We are IntechOpen, the world’s leading publisher of Open Access books
Built by scientists, for scientists

4,000 Open access books available
116,000 International authors and editors
120M Downloads

154 Countries delivered to
TOP 1% Our authors are among the most cited scientists
12.2% Contributors from top 500 universities

WEB OF SCIENCE™
Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com
A Recurrent Neural Network to Traveling Salesman Problem

Paulo Henrique Siqueira, Sérgio Scheer, and Maria Teresinha Arns Steiner

Federal University of Paraná
Curitiba, Paraná, Brazil

1. Introduction

One technique that uses Wang’s Recurrent Neural Networks with the “Winner Takes All” principle is presented to solve two classical problems of combinatorial optimization: Assignment Problem (AP) and Traveling Salesman Problem (TSP).

With a set of appropriate choices for the parameters in Wang’s Recurrent Neural Network, this technique appears to be efficient in solving the mentioned problems in real time. In cases of solutions that are very close to each other or multiple optimal solutions to Assignment Problem, the Wang’s Neural Network does not converge. The proposed technique solves these types of problems by applying the “Winner Takes All” principle to Wang’s Recurrent Neural Network, and could be applied to solve the Traveling Salesman Problem as well. This application to the Traveling Salesman Problem can easily be implemented, since the formulation of this problem is the same that of the Assignment Problem, with the additional constraint of Hamiltonian circuit.

Comparisons between some traditional ways to adjust parameters of Recurrent Neural Networks are made, and some proposals concerning to parameters with dispersion measures of the cost matrix coefficients to the Assignment Problem are shown. Wang’s Neural Network with principle Winner Takes All performs only 1% of the average number of iterations of Wang’s Neural Network without this principle. In this work 100 matrices with dimension varying of 3×3 to 20×20 are tested to choose the better combination of parameters to Wang’s recurrent neural network.

When the Wang’s Neural Network presents feasible solutions for the Assignment Problem, the "Winner Takes All" principle is applied to the values of the Neural Network’s decision variables, with the additional constraint that the new solution must form a feasible route for the Traveling Salesman Problem.

The results from this new technique are compared to other heuristics, with data from the TSPLIB (Traveling Salesman Problem Library). The 2-opt local search technique is applied to the final solutions of the proposed technique and shows a considerable improvement of the results. The results of problem “dantzig42” of TSPLIB and an example with some iterations of technique proposed in this work are shown.

This work is divided in 11 sections, including this introduction. In section 2, the Assignment Problem is defined. In section 3, the Wang’s recurrent neural network is presented and a...
Travelling Salesman Problem

problem with multiple optimal solutions is shown. In section 4, the technique based on the “Winner takes all” principle is presented and an example of application to Assignment Problem is shown. In section 5, some alternatives for parameters of Wang’s neural network to Assignment Problem are presented. In section 6 the results to 100 matrices are shown. In Section 7, it is presented the formulation of Traveling Salesman Problem. In section 8 the application of Wang’s neural network with “Winner Takes all” is shown with five examples of TSPLIB. In Section 9, results to others problems of TSPLIB are compared to the ones obtained through other techniques. Findings are presented in section 10, and section 11 contains the references.

2. The assignment problem

The objective of this problem is assigning a number of elements to the same number of positions, and minimizing the linear cost function. This problem is known in literature as Linear Assignment Problem or problem of Matching with Costs (Ajhuha et al., 1993; Siqueira et al., 2004), and can be formulated as follows:

Minimize $C = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}x_{ij}$  

Subject to $\sum_{i=1}^{n} x_{ij} = 1, j = 1, 2, ..., n$  

$\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, ..., n$  

$x_{ij} \in \{0, 1\}, i, j = 1, 2, ..., n$  

where $c_{ij}$ and $x_{ij}$ are, respectively, the cost and the decision variable associated to the assignment of element $i$ to position $j$. The usual representation form of $c$ in the Hungarian method is the matrix form. When $x_{ij} = 1$, element $i$ is assigned to position $j$.

