesso em 19 de junho de 2007.

[40] W. Jeffrey, R. Rosner, "Neural Network Processing as a Tool for Function Optimization", *Proceeding of the International Conference on Neural Networks*, pp. 241, 1986.

[41] ICANN – *Internet Corporation for Assigned Names and Numbers*, http://www.icann.org. Acesso em 23 de maio de 2007.

[42] TCP – *Transmission Control Protocol*, RFC 793, http://www.ietf.org/rfc/rfc793.txt, Acesso em 23 de maio de 2007.

[43] IP – *Internet Protocol*, RFC 791, http://www.ietf.org/rfc/rfc791.txt, Acesso em 23 de maio de 2007.

[44] H. Zimmermann, “OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection”, *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425 – 432, April 1980.

[45] ISO – *International Standardization Organization*, http://www.iso.org. Acesso em 23 de maio de 2007.

[46] J. Kurose, K. Ross, “Redes de Computadores e a Internet - Uma Nova Abordagem”, *Makron Books*, 2003.

[47] IS-IS – *Intermediate system to intermediate system* , RFC 1195, http://www.ietf.org/rfc/rfc1195.txt, Acesso em 23 de maio de 2007.

[48] OSPF – *Open Shortest Path First*, RFC 2328, http://www.ietf.org/rfc/rfc2328.txt, Acesso em 23 de maio de 2007.

[49] RIPv1 – *Routing Information Protocol*, RFC 1058, http://www.ietf.org/rfc/rfc1058.txt, Acesso em 23 de maio de 2007.

[50] R. Bellman, “On a Routing Problem”, *in Quarterly of Applied Mathematics*, 16(1), pp.87-90, 1958.

[51] L. R. Ford, D. R. Fulkerson, “Flows in Networks”, *Princeton University Press*, 1962.

[52] RIPv2 – *Routing Information Protocol*, RFC 2453, http://www.ietf.org/rfc/rfc2453.txt, Acesso em 23 de maio de 2007.

[53] IGRP – *Interior Gateway Routing Protocol*, http://www.cisco.com/warp/public/103/5.html, Acesso em 23 de maio de 2007.

[54] EGP – *Exterior Gateway Protocol*, RFC 904, http://www.ietf.org/rfc/rfc904.txt, Acesso em 23 de maio de 2007.

[55] BGP – *Border Gateway Protocol*, RFC 4271, http://www.ietf.org/rfc/rfc4271.txt, Acesso em 23 de maio de 2007.

[56] G. G. Yen, A. N. Michel, “Stability analysis and synthesis algorithm of a class of discrete-time neural networks”, *Mathematical and Computer Modelling*, vol. 21, no. 1-2, pp. 1-29, January, 1995.

[57] S. Guo, L. Huang, L. Wang, “Exponential stability of discrete-time Hopfield neural networks”, *Computers & Mathematics with Applications*, vol. 47, no. 8-9 , pp. 1249-1256, April-May 2004.

[58] Java Technology: http://java.sun.com/. Acesso em 14 de março de 2007.

Anexo 1

Apresentaremos aqui o artigo baseado nesta monografia que foi publicado numa conferência internacional do FOCI 2007 do IEEE [21].

A Novel Approach for a Routing Algorithm Based on a Discrete Time Hopfield Neural Network

C. J. A. Bastos-Filho
Department of Computing Systems, UPE
50720-001
Recife - PE – Brazil
cjabf@dsc.upe.br

R. A. Santana
Department of Computing Systems, UPE
50720-001
Recife - PE - Brazil
Robson_poli@yahoo.com.br

A. L. I. Oliveira, *IEEE Senior Member*
Department of Computing Systems, UPE
50720-001
Recife - PE - Brazil
alio @dsc.upe.br

***Abstract*—This articles proposes a new approach to accelerate the routing algorithm based on Hopfield Neural Network. We showed that one can calculate the best route in terms of cost in a network using a discrete equation instead of the common used differential formulation. We also demonstrated that the formulation based on discrete parameters outperforms the well known formulation in terms of simulation time.**

***Index Terms*— Communication network, Hopfield neural network, Shortest path, Routing.**

I. Introduction

Routing algorithms have been hardly discussed in the scientific community, mainly because the routing process impacts drastically on communications networks performance. Ideal routing algorithm comprises finding the best path between source and destination nodes, enabling high quality transmission and avoiding penalties caused by physical layer impairments. There are different ways to find the optimal route. Some algorithms determine the routes based on the shortest path (SP) [1], minor delay [2], higher signal-to-noise ratio (in All-Optical Networks case) [3], load balance [4], among others.

