Uma resolução do problema do caixeiro-viajante por mapa auto-organizável com aprendizado winner takes all

Alexandre A. A. M. de Abreu [1]
Sanderson L. Gonzaga de Oliveira [1]
Wilian Soares Lacerda [1]

Resumo: Utilizou-se mapa auto-organizável unidimensional com aprendizado winner takes all para a resolução do problema do caixeiro-viajante. Nessa implementação, cada neurônio representa um vértice. Após a execução, a ordem dos neurônios indica a rota que representa a solução encontrada. Foram realizadas simulações com seis instâncias da base TSPLIB de tamanhos de 51 a 1379 vértices. Foram utilizadas uma taxa de aprendizagem de 0,8 e 3 mil épocas de treinamento em todas as execuções. Essa abordagem se mostrou eficiente e consistente. Todavia, as soluções encontradas não são melhores do que as obtidas por outros pesquisadores, por ter sido empregada, aqui, uma técnica de aprendizado com menor custo computacional, a saber, parâmetros iguais para todas as instâncias, e por não ter sido utilizado um algoritmo de otimização por busca local. Consequentemente, obteve-se uma solução com baixo custo computacional.

***Palavras-chave*:** Abordagem winner takes all. Aprendizado competitivo. Aprendizado não supervisionado. Heurística. Mapa auto-organizável. Problema do caixeiro-viajante. Rede neural artificial.

Abstract: *A one-dimensional self-organizing map with winner takes all learning was used to solve the traveling salesman problem. In this implementation, each neuron represents a vertex. After execution, the order of neurons indicates the route that represents the solution. Simulations were conducted with six instances of the TSPLIB base of sizes from 51 to 1379 vertices. A learning rate of 0.8 and 3000 training epochs in all executions were used. This approach proved to be efficient and consistent. However, the solutions are not better than those obtained by other researchers due to the use of a learning technique with lower computational cost, the use of the same parameters for all instances and the absence of an optimization algorithm for local search. Consequently, a solution with low computational cost was obtained.*

***Keywords*:** *Artificial neural network. Competitive learning. Heuristic. Self-organizing maps. Travelling salesman problem. Unsupervised Learning. Winner takes all approach.*

1 Introdução

O caixeiro-viajante pode ser definido como um vendedor que precisa visitar diversas cidades. Seu objetivo é realizar este trabalho percorrendo o menor caminho possível. De fácil compreensão, o problema do caixeiro-viajante é um dos mais estudados em otimização combinatória [1] e, ao mesmo tempo, um dos mais desafiadores.

O problema do caixeiro-viajante tem atraído a atenção de pesquisadores de diferentes áreas, como Inteligência Artificial, Engenharias, Matemática, Física, entre outras. Apesar de sua simplicidade de formulação, nesse problema é possível encontrar várias questões estudadas em otimização combinatória, e, por isso, ele tem sido utilizado como benchmark para avaliação de novos algoritmos e heurísticas, como busca tabu, algoritmos genéticos, simulated annealing e redes neurais artificiais, por exemplo [2].

[1]Departamento de Ciência da Computação, Universidade Federal de Lavras, UFLA, *Campus* Universitário - Caixa Postal 3037 - 37200-000 - Lavras (MG) - Brasil.
{alexandreabreu@posgrad.ufla.br}, {sanderson,lacerda@dcc.ufla.br}

http://dx.doi.org/10.5335/rbca.2015.4438

O desempenho dos métodos e das heurísticas para a solução do problema do caixeiro-viajante não é a única justificativa para o estudo do problema. Muitos problemas encontrados em situações reais são modelados de forma similar ao problema do tipo caixeiro-viajante ou suas variantes [2]. Por isso, novos métodos e heurísticas, assim como melhorias no resultado e na eficiência dos já existentes, têm sido propostos.

Como exemplo de problema que pode ser modelado de forma similar ao do caixeiro-viajante, tem-se o problema de produção, que consiste em realizar o sequenciamento de n tarefas em uma única máquina com o intuito de minimizar o tempo total de execução de todas as tarefas [2]. Cunha, Bonasser e Abrahão [2] exemplificam que "em linhas de montagem de componentes eletrônicos busca-se encontrar, por exemplo, o roteiro de mínima distância para um equipamento cuja tarefa é soldar todos os componentes de uma placa eletrônica". Outros exemplos podem ser encontrados em problemas que envolvem roteirização de veículos, os quais, muitas vezes, são definidos como problemas de um ou mais caixeiros viajantes que podem incluir restrições adicionais [2].