The objective function (1) represents the total cost to be minimized. The set of constraints (2) and (3) guarantees that each element $i$ will be assigned for exactly one position $j$. The set (4) represents the zero-one integrality constraints of the decision variables $x_{ij}$. The set of constraints (4) can be replaced by:

$x_{ij} \geq 0, i, j = 1, 2, ..., n.$  

Beyond traditional techniques, as the Hungarian method and the Simplex method, some ways of solving this problem has been presented in the last years. In problems of great scale, i.e., when the problem’s cost matrix is very large, the traditional techniques do not reveal efficiency, because the number of restrictions and the computational time are increased. Since the Hopfield and Tank’s publication (Hopfield & Tank, 1985), lots of works about the use of Neural Networks to solving optimization problems had been developed (Matsuda, 1998; Wang, 1992 and 1997). The Hopfield’s Neural Network, converges to the optimal solution of any Linear Programming problem, in particular for the AP.
Wang, 1992, considered a Recurrent Neural Network to solve the Assignment Problem; however, the necessary number of iterations to achieve an optimal solution is increased in problems of great scale. Moreover, in problems with solutions that are very close to each other or multiple optimal solutions, such network does not converge.

In this work, one technique based on the "Winner Takes All" principle is presented, revealing efficiency solving the problems found in the use of Wang’s Recurrent Neural Network. Some criteria to adjust the parameters of the Wang’s Neural Network are presented: some traditional ways and others that use dispersion measures between the cost matrix’ coefficients.

3. The Wang’s recurrent neural network to assignment problem

Consider the \( n^2 \times 1 \) vectors \( c^T \), that contains all the rows of matrix \( c; x \), that contains the decision elements \( x_{ij} \) and \( b \), that contains the number “1” in all positions. The matrix form of the problem described in (1)-(4) is due Hung & Wang, 2003:

\[
\text{Minimize } C = c^T x \\
\text{Subject to } Ax = b
\]

where matrix \( A \) has the following form:

\[
A = \begin{bmatrix}
I & I & \cdots & I \\
B_1 & B_2 & \cdots & B_n
\end{bmatrix} \in \mathbb{R}^{2n \times n^2}
\]

where \( I \) is an \( n \times n \) identity matrix, and each \( B_i \) matrix, for \( i = 1, 2, \ldots, n \), contains zeros, with exception of \( i \)th row, that contains the number “1” in all positions.

The Recurrent Neural Network proposed by Wang (published in Wang, 1992; Wang, 1997; and Hung & Wang, 2003) is characterized by the following differential equation:

\[
\frac{d u_{ij}(t)}{dt} = -\eta \sum_{k=1}^{n} x_{ik}(t) - \eta \sum_{l=1}^{n} x_{lj}(t) + \eta \theta_{ij} - \lambda c_{ij} e^{-\tau},
\]

where \( x_{ij} = g(u_{ij}(t)) \) and the equilibrium state of this Neural Network is a solution for the Assignment Problem, where \( g \) is the sigmoid function with a \( \beta \) parameter, i.e.,

\[
g(u) = \frac{1}{1 + e^{-\beta u}}.
\]

The threshold is defined as the \( n^2 \times 1 \) vector \( \theta = A^T b = (2, 2, \ldots, 2) \). Parameters \( \eta, \lambda \) and \( \tau \) are constants, and empirically chosen (Hung & Wang, 2003), affecting the convergence of the network. Parameter \( \eta \) serves to penalize violations in the problem’s constraints’ set, defined by (1)-(4). Parameters \( \lambda \) and \( \tau \) control the objective function’s minimization of the Assignment Problem (1). The Neural Network matrix form can be written as:

\[
\frac{d u(t)}{dt} = -\eta (Wx(t) - \theta) - \lambda c e^{-\tau},
\]
where \( x = g(u(t)) \) and \( W = A^T A \). The convergence properties of Wang’s Neural Network are demonstrated in Wang 1993, 1994 & 1995, and Hung & Wang, 2003.