Moreover, to maintain the Quality of Service (*QoS*) computations have to be carried out in real time and should be adaptative. To provide this flexibility, many techniques from Computational Intelligence (CI) have been tested. The most used techniques are Artificial Neural Networks (*ANN*) [2,4-10], Ant Colony Optimization [11-12], Genetic Algorithms [13-14], and Hybrid algorithms combining those techniques [15-16].

ANN are very good candidates for solving the problem due to its high computational speed [2]. Hopfield and Tank described an ANN with feedback configuration suitable for solving constrained optimization problems, especially the Traveling Salesman Problem (TSP) [5]. Therefore, this ANN configuration is called Hopfield neural network (*HNN*). The use of HNN to find the shortest path between two nodes in a communication network was initiated by Rauch and Winarske [6]. However, this propose requires a prior knowledge of the network topology. To outperform this limitation, Ali and Kamoun [2] proposed a novel adaptive algorithm, where the weight matrix just carries convergence information. The information about the link cost and the topology is added through the bias as shown in Fig. 1. Additional papers report techniques to avoid involved loops and problems in the algorithm convergence [7-8]. However these algorithms are based on the same differential equation proposed by Ali and Kamoun [2].

In this paper, we propose a new approach based on discrete and finite difference equation to speed up the convergence of the HNN. The rest of the paper is organized as follows. In section II, we describe the ANN based routing algorithm proposed by [2]. In section III, we show our contribution on the solution method and in section IV we provide a detailed description of the proposed algorithm and of a software tool that we have developed for simulations. In section V, we present the simulation results and compare our approach based on difference equations to the formulation given by [2]. In section VI, we present our conclusions.

II. Hopfield Neural Networks For Routing in Communications Networks

The block diagram of the Hopfield neural network (HNN) is depicted in Fig. l. The processing elements are the neurons and they are full-connected, *i.e.* every output of each neuron is connected to inputs of all other neurons via synaptic weights. To solve routing problems in communication networks, Ali and Kamoun [2] proposed that each link in the communication network between two adjacent nodes has one neuron associated. For example, a link from node *x* to node *i* is associated to a neuron that have the *xi* description. The output of a neuron V_{xi}, depends on its input U_{xi}, according the sigmoidal function as expressed in (1). Notice that the input of each neuron corresponds to the sum of all the outputs of the neurons of the HNN multiplied by a factor represented by the matrix $T_{xi,yj}$ added to an external bias I_{xi}. The $T_{xi,yj}$ element represents the synaptic weights connecting the output of the neuron *yj* to the sum point in the input of the neuron *xi*.

$$V_{xi} = \frac{1}{1+e^{-\lambda_{xi}U_{xi}}} \qquad \forall(x,i) \in \overline{N}\mathrm{X}\overline{N} \,/\, x \neq i \tag{1}$$

The parameter λ_{xi} determines the computation time to convergence and the correctness of the algorithm. In our

simulations we used the same λ_{xi} for all neurons in the HNN. Therefore, we call this parameter λ in the rest of the paper, instead of λ_{xi}. The behavior of the sigmoid logistic function is shown in the Fig. 2 for different values of λ. As higher as the parameter λ is, the logistic function tends to a step function.

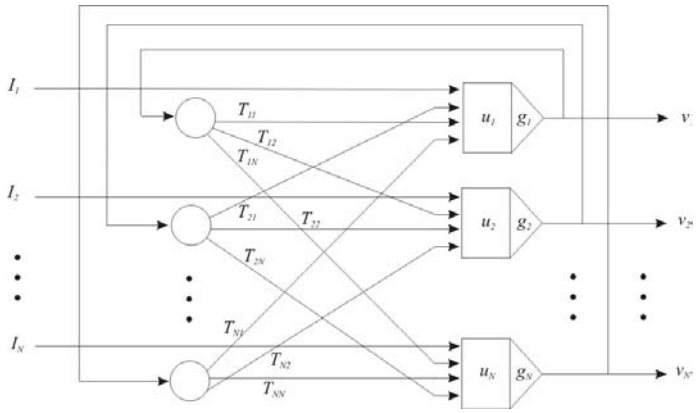

Fig. 1. Hopfield Neural Network Configuration.

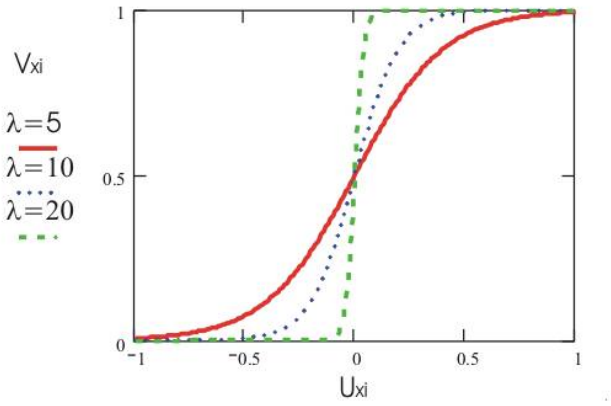

Fig. 2. Sigmoid Logistic function behavior for different values of .