O problema do caixeiro-viajante pertence à classe de problemas NP-Difícil [3]. Por isso, propostas de soluções aproximadas com baixo custo computacional são utilizadas em situações práticas.

Em conjunto com heurísticas que apresentam soluções aproximadas, podem ser utilizados métodos de melhorias da solução encontrada, para se encontrar um ótimo local. Como exemplos desses métodos, tem-se os do tipo k-opt, utilizados por Lin e Kernighan [4], em que k arestas de um percurso são removidas e substituídas por outras k arestas, com vistas de reduzir o tamanho do ciclo. Em outras palavras, essas melhorias são obtidas por meio de uma técnica de busca local. Quanto maior o valor de k, melhor tende a ser a solução final encontrada; porém, o custo computacional também será maior [1]. Em situações práticas, geralmente, são utilizados os métodos 2-opt e 3-opt.

Brocki e Korzinek [5] realizaram uma implementação de mapa auto-organizável unidimensional com aprendizado winner takes most para a resolução do problema do caixeiro-viajante. Na implementação de Brocki e Korzinek [5], a solução encontrada pelo mapa auto-organizável foi otimizada pelo algoritmo de busca local 2-opt. Ainda, a taxa de aprendizado e a quantidade de épocas de treinamento foram variadas de acordo com a instância. Com isso, Brocki e Korzinek [5] obtiveram bons resultados em comparação com resultados obtidos por outros pesquisadores que aplicaram mapa auto-organizável no problema do caixeiro-viajante. O trabalho de Brocki e Korzinek [5] é um dos mais citados entre os que utilizaram mapa auto-organizável para resolução do problema do caixeiro-viajante; portanto, os resultados obtidos por esses autores [5] serão tomados como parâmetros para avaliação dos resultados obtidos neste trabalho.

Neste trabalho, utilizou-se um mapa auto-organizável unidimensional com aprendizado winner takes all para a resolução do problema do caixeiro-viajante. Um estudo comparativo entre mapa auto-organizável e outras técnicas de inteligência artificial aplicadas ao problema do caixeiro-viajante pode ser encontrado em Calado e Ladeira [6].

A seguir, a seção 2 apresenta a descrição do problema do caixeiro-viajante, métodos para a sua resolução e exemplos de aplicações práticas. Na seção 3, são abordados os conceitos de redes neurais artificiais, dois dos principais processos de aprendizagem e os conceitos de mapas auto-organizáveis. Na seção 4, são expostos os detalhes da utilização de mapa auto-organizável com aprendizado winner takes all aplicado ao problema do caixeiro-viajante, os resultados obtidos e uma análise desses resultados, de forma que possam ser comparados com os achados de Brocki e Korzinek [5]. Finalmente, as considerações finais e propostas de trabalhos futuros são apresentadas na seção 5.

2 O problema do caixeiro-viajante

Em termos mais técnicos, o problema do caixeiro-viajante consiste em determinar um ciclo de custo mínimo, isto é, cada vértice de um grafo ponderado deve ser percorrido uma só vez, e o primeiro vértice é também o último vértice da sequência, de forma que a soma dos pesos das arestas seja mínimo.

O problema do caixeiro-viajante pode ser descrito, formalmente, como encontrar um ciclo de custo mínimo

$$p(n) = min\left(\left(\sum_{i=1}^{n-1} d(v_i, v_{i+1})\right) + d(v_n, v_1)\right), \quad (1)$$

em um grafo $G(V, A)$ simples e ponderado, em que $V = \{v_1, v_2, ..., v_n\}$ é um conjunto de vértices, A é um conjunto de arestas e $d(v_i, v_j)$ é o custo da aresta (v_i, v_j).

