Influência do Caos em Redes Neurais Auto-Organizáveis ao Resolver Problemas de Otimização

Josué Ervin Musial
Paulo Henrique Siqueira
Programa de Pós-Graduação em Métodos Numéricos em Engenharia - PPGMNE
Universidade Federal do Paraná - UFPR
Curitiba, Paraná, Brasil
ajmusial@gmail.com

***Resumo*—Este artigo tem como objetivo apresentar uma modificação no algoritmo das redes de Mapas Auto-Organizáveis (Self-Organizing Maps – SOM) para resolver o Problema do Caixeiro Viajante (Travelling Salesman Problem – TSP) utilizando o conceito de caos determinístico. A solução para o TSP é classificada computacionalmente como um problema de otimização combinatória que demanda um grande esforço computacional e assim, surge um grande interesse em desenvolver heurísticas eficientes para resolvê-lo. Durante os testes computacionais das modificações propostas no trabalho a interferência do conceito de caos determinístico utilizado nos resultados despertou bastante atenção. Neste artigo ainda são apresentados os conceitos e a metodologia utilizada. Ao final são discutidos os resultamos e a influência do caos determinístico sobre os mesmos.**

***Palavras-chave*—mapas auto-organizáveis; caos; problemas de otimização**

I. INTRODUÇÃO

As redes neurais tem sido amplamente aplicadas nos mais diferentes campos científicos para a solução dos mais variados problemas, bem como as mais variadas meta-heurísticas baseadas em algum comportamento biológico [1]. Essas técnicas, em geral, tem alcançado um ótimo desempenho na solução de problemas de alta complexidade computacional, em especial nos problemas da pesquisa operacional como, por exemplo, em [2] e [3].

Admitindo heurísticas baseadas no modelo de aprendizado do cérebro humano e que a formação dessa experiência se dá de maneira aleatória, os modelos heurísticos clássicos se mostram rígidos [5] e um tanto quanto limitados se comparados com a capacidade de aprendizado dos organismos biológicos [9]. A melhor maneira de simular e/ou explicar o paradigma do aprendizado para aproximar os algoritmos heurísticos da capacidade cognitiva das estruturas biológicas é a utilização de caos nas redes neurais de qualquer natureza [5].

O Problema do Caixeiro Viajante (Travelling Salesman Problem – TSP) é utilizado neste estudo, porque serve como plataforma de testes para investigar as diversas propostas heurísticas, além de ser um problema de alta aplicabilidade no mundo real, de fácil compreensão e descrição. Entretanto de difícil solução, pois pertence à classe de problemas NP-hard e que demanda um tempo computacional que pode ser exponencial dependendo da instância a ser solucionada. Assim, vários métodos tem sido desenvolvidos com o propósito de se resolver instâncias cada vez maiores desse problema em um tempo computacional menor.

Os mapas auto-organizáveis [11] são conhecidos por sua capacidade de agrupar padrões de entrada devido ao modelo de sua função de densidade de probabilidade gerar uma ordem espacial. O trabalho de [5] apresenta simulações que ilustram a capacidade que o mapa auto-organizável caótico tem para executar cada uma destas tarefas. O caos é usado para escolher o neurônio vencedor de forma probabilística mas, o efeito não é examinado em relação aos problemas difíceis como por exemplo na resolução do TSP.

Este trabalho apresenta uma proposta de melhoria para a solução do clássico problema do Caixeiro Viajante, para tanto utilizou-se a meta-heurística clássica Redes de Kohonen, que foi adaptada para resolver este problema. Além dessa adaptação foi implementada a característica caótica nesse algoritmo com o objetivo de testá-la e comprovar sua eficiência. No final deste artigo apresentamos os primeiros resultados obtidos.

II. REDES NEURAIS AUTO-ORGANIZÁVEIS

A rede de Kohonen, mais conhecida como rede SOM (Self-Organizing Map) ou ainda, rede de mapas auto-organizáveis, foi desenvolvida pelo finlandês Teuvo Kohonen na década de 80. Seu algoritmo tem como princípio a auto-organização, um processo muito semelhante ao que ocorre no interior do cérebro humano [12]. O modelo de Kohonen pertence a uma classe de algoritmos de codificação vetorial e, gera um mapeamento topológico dos vetores de entrada.