### 3.1 Multiple optimal solutions and closer optimal solutions

In some cost matrices, the optimal solutions are very closer to each other, or in a different way, some optimal solutions are admissible. The cost matrix \( c \) given below:

\[
\begin{bmatrix}
0.6 & 0.9 & 1 & 2.7 & 0 & 1 & 0.1 & 0 \\
1 & 0 & 0.3 & 0.4 & 5.5 & 0.3 & 3.4 & 5.5 \\
0 & 2.2 & 0.2 & 0.6 & 0.2 & 0 & 0 & 0.2 \\
0.3 & 0.2 & 0 & 0.1 & 1.5 & 0 & 0.1 & 1.5 \\
0.6 & 0.9 & 1 & 2.7 & 0 & 1 & 0.1 & 0 \\
0 & 2.2 & 0.2 & 0.6 & 0.2 & 0 & 0.2 & 0 \\
0.6 & 0.9 & 1 & 2.7 & 0 & 1 & 0.1 & 0 \\
0 & 2.2 & 0.2 & 0.6 & 0.2 & 0 & 0.2 & 0 \\
\end{bmatrix}
\]

Equation (11)

has the solutions \( x^* \) and \( \tilde{x} \) given below:

\[
x^* = \begin{bmatrix}
0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0.5 & 0 & 0.25 & 0 & 0 & 0.25 & 0 \\
0 & 0 & 0.5 & 0 & 0 & 0.5 & 0 \\
0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\
0.5 & 0 & 0.25 & 0 & 0 & 0.25 & 0 \\
0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\quad \text{and} \quad
\tilde{x} = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}
\]

where \( x^* \) is found after 4,715 iterations using the Wang’s neural network, and \( \tilde{x} \) is an optimal solution.

The solution \( x^* \) isn’t feasible, therefore, some elements \( x^*_ij \) violate the set of restrictions (4), showing that the Wang’s neural network needs adjustments for these cases. The simple decision to place unitary value for any one of the elements \( x^*_ij \) that possess value 0.5 in solution \( x^* \) can become unfeasible or determine a local optimal solution. Another adjustment that can be made is the modification of the costs’ matrix’ coefficients, eliminating ties in the corresponding costs of the variable \( x^*_ij \) that possess value different from “0” and “1.” In this way, it can be found a local optimal solution when the modifications are not made in the adequate form. Hence, these decisions can cause unsatisfactory results.

### 4. Wang’s neural network and “Winner Takes All” principle to assignment problem

The method considered in this work uses one technique based on the “Winner Takes All” principle, speeding up the convergence of the Wang’s Neural Network, besides correcting eventual problems that can appear due the multiple optimal solutions or very closer optimal solutions (Siqueira et al., 2005).
The second term of equation (10), $Wx(t) - \theta$, measures the violation of the constraints to the Assignment Problem. After a certain number of iterations, this term does not suffer substantial changes in its value, evidencing the fact that problem's restrictions are almost satisfied. At this moment, the method considered in this section can be applied. When all elements of $x$ satisfy the condition $Wx(t) - \theta \leq \phi$, where $\phi \in [0, 2]$, the proposed technique can be used in all iterations of the Wang’s Neural Network, until a good approach of the Assignment Problem be found. An algorithm of this technique is presented as follows:

**Step 1:** Find a solution $x$ of the AP, using the Wang’s recurrent neural network. If $Wx(t) - \theta \leq \phi$, then go to **Step 2**. Else, find another solution $x$.

**Step 2:** Given the matrix of decision $x$, after a certain number of iterations of the Wang’s recurrent neural network. Let the matrix $\overline{x}$, where $\overline{x} = x$, $m = 1$, and go to **step 3**.

**Step 3:** Find the $m$th biggest array element of decision, $\overline{x}_{kl}$. The value of this element is replaced by the half of all elements sum of row $k$ and column $l$ of matrix $x$, or either,

$$\overline{x}_{kl} = \frac{1}{2} \left( \sum_{j=1}^{n} x_{kj} + \sum_{i=1}^{n} x_{ij} \right).$$  \hspace{1cm} (12)

The other elements of row $k$ and column $l$ become nulls. Go to **Step 4**.

**Step 4:** If $m \leq n$, makes $m = m + 1$, and go to **step 3**. Else, go to **step 5**.

**Step 5:** If a good approach to an AP solution is found, stop. Else, make $x = \overline{x}$, execute the Wang’s neural network again and go to **Step 2**.