O problema de decisão associado ao problema do caixeiro-viajante consiste em, dada uma constante K, verificar se existe um ciclo em que a soma dos pesos das arestas seja menor que K, ou seja, $p(n) < K$. Quando a distância entre dois vértices quaisquer v_i e v_j não depende de sentido, ou seja, $d(v_i, v_j) = d(v_j, v_i)$, o problema do caixeiro-viajante é denominado "simétrico"; caso contrário, se uma das distâncias não existe, ou $d(v_i, v_j) \neq d(v_j, v_i)$, o problema é considerado assimétrico. Alguns métodos e heurísticas podem ser desenvolvidos tanto para problemas simétricos quanto para problemas assimétricos. Na prática, um algoritmo específico para resolver o problema do caixeiro-viajante assimétrico é mais vantajoso do que um algoritmo desenvolvido tanto para problemas simétricos quanto para problemas assimétricos. Este trabalho tem como foco o problema de otimização descrito na equação (1).

O problema do caixeiro-viajante pode ser solucionado por métodos exatos ou por heurísticas. Métodos exatos, muitas vezes, baseiam-se em estratégias de branch and bound [7], em que as soluções são enumeradas sistematicamente e subconjuntos de soluções desvantajosas são desconsiderados. Apesar da existência de diversas funções limitadoras [2] para as estratégias de branch and bound, devido às características combinatórias do problema do caixeiro-viajante, métodos exatos podem ter aplicações limitadas e baixa escalabilidade. Mais detalhes sobre métodos exatos são apresentados nos trabalhos de Helsgaun [8] e Laporte [1]. Neste trabalho são abordadas estratégias heurísticas para a solução do problema do caixeiro-viajante.

O problema do caixeiro-viajante pode ser aplicado em diversas situações práticas, tais como:

1. na confecção de circuitos impressos, em que, geralmente, uma máquina precisa realizar uma sequência de furos em uma placa;
2. no cabeamento de redes para minimizar a quantidade de cabos utilizados;
3. no roteamento de veículos;
4. no controle de robôs;
5. na sequência de serviços a serem executados em uma máquina [1];
6. na cristalografia, em que um detector mede a intensidade dos raios-X refletidos em um material a partir de diversas posições [1].

3 Redes neurais artificiais

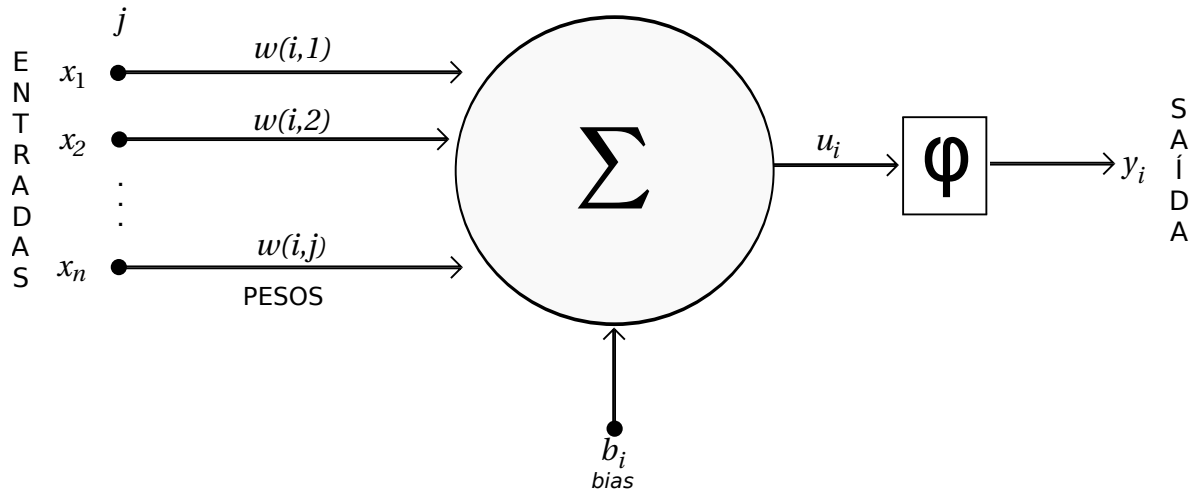
Uma rede neural artificial pode ser entendida como uma técnica computacional que apresenta um modelo matemático baseado na estrutura neural de organismos inteligentes. Em outra definição, ela é descrita como uma técnica de aproximação de funções por regressão não linear. Formalmente, é conceituada por Vellasco [9], da seguinte forma:

> *Uma rede neural é um sistema computacional constituído por unidades conhecidas como neurônios. Os neurônios são elementos processadores interligados, trabalhando em paralelo para desempenhar uma determinada tarefa. Os modelos RNAs formam uma importante técnica estatística não linear capaz de resolver uma gama de problemas de grande complexidade. Por isso, são modelos úteis em situações que não é possível definir explicitamente uma lista de regras. Em geral, isso acontece quando o ambiente gerador dos dados muda constantemente. As principais áreas de atuação são para classificação de padrões e previsão.*

O neurônio é a unidade fundamental de uma rede neural artificial. As entradas de um neurônio i podem ser representadas por x_j, com $1 \leq j \leq n$, que são as saídas dos neurônios da camada anterior, e n é a quantidade de entradas do neurônio i. Para cada x_j, há um peso associado ao neurônio i. Mais especificamente, os pesos da rede são representados por $w(i, j)$.

Os pesos representam a memória da rede, ou seja, a experiência adquirida como resultado das apresentações dos padrões à rede. São os pesos que combinam a não linearidade da rede neural. Em uma rede neural, os parâmetros a serem estimados são os pesos. Na Figura 1, é apresentado um esquema funcional do i-ésimo neurônio em uma rede, ou seja, sua estrutura esquemática interior.

Figura 1: Esquema do i-ésimo neurônio, adaptado de Haykin [10].



A combinação das entradas e dos pesos é representada por u_i em um neurônio. Um valor u_i corresponde à soma ponderada das entradas e é utilizado como entrada para uma função de ativação φ. A saída y_i do i-ésimo neurônio depende do nível de ativação aplicado ao neurônio pela função de ativação φ. O bias b_i é tratado como uma entrada de valor fixo, de modo que, durante o processo de treinamento, a atualização dos pesos aconteça para todos os parâmetros. Como a cada neurônio chegam todas as entradas, então, o bias aparecerá associado a uma entrada fixa $+1$ ou -1.

Uma característica importante das redes neurais artificiais é a sua capacidade de generalização. Isso significa que, após o treinamento, a rede é capaz de encontrar bons resultados para entradas que não foram apresentadas à rede durante seu treinamento. A capacidade de generalização da rede neural artificial é adquirida durante o seu aprendizado. Os dois principais processos de aprendizagem são: aprendizado supervisionado e aprendizado não supervisionado.

A seguir, na subseção 3.1, são apresentados os dois principais processos de aprendizagem utilizados no treinamento de redes neurais artificiais. Na subseção 3.2, são apresentados os conceitos de mapas auto-organizáveis e detalhes sobre as abordagens competitivas de aprendizado não supervisionado winner takes all e winner takes most.

3.1 Treinamento de redes neurais artificiais

O aprendizado supervisionado é um dos processos mais comuns de aprendizado utilizados em inteligência artificial. Utiliza-se o aprendizado supervisionado quando se tem disponível as amostras de entrada e o resultado esperado. Isso significa que é possível apresentar à rede um conjunto de entradas e o resultado que se deseja obter.

No aprendizado supervisionado, pode-se calcular o erro pela diferença entre a saída desejada e o resultado atual da rede. O aprendizado ocorre à medida que os pesos são atualizados por meio do erro obtido na iteração anterior do processo de treinamento da rede neural artificial. Como exemplo, tem-se o algoritmo backpropagation que pode ser utilizado para realizar o aprendizado supervisionado em redes neurais artificiais do tipo multilayer perceptron.

O aprendizado não supervisionado pode ser utilizado quando não se pode estabelecer um conjunto de amostras de entrada e nem resultado esperado. Nesse caso, o treinamento da rede neural artificial não pode ser feito por

um algoritmo que utiliza o erro para ajustar os pesos da rede.

Segundo Zuben e Attux [11], o aprendizado não supervisionado é predominante no cérebro humano. É sabido que as propriedades estruturais e fisiológicas das sinapses no córtex cerebral são influenciadas pelos padrões de atividade que ocorrem nos neurônios sensoriais. No entanto, em essência, nenhuma informação prévia acerca do conteúdo ou do significado do fenômeno sensorial está disponível. A implementação de modelos computacionais para ajuste de pesos sinápticos via treinamento não supervisionado deve recorrer apenas aos dados de entrada, tomados como amostras independentes de uma distribuição de probabilidade desconhecida [11].