If every link in the network has a nonnegative cost associated C_{ij}, the goal of the HNN is to find the path that minimizes the cost from some source node *s* to a destination node *d* through the communication network. Thus, the HNN should indicate a directed path as an ordered sequence of nodes connecting *s* to *d*, so the sum of all the costs connecting these nodes provides the lower possible cost. The path that provides minimum cost is defined as L_{sd}. In most cases, the cost matrix is symmetric ($C_{xi} = C_{ix}$), though exists some papers considering a asymmetric cost matrix ($C_{xi} \neq C_{ix}$) [4]. Notice that elements C_{ii} are nulls because one node cannot be connected to itself.

The matrix ρ_{xi} defines if the arc *xi* exists in the topology of the communication network used in the simulation. If the arc *xi* exists then $\rho_{xi} = 0$, otherwise $\rho_{xi} = 1$.

When the simulation converges, *i.e.* the change of every output values V_{xi} in an interactions are below a predefined criteria, an adjustment is done in each output. If an output has a value greater than 0.5 it is adjusted to "1", otherwise it is adjusted to "0". The final value of the V_{xi} will define if the arc related with the neuron *xi* belongs or not to the shortest path L_{sd}.

$$V_{xi} = \begin{cases} 1, arc_{xi} \in L_{sd} \\ 0, otherwise \end{cases} \quad (2)$$

Beside this, each cell can be externally excited by input bias I_{xi}. These system inputs can be used to set the general level of excitability of the whole network and represent the actual data provided by the user to the neural network. We used the following expression for the bias [2]:

$$I_{xi} = -\frac{\mu_1}{2} C_{xi}\left(1 - \delta_{xd}\delta_{is}\right) - \frac{\mu_2}{2}\rho_{xi}\left(1 - \delta_{xd}\delta_{is}\right) - \frac{\mu_4}{2} + \frac{\mu_5}{2}\delta_{xd}\delta_{is} \quad \forall(x \neq i), \forall(y \neq i) \quad (3)$$

where δ is the Kronecker function and μ_1, μ_2, μ_4 and μ_5 are constants.

The synaptic matrix $T_{xi,yj}$ and the energy function of the HNN are described as [2]:

$$T_{xi,yj} = \mu_4\delta_{xy}\delta_{ij} - \mu_3\delta_{xy} - \mu_3\delta_{ij} + \mu_3\delta_{jx} + \mu_3\delta_{iy} \quad (4)$$

$$E = \frac{\mu_1}{2}\sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n}\sum_{\substack{i=1 \\ i\neq x}}^{n} C_{xi}V_{xi} + \frac{\mu_2}{2}\sum_{\substack{x=1 \\ (x,i)\neq(d,s)}}^{n}\sum_{\substack{i=1 \\ i\neq x}}^{n}\rho_{xi}V_{xi} + \frac{\mu_3}{2}\sum_{x=1}^{n}\left\{\sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi} - \sum_{\substack{i=1 \\ i\neq x}}^{n} V_{ix}\right\}^2 + \frac{\mu_4}{2}\sum_{x=1}^{n}\sum_{\substack{i=1 \\ i\neq x}}^{n} V_{xi}\left(1 - V_{xi}\right) + \frac{\mu_5}{2}\left(1 - V_{ds}\right) \quad (5)$$

where *E* is the energy of the HNN and μ_3 is a constant.

These parameters have specific functions on the energy function: μ_1 minimizes the total cost; μ_2 prevents nonexistent links from being included in the chosen path; μ_3 is zero for every node in the valid path; μ_4 forces the HNN to converge to a stable state and μ_5 is introduced to ensure that the source and the destination nodes belong to the solution. Ahn *et al* proposed others terms to avoid loops [7], but we did not considered them in this work.

Therefore, if the system is stable in Liapunov sense, then iterations lead to smaller output changes. As a consequence, after a convenient number of iterations the HNN reaches some

minima of the system energy. Ali and Kamoun [2] resolve the system using the differential equation described below.

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} + \sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj} + I_{xi} \tag{6}$$

Notice that the sum of second and the third terms represents the energy variation of the HNN. Therefore, one can rewrite (6) as follow:

$$\frac{dU_{xi}}{dt} = -\frac{U_{xi}}{\tau} - \frac{dE}{dV_{xi}} \tag{7}$$

Ali and Kamoun [2] use the Runge-Kutta method to solve the equation. To simplify and accelerate the resolution we propose in the next section a new approach based on a discrete difference equation.