A rede SOM emprega uma mistura dinâmica de competição e cooperação para permitir a formação emergente de um isomorfismo entre um espaço de característica e uma matriz de neurônios [9]. Ela simplesmente inspeciona os dados de entrada, buscando regularidades e padrões e organiza-os de tal maneira a formar uma ordenada descrição dos dados de

entrada. Esta descrição pode levar a uma solução do problema em questão [6].

O algoritmo se resume em alguns passos elementares: apresentação de uma entrada, cálculo do neurônio vencedor e atualização dos pesos até que o mapa de características esteja completo, ou seja, não sofra modificações significativas em seus pesos a cada interação. Os elementos fundamentais do algoritmo são os seguintes:

1) *Inicialização.* Atribua valores iniciais aleatórios e distintos para cada um dos pesos $w_j(0)$, onde $j = 1, 2, \ldots, l$ e l é o número de neurônios da rede.
2) *Apresentação de um padrão de entrada.* Sorteie um vetor de entrada x ainda não apresentado à rede no tempo n.
3) *Cálculo do neurônio vencedor.* Encontre o neurônio vencedor $i(x)$ para o padrão de entrada x na iteração n utilizando o critério da distância Euclidiana.

$$i(x) = \arg\min_j \|x(n) - w_j\|, \quad j = 1, 2, \ldots, l \qquad (1)$$

4) *Atualização.* Ajuste o valor dos pesos sinápticos de todos os neurônios usando a seguinte equação de atualização:

$$w_j(n+1) = w_j(n) + \eta(n) h_{j,i(x)}(n)(x - w_j(n)) \quad (2)$$

onde $\eta(n)$ é o parâmetro de aprendizagem e $h_{j,i(x)}(n)$ é a função de vizinhança onde o centro é o neurônio vencedor $i(x)$.
5) *Cálculo do Erro.* Continue até que não ocorram mudanças significativas no ajuste dos pesos.

III. CAOS

Caos Determinístico ou simplesmente Caos é o fenômeno pelo qual um comportamento imprevisível, aparentemente aleatório é produzido por um sistema completamente determinístico. Tal atividade é sensivelmente dependente das condições iniciais e, por consequência, imprevisível. Deste modo, pequenas diferenças nas condições iniciais podem gerar grandes diferenças nas saídas dentro de um tempo finito [15].

Um dos sistemas clássicos mais simples, capazes de produzir caos determinístico é a equação logística:

$$y(n+1) = 4\,\mu\, y(n)\,[1 - y(n)] \qquad (3)$$

O comportamento dessa equação de diferença de primeira ordem muda drasticamente quando o parâmetro de bifurcação μ é alterado, conforme mostra a Figura 1. Para $2 < \mu < 2,9$ a saída converge para um único valor diferente de zero. Quando $\mu > 2,9$ a saída começa a oscilar primeiro entre 2 valores, depois 4 valores, em seguida, 8 valores e assim por diante, até que para $\mu > 3,6$ a saída se torna caótica.

Num regime caótico um sistema apresenta sensibilidade às condições iniciais, ou seja, mesmo se as condições iniciais são próximas o resultado das iterações pode divergir exponencialmente, com o decorrer do tempo. Uma maneira de medir essa divergência é através do *expoente de Lyapunov* λ [15].

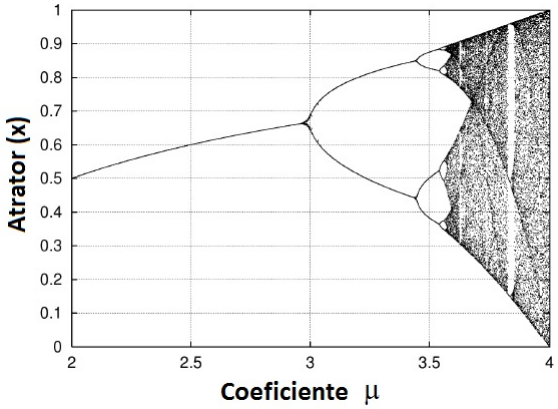

Figura 1. Diagrama de Bifurcação para o mapa logístico.