No aprendizado não supervisionado, os pesos se agrupam de acordo com as características dos subconjuntos dos estímulos apresentados às regiões da camada de entrada [12]. Assim, os pesos se ajustam, gradualmente, de acordo com a entrada da rede neural. Benini [12] conclui que, quanto mais evidentes forem as particularidades da população de entrada, melhor será o aprendizado da rede e vice-versa.

Em ambos os tipos de treinamento, supervisionado e não supervisionado, existem alguns problemas que precisam ser observados durante o treinamento. Um exemplo, conhecido como overfitting, ocorre quando a rede é treinada excessivamente e perde sua capacidade de generalização, passando a memorizar os dados de entrada [9]. Em contrapartida, abortar o treinamento de uma rede neural artificial de forma prematura pode prejudicar seu desempenho, pois a rede terá valores sensíveis aos valores iniciais aleatórios, como pode ser observado no trabalho de Abreu [13], por exemplo. Nesse caso, tem-se uma situação de underfitting.

3.2 Mapas auto-organizáveis

Os mapas auto-organizáveis, ou redes de Kohonen [14], consistem em redes neurais artificiais interconectadas. Geralmente, um mapa auto-organizável tem dimensão 1 ou 2 e procura estabelecer e preservar noções de vizinhança dos elementos neuronais.

No caso de um mapa auto-organizável unidimensional, tem-se uma sequência ordenada de neurônios, e o número de pesos de cada neurônio é igual ao número de entradas. Devido à propriedade de auto-organização, os mapas auto-organizáveis podem ser aplicados a problemas de clusterização e ordenação espacial de dados [11].

Na Figura 2, tem-se um exemplo de topologia de um mapa auto-organizável unidimensional. Os valores de entrada e saída para o mapa auto-organizável são, respectivamente, $x_1, x_2, \cdots, x_{dim}$ e $y_1, y_2, \cdots, y_n$. Os pesos dos neurônios são representados por $w(i, j)$, que corresponde à conexão do peso da entrada x_j com o neurônio i, em que $1 \leq j \leq dim$, $1 \leq i \leq n$, dim é a quantidade de entradas e n é a quantidade de neurônios do mapa auto-organizável.

Uma abordagem competitiva de aprendizado não supervisionado muito utilizada é a winner takes all. Quando um vetor de entrada é passado para o mapa auto-organizável, executa-se o processo de aprendizagem. O neurônio vencedor será o que tiver pesos mais correlacionados aos valores de uma determinada entrada, definida aleatoriamente. Correlação é o produto escalar do vetor de entrada com os pesos sinápticos considerados [5]. Em outras palavras, o neurônio vencedor será aquele que apresentar maior valor para o produto escalar entre o vetor de entrada e o vetor que armazena seus pesos. Somente o neurônio vencedor terá seu peso sináptico atualizado nesse tipo de abordagem.

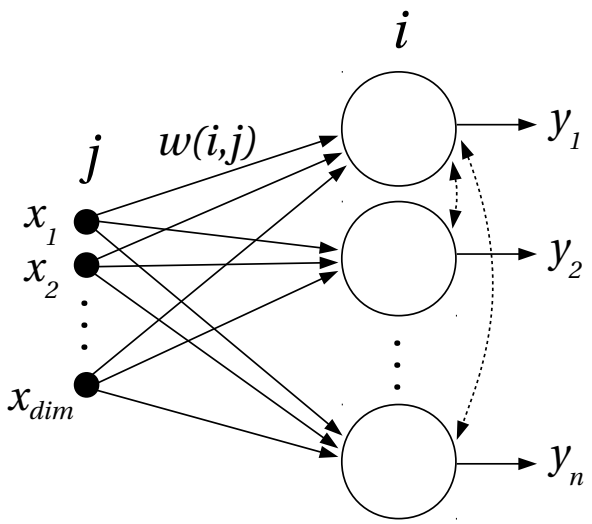
O processo de aprendizado não supervisionado para o mapa auto-organizável, utilizando-se a abordagem winner takes all, é descrito por

$$w(i, j) = w(i, j) + \eta(x_j - w(i, j)). \tag{2}$$

O processo de aprendizado consiste na atualização dos pesos $w(i, j)$ do neurônio vencedor i, com $0 \leq i \leq n$, em que n é a quantidade de neurônios, x_j é valor de entrada e a taxa de aprendizado é η, com $0 < \eta < 1$. A abordagem winner takes all pode ser implementada facilmente e possui um custo computacional pequeno, devido à sua simplicidade.