III. HNN Approach Based on Energy Finite Difference Equation

Some authors have proposed to solve Hopfield neural networks using discrete time energy equations [17-18]. However, none of them applied it to the routing problem in communications networks. Thus, we adapted the discrete time formulation to HNN modeled to solve the routing problem. We believed that it could accelerate the calculus of routes. Therefore, instead of solving the systems using a differential equation that requires sophisticated methods (like Runge-Kutta, *i.e.*, Eq. (7)), we propose a simple approach based on a simple difference equation based on discrete time reference. The equation used to calculate the next input value of the neurons is shown below.

$$\begin{aligned} U_{xi}[n+1] &= -AU_{xi}[n] - BU_{xi}[n-1] - CU_{xi}[n-2] \\ &+ \sum_{y=1}^{n}\sum_{\substack{j=1 \\ j\neq y}}^{n} T_{xi,yj}V_{yj}[n] + I_{xi}[n] \end{aligned} \tag{8}$$

where $U_{xi}[n+1]$ is the next input of the neuron *xi* calculated based on the output values of all the neurons of the network $V_{yj}[n]$, the external bias $I_{xi}[n]$ and on its own input in previous instants $U_{xi}[n]$, $U_{xi}[n-1]$ and $U_{xi}[n-2]$. A, B and C are constants that regulate the weight of the previous inputs.

IV. Algorithm Description and Simulation Tool

We developed a simulation tool in *Java* to calculate the route based on HNN that follows the flow chart shown in Fig. 3. The first step is to get the parameters μ_1, μ_2, μ_3, μ_4 and μ_5 to determine the weight matrix. After that, the simulations parameters are required. Using this data the software calculates the topology and the bias matrixes. So, the neurons are initialized (the neurons input are set with a little noise to accelerate the convergence). Default values of the simulation parameters are presented in the table I.

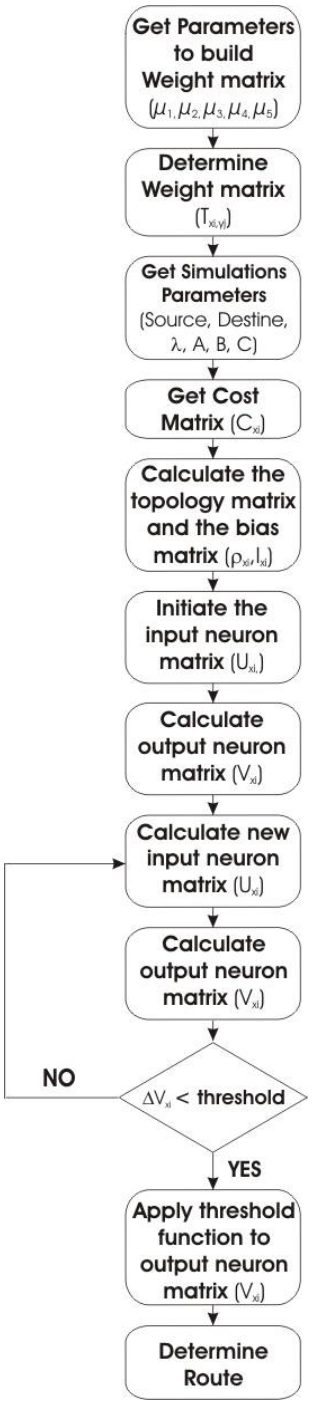

Fig. 3. Flow chart of the discrete time Hopfield Neural Network algorithm for routing in communications networks.

The value of the logistic function parameter used in the simulations is also presented in table I. It is well known that higher values for λ leads to faster convergence of the algorithm, despite it can lead to error in the route

determination.

TABLE I
PARAMETERS USED IN SIMULATIONS AND THEIR DEFAULT VALUES.

PARAMETER	VALUE (DEFAULT)
μ_1	950
μ_2	2500
μ_3	1500
μ_4	475
μ_5	2500
A	0.0001
B	0.00001
C	0.00001
λ	1

V. SIMULATIONS RESULTS

In this section we report on a number of simulations performed to compare the routing method proposed in this paper with the method proposed by Ali and Kamoun [2]. The routing algorithms based on Hopfiel neural networks are also compared to Dijkstra's algorithm [1]. We have considered three computer networks topologies in our simulations. They are depicted in Figures 3, 4 and 5, respectively.

In each simulation set we are interested in comparing the three routing methods regarding the number of optimal routes found as well as simulation time. In addition, for the methods based on HNN we have compared the number of iterations needed for convergence.