O expoente de Lyapunov $\lambda(x_0)$ é definido na equação (4).

$$\lambda(x_0) = \lim_{N\to\infty} \frac{1}{N} \sum_{n-1}^{N-1} \ln \mid f'(x_n) \mid \qquad (4)$$

onde N é o número de iterações, $x_1 = f(x_0),\ x_2 = f(x_1) \ldots$. Assim se $\lambda(x_0) > 0$ o sistema apresenta regime caótico e se $\lambda(x_0) \leq 0$ o sistema apresenta regime regular [15] Outro conceito utilizado neste estudo é a unidade neural caótica fundamentada na equação logística invertida, na qual a entrada convencional da unidade neural é dada por:

$$net = \sum_i w_i x_i \qquad (5)$$

onde x_i são as entradas da rede neural e w_i são os pesos associados. A próxima saída $y(n+1)$ da unidade neural depende da entrada da saída anterior como segue:

$$y(n+1) = 1 - 4\,[1 - net]\, y(n)\,[1 - y(n)] \qquad (6)$$

O comportamento desta unidade neural depende da sua entrada, se $net < 0,11$ a unidade exibe atividade caótica neural, pois satisfaz as mesmas condições da Equação (5) que apresenta atividade caótica para $\mu > 0,89$. Para $0,11 < net < 0,25$ produz atividade periódica, enquanto que para $net > 0,25$ produz um único valor que cresce com o aumento do valor net e é igual a unidade para $net > 0,75$.

Um dos pioneiros na utilização de caos em redes neurais foi Aihara e Toyoda (1990)[1]. Em seu trabalho é proposto o modelo com um único neurônio com dinâmica caótica, considerando as seguintes propriedades dos neurônios biológicos: memória, refratariedade e soma espaço-temporal das entradas [1]. O modelo contempla os modelos convencionais de um neurônio como um dos seus casos especiais; ou seja, a dinâmica caótica é apresentada como uma extensão natural dos modelos de neurônios não-caóticos. Em resumo o trabalho mostra a viabilidade da utilização de caos determinístico em redes neurais.

IV. METODOLOGIA

A. Problema do Caixeiro Viajante (TSP)

Utiliza-se amplamente nos experimentos dos mais diversos métodos de otimização o TSP, pois trata-se de um problema de fácil descrição e compreensão. Entretanto apresenta uma grande dificuldade na obtenção da solução por que é um problema NP-Difícil [10], apesar disso apresenta uma aplicabilidade ampla.

O TSP é um típico problema de otimização combinatória, ele pode ser entendido como uma busca pelo menor caminho fechado que visita cada cidade uma vez e apenas uma vez. A forma de problema de decisão do TSP é um NP-completo, por isso o grande interesse em heurísticas eficientes para resolvê-lo.

Existem muitas das heurísticas que utilizam o paradigma da computação neural ou noções relacionadas recentemente. A primeira abordagem para o TSP via redes neurais foi no trabalho de Hopfield e Tank (1985), que foi baseado na minimização de uma função de energia, e os mínimos locais devem corresponder a uma boa solução de uma instância do TSP [14]. No entanto, esta abordagem não garante a viabilidade, ou seja, nem todos os pontos de mínimos da função energia representam solução viável para o TSP.

A formulação matemática para o *problema do caixeiro viajante* é a seguinte:

$$\min C(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{7}$$

sujeito às restrições

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \text{para } j = 1, 2, ..., n, \tag{8}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \text{para } i = 1, 2, ..., n, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \quad \text{para } i, j = 1, 2, ..., n, \tag{10}$$

$$\tilde{x} \quad \text{forma um ciclo Hamiltoniano} \tag{11}$$

onde x_{ij} são as variáveis de decisão e c_{ij} é o custo de designar a distância da cidade i para a cidade j.

Outras técnicas clássicas são utilizadas para resolver o TSP, por exemplo, o método Simplex e os métodos exatos como a inserção mais económica e a inserção do nó mais distante. Além das técnicas existem outras maneiras para resolver o TSP, destacando-se os Algoritmos Genéticos [7] e as Redes Neurais Artificiais [6].

Uma técnica para a solução do TSP apresentada em [16], utiliza a resolução clássica do problema da Designação como uma fase preliminar da resolução do TSP, na qual considera-se a solução ótima para a matriz de custos do problema da Designação como uma solução inicial para o TSP. Se a solução inicial determina uma rota viável então esta já é a solução ótima do TSP, caso contrário, um método "Winner Takes All" com rede recorrente é aplicado para viabilizar a solução.