### 4.1 Illustrative example

Consider the matrix below, which it is a partial solution of the Assignment Problem defined by matrix $C$, in (13), after 14 iterations of the Wang’s recurrent neural network. The biggest array element of $\overline{x}$ is in row 1, column 7.

$$\overline{x} = \begin{pmatrix}
0.0808 & 0.0011 & 0.0168 & 0.1083 & 0.0514 & 0.0033 & \ast 0.422 & 0.3551 \\
0.0056 & 0.2827 & 0.0168 & 0.1525 & 0.1484 & 0.1648 & 0.1866 & 0 \\
0.1754 & 0.0709 & 0.0688 & 0.3449 & 0 & 0.3438 & 0.0425 & 0.0024 \\
0.1456 & 0.2412 & 0.2184 & 0.0521 & 0.1131 & 0.0747 & 0.0598 & 0.1571 \\
0.0711 & 0 & 0.2674 & 0.272 & 0.3931 & 0.0024 & 0.0306 & 0.0061 \\
0.2037 & 0.2823 & 0.2956 & 0.0366 & 0 & 0.0025 & 0.0136 & 0.2186 \\
0.1681 & 0.174 & 0.1562 & 0 & 0.3053 & 0.2016 & 0.0369 & 0.0144 \\
0.1142 & 0.0031 & 0.0138 & 0.0829 & 0.0353 & 0.2592 & 0.251 & 0.2907
\end{pmatrix}$$

$$c = \begin{pmatrix}
1.4 & 6.1 & 3.1 & 0.4 & 2.2 & 4.4 & 0.1 & 0 \\
1.3 & 0.2 & 3.3 & 0.2 & 1.2 & 0.4 & 1.5 & 8.2 \\
1.7 & 2.9 & 2.8 & 0.1 & 9.8 & 0.4 & 4.2 & 6.8 \\
0.5 & 0 & 0 & 1.0 & 1.1 & 0.9 & 2.4 & 0.9 \\
2.8 & 7.7 & 1.2 & 0.5 & 0.9 & 6 & 4.6 & 5.9 \\
0.5 & 0.2 & 0 & 1.8 & 8.5 & 4.9 & 4.4 & 0.9 \\
0.6 & 0.7 & 0.7 & 6.9 & 0.1 & 0 & 3.2 & 3.8 \\
1.4 & 5.4 & 3.7 & 1.1 & 3 & 0 & 1.3 & 0.7
\end{pmatrix}$$  \hspace{1cm} (13)

www.intechopen.com
After the update of this element through equation (12), the result given below is found. The second biggest element of $x$ is in row 5, column 5.

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1.0412 & 0 \\
0.0056 & 0.2827 & 0.0168 & 0.1525 & 0.1484 & 0.1648 & 0 & 0 \\
0.1754 & 0.0709 & 0.0688 & 0.3449 & 0 & 0.3438 & 0 & 0.0024 \\
0.1456 & 0.2412 & 0.2184 & 0.0521 & 0.1131 & 0.0747 & 0 & 0.1571 \\
0.0711 & 0 & 0.2674 & 0.272 & *0.393* & 0.0024 & 0 & 0.0061 \\
0.2037 & 0.2823 & 0.2956 & 0.0366 & 0 & 0.0025 & 0 & 0.2186 \\
0.1681 & 0.174 & 0.1562 & 0 & 0.3053 & 0.2016 & 0 & 0.0144 \\
0.1142 & 0.0031 & 0.0138 & 0.0829 & 0.0353 & 0.2592 & 0 & 0.2907
\end{pmatrix}
\]

After the update of all elements of $x$, get the following solution:

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1.0412 & 0 \\
0 & 1.0564 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1.0491 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0446 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1.0533 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0544 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0473
\end{pmatrix}
\]

This solution is presented to the Wang’s neural network, and after finding another $x$ solution, a new $x$ solution is calculated through the “Winner Takes All” principle. This procedure is made until a good approach to feasible solution is found. In this example, after more 5 iterations, the matrix $x$ presents one approach of the optimal solutions:

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0.9992 & 0 \\
0 & 0.9996 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.9994 & 0 & 0 & 0 & 0 \\
0.999 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.9985 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9999 \\
0 & 0 & 1.0003 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.0003 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9999
\end{pmatrix}
\]

Two important aspects of this technique that must be taken in consideration are the following: the reduced number of iterations necessary to find a feasible solution, and the absence of problems related to the matrices with multiple optimal solutions. The adjustments of the Wang’s neural network parameters are essential to guarantee the convergence of this technique, and some forms of adjusting are presented on the next section.
5. The parameters of Wang's recurrent neural network

In this work, the used parameters play basic roles for the convergence of the Wang's neural network. In all the tested matrices, \( \eta = 1 \) had been considered, and parameters \( \tau \) and \( \lambda \) had been calculated in many ways, described as follows (Siqueira et al., 2005).