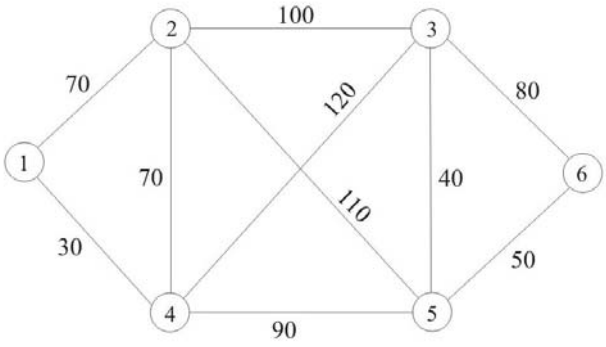

Fig. 3. Network 1: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

One simulation set for a given computer network topology consists of a number of simulations considering the three routing algorithms. In each simulation, the source and destination nodes are randomly selected. Next, Dijkstra's algorithm is used to compute the shortest path between the source and destination nodes. Finally, both the continuous version of the HNN routing method [2] and the discrete version proposed in this paper are executed independently to obtain the shortest path. The paths obtained by the HNN methods are compared to the one computed via Dijkstra's algorithm.

A simulation set is executed considering the same values for the parameters of the HNNs. After a simulation set is executed, we obtain the total simulation time for each of the HNN-based routing methods. In addition, we obtain the mean and standard deviation of the number of iterations needed for each of the HNN-based routing methods. Finally, we compute, for the HNN-based methods, the percentage of the simulations in which these method produced an optimal path, that is, a path identical to that produced by Dijkstra's algorithm for the same simulation (that is, a simulation using the same source and destination nodes).

Tables 2, 3 and 4 report some simulation results for the computer network 1, depicted in Figure 3. Each line of tables 2, 3 and 4 report the results obtained for a given simulation set. Each simulation set comprised 100 simulations with randomly selected source and destination nodes. In all simulations, the HNN-based methods employed the default values of the parameters μ_1, μ_2, μ_3 , μ_4 and μ_5 given in Table 1. We varied the values of parameters A, B, C and λ to analyze their influence on performance.

Table 2 shows the accuracy of the HNN methods with respect to the results furnished by Dijkstra's algorithm. The results of table 2 show that for most values of the parameters the HNN algorithms obtain the same paths obtained by Dijkstra's algorithm (that is, HNNs obtained 100% accuracy).

TABLE II
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	**Accuracy**	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	82.00%

The mean (and standard deviation) number of iterations for convergence of each HNN-based method for network 1 are presented in Table 3. It can be observed that the number of iterations depends significantly on the values of the parameters. By comparing the best results obtained by each method (in bold) we conclude that the proposed method is able to converge using less iterations.

TABLE III
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4163.79 (1059.71)	4277.45 (998.71)
A=1e-4, B=1e-5, C=1e-5, λ=2	2569.23 (762.26)	2575.49 (664.21)
A=1e-4, B=1e-5, C=1e-5, λ=5	1265.14 (407.82)	1267.40 (400.77)
A=1e-4, B=1e-5, C=1e-5, λ=10	713.69 (241.67)	721.45 (229.99)
A=1e-4, B=1e-5, C=1e-5, λ=20	**464.93** (192.78)	460.52 (183.60)
A=1e-4, B=1e-5, C=1e-5, λ=50	10157.82 (19921.19)	8321.70 (18238.32)
A=2e-4, B=2e-5, C=2e-5, λ=10	835.0 (340.93)	487.72 (201.69)
A=3e-4, B=3e-5, C=3e-5, λ=10	778.57 (269.36)	**332.73** (126.98)
A=4e-4, B=4e-5, C=4e-5, λ=10	765.52 (262.48)	2751.17 (201.69)
A=5e-2, B=5e-3, C=5e-3, λ=10	4371.56 (1173.78)	29557.71 (24574.91)

Table 4 compares the routing methods in terms of simulation time. The results show that for most combinations of the parameters (expect that in the last line of the table) the method proposed in this paper is much faster than HNN solved with the Runge-Kutta method [2]. Moreover, the smaller simulation time for the proposed method was much smaller than that of HNN solved via Runge-Kutta (entries in bold in table 4). For network 1, the smaller simulation time obtained by Runge-Kutta was 8250ms whereas the proposed method achieved 2021ms.