Uma variação do winner takes all que geralmente apresenta melhor convergência é o winner takes most, pois permite a ativação de neurônios que raramente seriam ativados [5]. Ativação é o processo de modificação dos pesos sinápticos de um neurônio. No winner takes most, mais de um neurônio têm seus pesos modificados em uma mesma iteração. Portanto, o vencedor e seus vizinhos serão ativados de acordo com uma função de vizinhança determinada.

Figura 2: Exemplo de topologia de mapa auto-organizável, adaptado de Zuben e Attux [11].



Uma forma simplificada de implementar o winner takes most, em um mapa auto-organizável de vizinhança simples, é acrescentando à equação (2) as equações de ativação

$$w(i-1,j) = w(i-1,j) + \eta(x_j - w(i-1,j)), \tag{3}$$

e

$$w(i+1,j) = w(i+1,j) + \eta(x_j - w(i+1,j)). \tag{4}$$

O algoritmo 1 é um exemplo dos passos para o treinamento de um mapa auto-organizável com aprendizado winner takes most. A taxa de aprendizado η é utilizada na linha **7** para modificar os pesos sinápticos do neurônio vencedor e seus vizinhos pelas equações (2), (3) e (4). No treinamento do mapa auto-organizável com aprendizado winner takes all, a linha 6 não é considerada, e apenas o neurônio vencedor terá seu peso modificado na linha 7.

Algoritmo 1: treinamento de um mapa auto-organizável.

Entrada: um vetor em que as entradas são os valores utilizados no treinamento da rede neural, número máximo de iterações *iter_max* e taxa de aprendizado η.
Saída: rede neural de Kohonen treinada.

```
1  iter ← 1;
2  atribua pesos aleatórios aos neurônios;
3  enquanto ( iter ≤ iter_max ) faça
4      selecione um valor aleatório do conjunto de treinamento;
5      encontre o neurônio vencedor que apresentar maior correlação ao valor selecionado;
6      encontre os neurônios vizinhos do neurônio vencedor;
7      os pesos sinápticos dos neurônios são modificados, considerando-se a taxa de
       aprendizado η;
8      reduza o valor de η;
9      iter ← iter + 1;
10 fim-enquanto;
11 retorna rede neural de Kohonen treinada.
```

4 Mapa auto-organizável aplicado ao problema do caixeiro-viajante

Utilizou-se um mapa auto-organizável unidimensional de vizinhança simples, ou seja, o neurônio i é conectado ao neurônio $i-1$ e ao neurônio $i+1$. Um par (x, y) representa as coordenadas de cada cidade como

entrada de cada neurônio. Portanto, para cada instância, utilizou-se um mapa auto-organizável unidimensional com quantidade de neurônios igual à quantidade de cidades da instância.

Assim como os neurônios de uma rede *Adaptive Linear Element* (ADALINE) [15], os neurônios do mapa auto-organizável utilizado não têm função de ativação. Realizou-se o treinamento do mapa auto-organizável pela abordagem winner takes all, conforme mostrado no algoritmo 1. Ao final de cada época de treinamento, o valor de η foi reduzido em 2% para refinar o aprendizado no decorrer do treinamento do mapa auto-organizável.

O mapa auto-organizável tenta encontrar uma ordenação nas cidades com a intenção de que as cidades próximas tenham ligação entre si no caminho final encontrado. Após o treinamento do mapa auto-organizável, cada neurônio representará uma cidade, e a ordem dos neurônios indicará o caminho solução para o problema do caixeiro-viajante. Caso os valores obtidos pela rede neural não representem exatamente as coordenadas de uma cidade, é feito um ajuste nos valores dos neurônios. Esse ajuste se dá pelo arredondamento para os valores mais próximos que correspondem às coordenadas de uma cidade que ainda não conta com um neurônio no mapa auto-organizável. Por exemplo, se o valor obtido pelo i-ésimo neurônio for $(x_i = 2, y_i = 3,94)$, o seu valor será ajustado para $(x_i = 2, y_i = 4)$, caso nenhum outro neurônio represente a cidade da coordenada $(2, 4)$. A abordagem utilizada pode ser aplicada tanto no problema do caixeiro-viajante simétrico quanto no problema do caixeiro-viajante assimétrico.