B. Algoritmo SOM para Resolver o TSP

O algoritmo SOM para ser aplicado na resolução do TSP deve sofrer uma adaptação para que se obtenha como resposta, após o final do processo, uma sequência de cidades a serem visitadas, ou seja, uma rota solução para o TSP. Para tal, é criada uma rede bidimensional com m neurônios de saída e entrada de dados bidimensional com $2m$ pesos dispostos em forma de circunferência ao redor das entradas do TSP em questão conforme exemplificado na Figura 2.

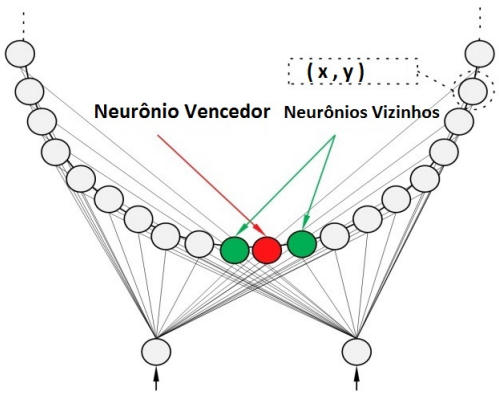

Figura 2. Distribuição dos neurônios.

Para simplificar a implementação os dados de entrada bem como a camada de pesos, estes foram convertidos em uma escala que varia no intervalo $[0, 1]$ utilizando a equação (12)

$$x_{escala} = \frac{(x - x_{min})}{(x - x_{max})} \tag{12}$$

onde x é o valor da entrada a ser convertida e x_{min} e x_{max} são respectivamente os valores de mínimo e máximo das entradas antes da conversão para a escala [2].

A arquitetura SOM consiste além das entradas bidimensionais das coordenadas espaciais das cidades e de um anel sobre a qual os neurônios estão espacialmente distribuídos como pode ser observado na Figura 2. Dispor os neurônios em forma de anel é a maneira mais citada na literatura de adaptar o algoritmo SOM para resolver o TSP.

Os pesos dos neurônios definem a posição do neurônio no espaço, e inicialmente são definidos do seguinte modo: assumindo um anel de m neurônios, os neurônios são igualmente posicionado em um círculo de raio unitário, utilizando a medida em graus, a posição do ângulo de dado neurônio que é igual a $360°$ dividido por m. Os dados de entrada (um conjunto de n cidades) são apresentados ao SOM em uma ordem aleatória e uma competição baseada em distância euclidiana é realizada entre os neurônios no anel. O neurônio vencedor é o neurônio i^* que possui a distância mínima para a cidade apresentada.

$i^* = \min_j \{\|x_i - w_j\|_2\}$, onde x_i é a coordenada do i-ésima entrada, w_j é a posição do neurônio j e $\| \bullet \|_2$ é a distância euclidiana. Assim, o neurônio vencedor i^*, bem como os neurônios vizinhos movem-se na direção desse neurônio,

usando a função de vizinhança:

$$f(\sigma, d) = \exp\left(\frac{-d^2}{\sigma^2}\right). \tag{13}$$

Os neurônios são atualizados segundo a seguinte função de atualização:

$$y_j^{novo} = y_j^{atual} + \alpha \cdot f(\sigma, d) \cdot (x_i - y_j^{atual}) \tag{14}$$

onde α e σ são a taxa de aprendizagem e o parâmetro de vizinhança respectivamente e $d = \min\{\|j - J\|, m - \|j - J\|\}$ é a distância cardinal calculada ao longo dos neurônios j e J onde $\|\cdot\|$ representa o valor absoluto. Taxa de aprendizagem tem característica dinâmica e diminui gradualmente durante o processo de aprendizado, assumindo valores no intervalo $(0; 1)$. Do mesmo modo, a função de vizinhança $f(\sigma, d)$ assume um valor muito grande no início do treinamento, e diminui lentamente, em abrangência, com o progresso do treinamento. Depois de muitas iterações os neurônios tendem a se aproximar das entradas e finalmente convergem para elas. Uma vez que todos neurônios convergiram para os valores de entrada, basta ler as coordenadas na ordem em que aparecem. O resultando dessa sequência constitui uma solução para o TSP em questão. A Figura 3 ilustra a evolução do algoritmo a partir do estado inicial do anel, (a) atingindo uma fase intermédia, após algumas iterações (b) e estabilizando no estado final (c).