TABLE IV
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 1 (FIGURE 3)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	73735	26687
A=1e-4, B=1e-5, C=1e-5, λ=2	45922	16078
A=1e-4, B=1e-5, C=1e-5, λ=5	22782	7921
A=1e-4, B=1e-5, C=1e-5, λ=10	12719	4562
A=1e-4, B=1e-5, C=1e-5, λ=20	**8250**	2875
A=1e-4, B=1e-5, C=1e-5, λ=50	172890	47125
A=2e-4, B=2e-5, C=2e-5, λ=10	14641	3015
A=3e-4, B=3e-5, C=3e-5, λ=10	13797	**2031**
A=4e-4, B=4e-5, C=4e-5, λ=10	13703	16828
A=5e-2, B=5e-3, C=5e-3, λ=10	81422	192015

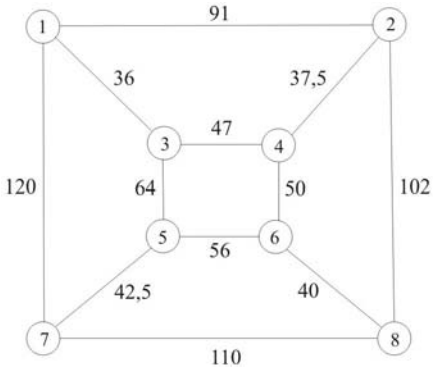

Fig. 4. Network 2: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

The simulation results for network 2 (Fig. 4) are show in Tables 5, 6 and 7, which follow the same structure of tables 2, 3, and 4, respectively. We have also tested the proposed method on a third computer network topology (network 3), which is depicted in Figure 5. The simulation results for network 3 are presented in Tables 8, 9, and 10. The same observations made for network 1 can be made for networks 2 and 3, namely, the proposed method is able to produce optimal routes with much smaller simulation times than the HNN-based method solved via Runge-Kutta [2].

TABLE V
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Accuracy	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=100	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=4e-4, B=4e-5, C=4e-5, λ=10	100.00%	100.00%
A=5e-4, B=5e-5, C=5e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-3, B=5e-4, C=5e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	95.00%

TABLE VI
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4433.95 (1372.24)	4623.90 (1451.87)
A=1e-4, B=1e-5, C=1e-5, λ=2	2705.53 (836.33)	2802.07 (883.75)
A=1e-4, B=1e-5, C=1e-5, λ=5	1420.08 (385.44)	1463.95 (404.93)
A=1e-4, B=1e-5, C=1e-5, λ=10	834.88 (245.88)	858.07 (252.09)
A=1e-4, B=1e-5, C=1e-5, λ=20	495.75 (146.07)	502.75 (152.82)
A=1e-4, B=1e-5, C=1e-5, λ=50	**253.49** (77.82)	260.12 (74.1)
A=1e-4, B=1e-5, C=1e-5, λ=100	4634.77 (14266.76)	192.06 (66.89)
A=2e-4, B=2e-5, C=2e-5, λ=10	786.28 (248.24)	490.88 (160.81)
A=3e-4, B=3e-5, C=3e-5, λ=10	830.34 (246.89)	391.53 (124.89)
A=4e-4, B=4e-5, C=4e-5, λ=10	832.49 (233.22)	327.94 (94.86)
A=5e-4, B=5e-5, C=5e-5, λ=10	822.15 (242.44)	267.20 (90.22)
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 (213.48)	**162.08** (48.73)
A=5e-3, B=5e-4, C=5e-4, λ=10	805.86 (234.75)	552.26 (4970.55)
A=5e-2, B=5e-3, C=5e-3, λ=10	869.77 (247.34)	4030.47 (13558.24)

TABLE VII
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 2 (FIGURE 4)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	259093	77844
A=1e-4, B=1e-5, C=1e-5, λ=2	149906	46266
A=1e-4, B=1e-5, C=1e-5, λ=5	81859	25188
A=1e-4, B=1e-5, C=1e-5, λ=10	46156	15375
A=1e-4, B=1e-5, C=1e-5, λ=20	27360	9000
A=1e-4, B=1e-5, C=1e-5, λ=50	**14359**	4672
A=1e-4, B=1e-5, C=1e-5, λ=100	257406	**2579**
A=2e-4, B=2e-5, C=2e-5, λ=10	45828	8672
A=3e-4, B=3e-5, C=3e-5, λ=10	48500	6766
A=4e-4, B=4e-5, C=4e-5, λ=10	48500	5703
A=5e-4, B=5e-5, C=5e-5, λ=10	48078	4532
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	2703
A=5e-3, B=5e-4, C=5e-4, λ=10	46969	9250
A=5e-2, B=5e-3, C=5e-3, λ=10	51672	69203

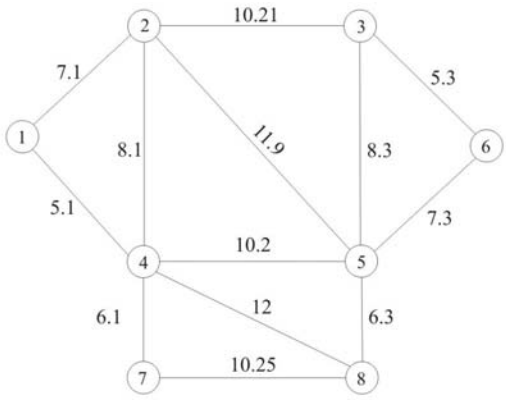

Fig. 5. Network 3: one of the computer network topologies used in the simulations. The numbers represent the costs of the links.