One of the most usual forms to calculate parameter \( \lambda \) for the AP can be found in Wang, 1992, where \( \lambda \) is given by:

\[
\lambda = \eta / C_{\text{max}}, \tag{14}
\]

where \( C_{\text{max}} = \max \{c_{ij}; i, j = 1, 2, ..., n\} \).

The use of dispersion measures between the \( c \) matrix coefficients had revealed to be efficient adjusting parameters \( \tau \) and \( \lambda \). Considering \( \delta \) as the standard deviation between the \( c \) cost matrix' coefficients, the parameter \( \lambda \) can be given as:

\[
\lambda = \eta / \delta \tag{15}
\]

Another way to adjust \( \lambda \) is to consider it a vector, defined by:

\[
\lambda = \eta \left( \frac{1}{\delta_1}, \frac{1}{\delta_2}, ..., \frac{1}{\delta_n} \right), \tag{16}
\]

where \( \delta_i \), for \( i = 1, 2, ..., n \), represents the standard deviation of each row of the matrix \( c \). Each element of the vector \( \lambda \) is used to update the corresponding row of the decision matrix.

This form to calculate \( \lambda \) revealed to be more efficient in cost matrices with great dispersion between its values, as shown by the results presented in the next section.

A variation of the expression (14), that uses the same principle of the expression (16), is to define \( \lambda \) by the vector:

\[
\lambda = \eta \left( \frac{1}{c_{1, \text{max}}}, \frac{1}{c_{2, \text{max}}}, ..., \frac{1}{c_{n, \text{max}}} \right), \tag{17}
\]

where \( c_{i, \text{max}} = \max \{c_{ij}; j = 1, 2, ..., n\} \), for each \( i = 1, 2, ..., n \). This definition to \( \lambda \) also produces good results in matrices with great dispersion between its coefficients.

The parameter \( \tau \) depends on the necessary number of iterations for the convergence of the Wang's neural network. When the presented correction "Winner Takes All" technique isn't used, the necessary number of iterations for the convergence of the Wang's neural network varies between 1,000 and 15,000 iterations. In this case, \( \tau \) is a constant, such that:

\[
1,000 \leq \tau \leq 15,000. \tag{18}
\]

When the "Winner Takes All" correction is used, the necessary number of iterations varies between 5 and 300. Hence, the value of \( \tau \) is such that:

\[
5 \leq \tau \leq 300. \tag{19}
\]

In this work, two other forms of \( \tau \) parameter adjustment had been used, besides considering it constant, in the intervals showed in expressions (18) and (19). In one of the techniques, \( \tau \) is given by:
where \( \mu \) are the coefficients average of \( i \)th row of matrix \( c \), \( \delta \) is the standard deviation of \( i \)th row of matrix \( c \), and \( \mu \) is the average between the values of all the coefficients of \( c \).

The second proposal of adjustment for \( \tau \) uses the third term of definition of neural network of Wang (8). When \( c_{ij} = c_{\text{max}} \), the term \(-\lambda c_{ij} \exp(-t / \tau_i) = k_i \) must satisfied \( g(k_i) \cong 0 \), so \( x_{ij} \) has minor value, minimizing the final cost of the Assignment Problem. Isolating \( \tau_i \) and considering \( c_{ij} = c_{\text{max}} \) and \( \lambda_i = 1 / \delta_i \), where \( i = 1, 2..., n \), \( \tau \) is got, as follows:

\[
\tau_i = \frac{-t}{\ln \left( \frac{-k_i}{\lambda_i c_{\text{max}}} \right)},
\]

The parameters’ application results given by (14)-(21) are presented on next section.

6. Results to assignment problem

In this work, 100 matrices (with dimensions varying of \( 3 \times 3 \) until \( 20 \times 20 \)) had been used to test the techniques of adjustments to parameters presented in previous section, beyond the proposed “Winner Takes All” correction applied to the Wang’s recurrent neural network. These matrices had been generated randomly, with some cases of multiple optimal solutions and very closer optimal solutions.