Como parâmetros, foram utilizados $\eta = 0,8$ e 3 mil épocas de treinamento para todas as instâncias. Utilizou-se uma grande quantidade de épocas de treinamento na intenção de se obter melhores soluções para o problema do caixeiro-viajante. Caso seja utilizada uma técnica de otimização, como busca local, por exemplo, podem ser utilizadas menos épocas de treinamento. Com isso, o custo computacional de treinamento do mapa auto-organizável será ainda menor.

O computador utilizado na implementação e execução dos algoritmos continha um processador Intel Core i5 3,4GHz com 6144 KB de memória cache e 8GB de memória principal. O sistema operacional utilizado foi o Linux Slackware 14.0, e a linguagem de programação empregada foi C++, em conjunto com o framework KNNL [16]. Esse framework, desenvolvido em *C++*, é gratuito e inclui diversos métodos que auxiliam na implementação de mapas auto-organizáveis.

Executou-se a implementação do mapa auto-organizável dez vezes para cada instância mostrada na Tabela 1, em que também são apresentados valores de desvio padrão σ para as soluções encontradas e para os tempos de execução em segundos. A solução encontrada é a soma das distâncias euclidianas entre as cidades no caminho traçado. As instâncias utilizadas nos experimentos foram obtidas na base de dados TSPLIB [17], e o número ao final do nome da instância refere-se ao seu tamanho.

Tabela 1: Resultados dos experimentos na resolução do problema do caixeiro-viajante por mapa auto-organizável com aprendizado winner takes all.

	Solução encontrada (distância)		**Tempo de execução (s)**	
Instância	**Média**	σ	**Média**	σ
eil51	1614	4,53E+01	0,46072	7,28E-03
eil101	3687	1,42E+02	1,39105	1,25E-02
a280	32953	6,90E+02	1,89803	2,56E-02
rat783	17900	4,74E+03	6,86503	4,81E-02
u1060	7760000	1,55E+05	12,02647	2,00E-02
nrw1379	1690000	2,25E+04	20,43462	1,41E-01

Observa-se que, mesmo para as maiores instâncias mostradas na Tabela 1, o tempo necessário para se

encontrar uma solução razoável é pequeno. As soluções para o problema do caixeiro-viajante encontradas não são melhores do que as soluções obtidas por Brocki e Korzinek [5], reproduzidas na Tabela 2, devido à utilização, neste trabalho, da técnica de aprendizado winner takes all e de parâmetros iguais para todas as instâncias. Ainda, os resultados obtidos por Brocki e Korzinek [5] foram otimizados pelo algoritmo de busca local 2-opt. Também, a taxa de aprendizado e a quantidade de épocas de treinamento foram variadas de acordo com a instância, e os experimentos foram realizados, por Brocki e Korzinek [5], em um computador com processador AMD Athlon 64-bits 3500+.

Tabela 2: Resultados dos experimentos na resolução do problema do caixeiro-viajante obtidos por Brocki e Korzinek [5].

Instâncias	Resultado médio (distância)	Tempo médio de execução (s)
eil51	444	0,068
eil101	662	0,127
tsp225	4193	0,302
pcb442	56634	0,703
pr1002	278481	2,425
pr2392	418739	12,965
rand100	4051	0,131
rand500	8888	0,824
rand1000	12483	2,311

5 Conclusões

A utilização de mapa auto-organizável unidimensional com aprendizado winner takes all mostrou-se eficiente e consistente para solucionar o problema do caixeiro-viajante. Essa abordagem é simples de ser implementada e apresentou custo computacional pequeno. Isso porque, em cada iteração do treinamento do mapa auto-organizável com winner takes all, são realizadas apenas uma subtração, uma multiplicação e uma soma para atualizar o peso do neurônio ativado, como descrito na equação (2). No treinamento do mapa auto-organizável com winner takes most, como na implementação de Brocki e Korzinek [5], essas operações são feitas para mais de um neurônio em uma mesma iteração do treinamento do mapa auto-organizável. Portanto, em uma mesma iteração, o custo de alteração dos pesos sináptico utilizando-se a abordagem winner takes most é maior do que o custo de alteração dos pesos utilizando-se a abordagem winner takes all.