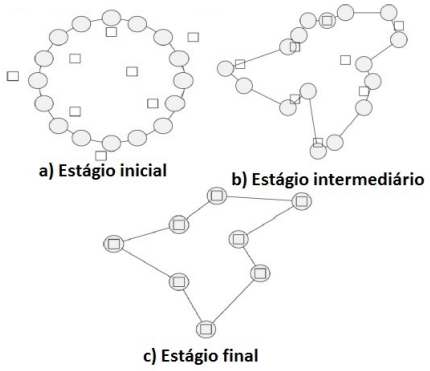

Figura 3. Exemplo.

O número de neurônios deve ser maior do que o número de cidades para evitar a oscilação de um neurônio e seus diferentes vizinhos. Uma boa alternativa é definir o número de neurônios m como sendo o dobro do número de vértices, ou seja, $m = 2n$.

C. Caos em Redes de Kohonen

A aplicação de caos determinístico em redes de Kohonen foi feita pela primeira vez por [5]. Os mapas auto-organizáveis convencionais utilizam unidades lineares:

$$y_{pj} = net_{pj} = \sum_{i=1}^{M} w_{ji} x_{pi} = w_j \;\text{ e }\; x_p = \|w_j\| \|x_p\| \cos\phi_{pj} \tag{15}$$

onde ϕ_{pj} é o ângulo entre o padrão de entrada x_p e o vetor de peso w_j. Tipicamente os vetores de entrada e de peso são unidades de comprimento, logo $y_{pj} = \cos\phi_{pj}$. Quando um padrão de entrada x_p é apresentado à rede, as unidades neurais competem para ter o privilégio de aprender e a unidade neural vencedora será aquela com a maior saída y_{pj}. Como $y_{pj} = \cos\phi_{pj}$, tem-se que a unidade neural vencedora é aquela cujo vetor de peso é mais próximo do padrão de entrada. O vetor de peso da unidade neural vencedora e os seus vizinhos são modificados de acordo com a seguinte regra, que gira no sentido w_j o padrão de entrada x_p:

$$w_j = \frac{w_j^{atual} + \lambda x_p}{\|w_j^{atual} + \lambda x_p\|} \tag{16}$$

A taxa de aprendizado λ, geralmente, diminui de forma linear ao longo da sessão de treinamento de um valor inicial $\lambda(0)$ para um valor final$\lambda_{\min}$.

O caos pode ser introduzido no mapa auto-organizável por substituição dos neurônios lineares da rede convencional por neurônios caóticos (6). Assim, quando um padrão de entrada x_p é apresentado para a rede, cada neurônio produz uma sequência de saídas $y_{pj}(n)$. Tal como acontece com o mapa auto-organizável convencional, os neurônios competem para ter o privilégio de aprender, mas, neste caso, **o neurônio vencedor é escolhido para ser o único com a maior produção média $\bar{y_{pj}}$ calculada sobre os elementos T da sequência de saída**:

$$\bar{y_{pj}} = \frac{1}{T} \sum_{n=1}^{T} y_{pj}(n). \tag{17}$$

Na prática, T é relativamente pequena e o $\bar{y_{pj}}$ média de saída depende do valor de entrada net_{pj} e o valor inicial $y_{pj}(0)$. Assim, a competição para aprender efetivamente se torna probabilística, com a unidade neural possuindo uma maior entrada líquida (ou seja, mais próximo do padrão de entrada), geralmente tendo maior probabilidade de ganhar. Tal como acontece com o mapa auto-organizável convencional, o neurônio vencedor e seus vizinhos ajustaram seus vetores de pesos de acordo com (16).

V. RESULTADO E DISCUSSÕES

O algoritmo SOM possui dois parâmetros adaptativos, o coeficiente de aprendizagem α_n e a função de vizinhança (normalmente a função gaussiana é adotada) e o coeficiente de variância σ_n propostos por Kohonen apresentam um decréscimo exponencial da seguinte forma:

$$\alpha_n = \alpha_0 \times \left(-\frac{n}{\tau_1}\right) \tag{18}$$

$$\sigma_n = \sigma_0 \times \left(-\frac{n}{\tau_2}\right) \tag{19}$$

em que $n = 1, 2, 3, \ldots$ é o número de iterações e τ_1 e τ_2 são constantes de tempo exponencial.