TABLE VIII
COMPARISON OF HNN-BASED METHODS IN TERMS OF ACCURACY (OF THE PATHS FOUND) FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Accuracy	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=2	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=5	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=10	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=20	100.00%	100.00%
A=1e-4, B=1e-5, C=1e-5, λ=50	100.00%	100.00%
A=2e-4, B=2e-5, C=2e-5, λ=10	100.00%	100.00%
A=3e-4, B=3e-5, C=3e-5, λ=10	100.00%	100.00%
A=1e-3, B=1e-4, C=1e-4, λ=10	100.00%	100.00%
A=5e-2, B=5e-3, C=5e-3, λ=10	100.00%	86.00%

TABLE IX
COMPARISON OF HNN-BASED METHODS IN TERMS OF ITERATIONS FOR CONVERGENCE FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Iterations	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	4674.93 (2433.65)	4742.56 (2572.75)
A=1e-4, B=1e-5, C=1e-5, λ=2	2663.18 (1197.25)	2692.09 (1214.80)
A=1e-4, B=1e-5, C=1e-5, λ=5	1241.41 (420.57)	1246.74 (418.65)
A=1e-4, B=1e-5, C=1e-5, λ=10	752.1 (311.52)	748.62 (307.55)
A=1e-4, B=1e-5, C=1e-5, λ=20	3884.67 (12652.35)	472.11 (152.88)
A=1e-4, B=1e-5, C=1e-5, λ=50	23597.75 (24862.92)	10874.62 (20295.35)
A=2e-4, B=2e-5, C=2e-5, λ=10	764.47 (319.06)	4870.60 (14195.32)
A=3e-4, B=3e-5, C=3e-5, λ=10	**738.99** (268.14)	6753.91 (16719.99)
A=1e-3, B=1e-4, C=1e-4, λ=10	828.03 (213.48)	**162.08** (48.73)
A=5e-2, B=5e-3, C=5e-3, λ=10	751.8 (281.56)	529.62 (4972.73)

TABLE X
COMPARISON OF ROUTING METHODS IN TERMS OF SIMULATION TIMES FOR NETWOK 3 (FIGURE 5)

HNN PARAMETERS	Simulation time (ms)	
	HNN (Runge-Kutta)	HNN (discrete-time)
A=1e-4, B=1e-5, C=1e-5, λ=1	253860	78687
A=1e-4, B=1e-5, C=1e-5, λ=2	145579	45125
A=1e-4, B=1e-5, C=1e-5, λ=5	68235	20922
A=1e-4, B=1e-5, C=1e-5, λ=10	41063	12484
A=1e-4, B=1e-5, C=1e-5, λ=20	209032	7281
A=1e-4, B=1e-5, C=1e-5, λ=50	1258469	166000
A=2e-4, B=2e-5, C=2e-5, λ=10	41765	78953
A=3e-4, B=3e-5, C=3e-5, λ=10	**40297**	109797
A=1e-3, B=1e-4, C=1e-4, λ=10	48422	**2703**
A=5e-2, B=5e-3, C=5e-3, λ=10	40953	8406

Finally, Table 11 presents the best simulation results in terms of simulation times obtained by the proposed method and HNN solved via Runge-Kutta for each network topology. These results of table 11 are the best results of tables 4, 7, and 10. Notice that as the complexity of the network (given by the number of nodes and the node degree) increases, the simulation time of the HNN solved by Runge-Kutta increases at a significant pace. Conversely, for the method proposed in this paper, the results of table 11 show that the increase in simulation time with computer network complexity is much smaller. Therefore, these simulations show that the proposed method is more efficient than the solution based on Runge-Kutta [2] and more adequate to handle more complex computer networks.

TABLE XI
COMPARISON OF ROUTING METHODS IN TERMS OF THE SMALLER SIMULATION TIMES OBTAINED FOR EACH COMPUTER NETWORK TOPOLOGY

Network	**Number of nodes**	**Node degree**	**Simulation Time (ms)**	
			HNN (Runge-Kutta)	**HNN (discrete time)**
Network 1 (Fig. 3)	6	3.33	8250	**2031**
Network 2 (Fig. 4)	8	3.00	14359	**2579**
Network 3 (Fig. 5)	8	3.25	40297	**2703**

VI. CONCLUSIONS

In this paper we have proposed a routing algorithm based on discrete-time Hopfield neural networks (HNN). The proposed method is based on a previous formulation based on HNNs which uses a differential equation solved through the Runge-Kutta method. In contrast, our method is based on a discrete difference equation.