Como continuação deste trabalho, pode-se empregar uma técnica de otimização por busca local, como o 2-opt, para melhorar as soluções encontradas, com a desvantagem de maior custo computacional. Assim, a quantidade de épocas de treinamento poderá ser menor do que a utilizada neste trabalho, e o custo computacional de treinamento do mapa auto-organizável será ainda menor.

Pode-se, ainda, variar os parâmetros de acordo com a instância, como foi realizado por Faigl et al. [18]. Finalmente, experimentos poderão ser conduzidos com instâncias iguais e em um mesmo computador, para que seja possível comparar as soluções encontradas para o problema do caixeiro-viajante e os custos computacionais das abordagens winner takes all e winner takes most.

Agradecimentos

Este trabalho foi realizado com o apoio da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Fapemig). Os autores agradecem, também, aos revisores anônimos pelos comentários e pelas sugestões para a melhoria deste artigo.

Referências

[1] LAPORTE, G. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, v. 59, n. 2, p. 231 – 247, 1992.

[2] CUNHA, C. B.; BONASSER, U. O.; ABRAHÃO, F. T. M. Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. In: *Anais do XVI Congresso da Anpet – Associação Nacional de Pesquisa e Ensino em Transportes*. Natal, RN: ANPET, 2002. v. 2, p. 105–117. Disponível em: <http://sites.poli.usp.br/ptr/ptr/docentes/cbcunha/files/2-opt_TSP_Anpet_2002_CBC.pdf>. Acesso em: 8 Nov. 2014.

[3] KARP, R. Reducibility among combinatorial problems. In: MILLER, R.; THATCHER, J.; BOHLINGER, J. (Ed.). *Complexity of Computer Computations*. Springer US, 1972, (The IBM Research Symposia Series). p. 85–103.

[4] LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, v. 21, p. 498–516, 1973.

[5] BROCKI, L.; KORZINEK, D. Kohonen self-organizing map for the traveling salesperson problem. In: *Recent Advances in Mechatronics*. Springer Berlin Heidelberg, 2007. p. 116–119.

[6] CALADO, F. M.; LADEIRA, A. P. Problema do caixeiro viajante: Um estudo comparativo de técnicas de inteligência artificial. *e-Xacta*, v. 4, p. 5–16, Set. 2001.

[7] LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica*, The Econometric Society, v. 28, n. 3, p. 497–520, 1960.

[8] HELSGAUN, K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, v. 126, n. 1, p. 106 – 130, 2000.

[9] VELLASCO, M. M. B. R. *Redes Neurais Artificiais*. 2007. Disponível em: <http://www2.ica.ele.puc-rio.br/Downloads/33/ICA-introdu%C3%A7%C3%A3o%20RNs.pdf>. Acesso em: 15 Abr. de 2013.

[10] HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Upper Saddle River, NJ: Prentice Hall, 1998.

[11] ZUBEN, F. J. V.; ATTUX, R. R. F. *Rede Neural de Kohonen e Aprendizado Não-Supervisionado*. 2007. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s07/topico8_07.pdf>. Acesso em: 5 Abr. 2013.

[12] BENINI, F. A. V. *Rede Neural Recorrente com Perturbação Simultânea Aplicada no Problema do Caixeiro Viajante*. 68 p. Dissertação (Mestrado) — USP, São Carlos, SP, 2008.

[13] ABREU, A. A. A. M. de. *Reduções de largura de banda e de profile de matrizes por mapa auto-organizável*. 125 p. Dissertação (Mestrado) — Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, MG, Novembro 2014.

[14] KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Springer-Verlag, v. 43, n. 1, p. 59–69, 1982.

[15] WIDROW, B.; HOFF, M. E. Adaptive switching circuits. In: *Western Electronic Show and Convention, Part 4*. Institute of Radio Engineers: Convention Record, 1960. p. 96–104.

[16] HABDANK-WOJEWODZKI, S.; RYBARSKI, J. *KNNL, C++ Kohonen Neural Network Library*. 2006. Disponível em: <http://knnl.sourceforge.net/>. Acesso em: 8 Set. 2013.

[17] REINELT, G. TSPLIB – A Traveling Salesman Problem Library. *ORSA Journal On Computing*, v. 3, p. 376–384, 1991.

[18] FAIGL, J. et al. An application of the self-organizing map in the non-euclidean traveling salesman problem. *Neurocomputing*, v. 74, n. 5, p. 671 – 679, 2011.