Uma adaptação ao algoritmo SOM para a solução do TSP, proposta por [3], adapta o algoritmo clássico para obtenção dos coeficientes de aprendizagem e raio de vizinhança pelas seguintes equações:

$$\alpha_n = \frac{1}{\sqrt[3]{n}} \tag{20}$$

$$\sigma_n = \sigma_{n-1} \times (1 - 0.01 \times n) \tag{21}$$

e ainda

$$\sigma_0 = \frac{l}{4 \times c} \tag{22}$$

em que l é o número de neurônios e c é uma constante, neste trabalho foram adotados os seguintes valores: $c = 8$ ou $c = 10$.

O algoritmo modificado segue os seguintes passos:

1) Inicialização: as entradas do algoritmo são formadas pelas coordenadas cartesianas das cidades, juntamente com o número de neurônios escolhidos para os cálculos. O número de neurônios devem ser maiores que o número de cidades e inicialmente são dispostos em um retângulo que envolve a posição geométrica das cidades.
2) Parâmetros de adaptação: α_n e σ_n calculados de acordo com as equações 18 e 19.
3) Avaliação da vizinhança: que é obtida pela avaliação dos parâmetros de vizinhança e aprendizagem através de sua atualização.
4) Competição: Cálculo do neurônio vencedor
5) Cooperação O neurônio vencedor tem sua vizinhança adaptada de acordo com a evolução do raio e coeficiente cognitivo descritos no passo 3 e acrescido da soma de um termo com característica caótica.
6) Adaptação: Retorne ao passo 2 enquanto $\alpha_n > \alpha_{\min}$.

Para o teste inicial foram escolhidas algumas instâncias do TSP com um número pequeno de cidades, descritas brevemente na Tabela I.

Tabela I
INSTÂNCIAS UTILIZADAS NOS EXPERIMENTOS COMPUTACIONAIS.

Instâncias	Número de Cidades	Tamanho da rota ótima
Eil 51	51	426
Eil 76	76	538
Eil 101	101	629
Berlin 52	52	7542
St70	70	675
Pcb 442	442	50778
Tsp 225	225	3916
Rat 99	99	1211
Rat 575	575	6773
Att 48	48	10628

Os resultados foram obtidos utilizando o algoritmo SOM modificado com característica caótica. Os resultados são apresentados na Tabela 2, cada instância é testada pelo menos 100 vezes calculado o resultado médio e apresentando a melhor solução.

Inicialmente os neurônios são dispostos em uma malha com formato parabólico, envolvendo as cidades e o algoritmo SOM/Caos modela essa malha ajustando a rota solução do TSP apresentada pelo algoritmo. Na Figura 4 apresenta a rota final obtida para a instância Eil51.

Tabela II
INSTÂNCIAS UTILIZADAS NOS EXPERIMENTOS COMPUTACIONAIS.

Instâncias	Melhor SOM	Erro %	Média SOM	Erro Médio %
Eil 51	428,872	0,67	436,717	2,52
Eil 76	544,754	1,26	556,754	3,49
Eil 101	644,266	2,43	661,105	5,10
Berlin 52	7544,37	0,03	7928,93	5,13
St 70	677,194	0,33	690,412	2,28
Pcb 442	52377,5	3,15	53110,6	4,59
Tsp 225	3980,44	1,65	4021,74	2,70
Rat 99	1224,31	1,10	1264,18	4,39
Rat 575	7016,33	3,59	7119,6	5,12
Att 48	10628	0,0	10812,35	1,73

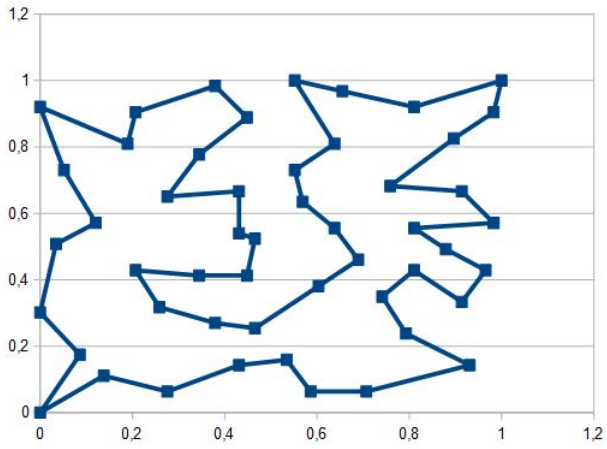

Figura 4. Rota final obtida para a instância Eil51.