We have carried out a number of simulations using three computer networks topologies to evaluate the proposed method. The results have shown that the proposed method is able to find optimal paths much faster than the method based on an HNN solved via Runge-Kutta. Furthermore, our simulations have shown that simulation time of the method based on continuous version of the HNN increases with the complexity of the computer networks much faster than in the case of the method proposed in this paper.

REFERENCES

[1] E. W. Dijkstra, 'A Note on Two Problems in Connection with Graphs', *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.

[2] M. K. Ali, and F. Kamoun, 'Neural Networks for Shortest Path Computation and Routing in Computer Networks', *IEEE Trans. on Neural Networks*, vol. 4, no. 6, pp. 941-953, 1993.

[3] J. F. Martins-Filho, C. J. A. Bastos-Filho, E. A. J. Arantes, S. C. Oliveira, L. D. Coelho, J. P. G. Oliveira, R. G. Dante, E. Fontana and F. D. Nunes, "Novel Routing Algorithm for Transparent Optical Networks Based on Noise Figure and Amplifier Saturation", *In Proc. of SBMO/IEEE MTT-S IMOC 2003*, vol.2, pp. 919-923, 2003.

[4] N. Kojic, I. Reljin, and B. Reljin, 'Neural Network for finding Optimal Path in Packet-Switched Network', *In: Proc. of IEEE 7th Seminar on Neural Network Applications in Electrical Engineering, NEUREL 2004,* Serbia and Montenegro, pp. 91-96. September 2004.

[5] J. J. Hopfield, and D. W Tank, '"Neural' computations of decision in optimization problems", *Biol. Cybern*., vol. 52, pp. 141- 152, 1985.

[6] H. E. Rauch, and T. Winarske, 'Neural Networks for Routing Communication Traffic', *IEEE Cont. Syst. Mag.*, pp. 26-30, April 1988.

[7] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, 'Shortest Path Routing Algorithm using Hopfield Neural Networks', *IEE Electronics Letters*, vol. 37, no. 19, pp. 1176-1178, 2001.

[8] D. C. Park, and S. E. Choi, 'A neural network based multi-destination routing algorithm for communication network'. *In Proc. Joint. Conf. Neural Networks*, Anchorage, pp. 1673-1678, USA, 1998.

[9] L. Zhang and S. C. A. Thomopoulos, 'Neural network implementation of the shortest path algorithm for traffic routing in communication networks', *In Proc. Int. Joint Con& Neural Networks*, pp. II. 591, June 1989.

[10] N. Shaikh-Husin, M; K. Hani, and T. G. Seng, 'Implementation of Recurrent Neural Network Algorithm for Shortest Path Calculation in Network Routing', *In: Proc. of IEEE International Symposium on Parallel Architectures, Algorithm and Networks, ISPAN 2002,* pp. 91-96, 2004.

[11] K. M. Sim and W. H. Sun, 'Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions', *IEEE Trans. On Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 5, pp. 560-572, September 2003.

[12] G. D. Caro and M. Dorigo, 'AntNet: Distributed stigmergetic control for communications networks', *J. Artif. Intell. Res.*, vol. 9, pp. 317–365, 1998.

[13] S.H. Ngo et al., 'Adaptive routing and wavelength assignment using ant-based algorithm', *In Proc. of 12th IEEE ICON*, vol. 2, pp 482-486, Singapore, November, 2004.

[14] D. Bisbal et al., 'Dynamic Routing and Wavelength Assignment in Optical Networks by Means of Genetic Algorithms', *Photonic Network Communications*, vol. 7, no. 1, pp. 43-58, 2004.

[15] Y.-W. Yuan, H.-H. Zhan, and L.-M. Yan, ' An Adaptative QoS Route Selection Algorithm Based on Genetic Approach in Combination with Neural Netwok*, In. Proc. of the Second Int. Conf. on Machine Learning and Cybernetics*, Xi´an, 2003.

[16] V. T. Le, X. Jiang, S. H. Ngo, S. Horiguchi, "Dynamic RWA Based on the Combination of Mobile Agents Technique and Genetic Algorithms in WDM Networks with Sparse Wavelength Conversion", *in Proc. of 19th IEEE IPDPS*, 2005.

[17] G. G. Yen, and A. N. Michel, 'Stability analysis and synthesis algorithm of a class of discrete-time neural networks', *Mathematical and Computer Modelling*, vol. 21, no. 1-2, pp. 1-29, January, 1995.

[18] S. Guo, L. Huang, and L. Wang, 'Exponential stability of discrete-time Hopfield neural networks', *Computers & Mathematics with Applications*, vol. 47, no. 8-9 , pp. 1249-1256, April-May 2004.