VI. CONSIDERAÇÕES FINAIS

Neste trabalho faz-se a proposta de acrescentar a característica caótica ao algoritmo heurístico clássico de mapas auto-organizáveis, na tentativa de tornar essa busca mais semelhante aos processos naturais biológicos.

A implementação dos algoritmos foi programada na linguagem C++. Foi desenvolvido um software utilizado como uma ferramenta capaz de testar as instâncias do TSP extraídos da biblioteca TSPLIB95 e, resolvidos pelos algoritmos clássicos de mapas auto-organizáveis e mapas auto-organizáveis com introdução de caos e variações . O aplicativo produz uma solução em um tempo computacional que permite realizar testes para instâncias com milhares de cidades.

Os resultados mostram que o algoritmo converge e, os neurônios inicialmente dispostos em um retângulo (ou anel) envolvendo as cidades se aproximam de uma solução. O algoritmo tem um bom potencial e aliado ao refinamento 2-opt ou 3-opt [4] apresenta excelentes resultados.

Um relaxamento na vizinhança da rede SOM, ou seja, atualizar apenas os neurônios imediatamente vizinhos ao vencedor permitiu fazer uma observação interessante. Para instâncias menores que 100 cidades os resultados obtidos pela técnica

relaxada é muito semelhante ao obtido gerando uma rota aleatória e aplicando refinamento 3-opt, entretanto quando o número de cidades na instância é maior que 200 a técnica relaxada se mostra bem mais eficiente que a geração de uma rota aleatória e aplicação de refinamento 3-opt.

Os resultados se mostraram satisfatórios para continuar utilizando a técnica para a solução de outros problemas, como por exemplo, na solução do Problema do Roteamento de Veículos (PRV)[13]. No trabalho de [8] encontra-se o pseudo-código para a utilização da rede SOM na solução do PRV. Uma adaptação neste pseudo-código permite a implementação da técnica que utiliza caos, conforme apresentado neste trabalho, para resolver o PRV.

REFERÊNCIAS

[1] K. Aihara and M. Toyoda, *Chaotic neural networks*, Physics Letters A, v. 144, p. 333 – 340, 1990.

[2] Y. Bai and W. J. Z. Zhang, *An new self-organizing maps strategy for solving the traveling salesman problem*, Chaos, Solutions and Fractals, v. 28, p. 1082–1089, 2006.

[3] L. N. Castro and T. A. S. Masutti, *A self-organizing neural network using ideasfrom the immune system to solve the traveling salesman problem*, Information Sciences, v. 179, p. 1454–1468, 2009.

[4] G.B. Dantzig and J. Rasmer, *The truck dispatching problem*, Management Science, v. 6, n. 1, p. 80–91, 1959.

[5] A. A. Dingle and J. H. J. R. D. Andreae, *The chaotic self-organizing map*, Proc. of First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, p. 15–18, 1993.

[6] L. Fausett, *Fundamentals of Neural Networks*, New Jersey: Prentice Hall, 1994.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, New Jersey: Menlo Park, Addison-Wesley, CA, 1986.

[8] L.C.T. Gomes e F. Von Zuben, *Uma abordagem baseada em mapas auto-organizáveis de kohonen aplicada ao problema de roteamento de veículos*, Simpósio Brasileiro de Automação Inteligente. Canela: [s.n.], 2001. v. 5.

[9] S. S. Haykin, *Redes Neurais: Princípios e Prática*, Canada: Pearson Education Inc., 2001.

[10] R. Karp, *On the computational complexity of combinatorial problems*, Networks, v. 5, p. 45–68, 1975.

[11] T. Kohonen, *The self-organizing map*, Proceedings of the IEEE, v. 78, p. 1464– 1480, 1990.

[12] T. Kohonen, *Self-Organizing Maps*, New York: Springer, 2001.

[13] G. Laporte and M. P. J. S. F. Gendreaub, *Classical and modern heuristics for the vehicle routing problem*, International Transactions in Operational Research, v. 7, p. 285–300, 2000.

[14] K. Murty, *Linear and combinatorial programming*, Florida: Springer, 1985.

[15] E. Ott, *Chaos in Dynamical Systems*, Maryland: Cambridge University Press, 2002.

[16] P. H. Siqueira, *Uma Nova Abordagem na Resolução do Problema do Caixeiro Viajante*, Tese (Doutorado) — Universidade Federal do Paraná, 2005.