The results to 47 tested matrices with only one optimal global appear in Table 1, and results to 53 matrices with multiple optimal solutions and/or very closer optimal solutions appear in Table 2. Table 3 shown results to all matrices tested to Assignment Problem.

To adjust \( \lambda_i \), the following expressions had been used on Tables 1, 2 and 3: (14) in the first and last column; (15) in the second column; (17) in the third column; and (16) in fourth and fifth columns. To calculate \( \tau \), the following expressions they had been used: (19) in the three firsts columns; (20) in the fourth column; (21) in the fifth column; and (18) in the last column. The results of the Wang’s neural network application, without the use of the proposed correction in this work, are meet in the last column of Tables 1, 2 and 3. In the last row of the Tables 1, 2 and 3 the numbers of iterations of each technique is given by the average between the numbers of iterations found for all tested matrices.

<table>
<thead>
<tr>
<th>parameter ( \lambda )</th>
<th>( \lambda = \eta / c_{\text{max}} )</th>
<th>( \lambda = \eta / \delta )</th>
<th>( \lambda_i = \eta / c_{\text{max}} )</th>
<th>( \lambda_i = \eta / \delta_i )</th>
<th>( \lambda = \eta / c_{\text{max}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter ( \tau )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( \tau_i = \frac{\mu_i \delta_i}{\mu} )</td>
<td>( \tau_i = \frac{-t}{\ln \left( \frac{-k_i}{\lambda_i c_{\text{max}}} \right)} )</td>
</tr>
<tr>
<td>( 1,000 \leq \tau \leq 15,000 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>global optimality</td>
<td>40</td>
<td>45</td>
<td>40</td>
<td>40</td>
<td>46</td>
</tr>
<tr>
<td>local optimality</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>infeasibility</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>global optim.(%)</td>
<td>85</td>
<td>96</td>
<td>85</td>
<td>85</td>
<td>98</td>
</tr>
<tr>
<td>average error (%)</td>
<td>2.35</td>
<td>0.98</td>
<td>0.74</td>
<td>5.10</td>
<td>0.02</td>
</tr>
<tr>
<td>iterations (average)</td>
<td>37</td>
<td>46</td>
<td>41</td>
<td>72</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 1. Results for 47 matrices with only one optimal solution
The results had been considered satisfactory, and the adjustments of the parameters that result in better solutions for the “Winner Takes All” correction are those that use the standard deviation and the average between the elements of matrix of costs, and the use of parameters in vector form revealed to be more efficient for these matrices. The results shown in Tables 1, 2 and 3 reveal that the dispersion techniques between the coefficients of matrix \( c \) are more efficient for the use of the correction “Winner Takes All” in matrices with multiple optimal solutions.

<table>
<thead>
<tr>
<th>Parameter ( \lambda )</th>
<th>( \lambda = \frac{\eta}{C_{\text{max}}} )</th>
<th>( \lambda = \frac{\eta}{\delta} )</th>
<th>( \lambda_t = \frac{\eta}{c_{\text{max}}} )</th>
<th>( \lambda = \frac{\eta}{\delta} )</th>
<th>( \lambda_t = \frac{\eta}{\delta_t} )</th>
<th>( \lambda = \frac{\eta}{C_{\text{max}}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter ( \tau )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( 5 \leq \tau \leq 300 )</td>
<td>( \tau = \frac{\mu_t \delta_t}{\mu} )</td>
<td>( \tau_t = \frac{-t}{\ln\left(\frac{c_{\text{max}}}{\lambda \delta_t}\right)} )</td>
<td>( 1,000 \leq \tau \leq 15,000 )</td>
</tr>
<tr>
<td>Global Optimality (%)</td>
<td>73</td>
<td>88</td>
<td>72</td>
<td>79</td>
<td>92</td>
<td>47</td>
</tr>
<tr>
<td>Local Optimality (%)</td>
<td>27</td>
<td>12</td>
<td>28</td>
<td>21</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Infeasibility (%)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>53</td>
</tr>
<tr>
<td>Average Error (%)</td>
<td>3.17</td>
<td>1.19</td>
<td>2.57</td>
<td>2.11</td>
<td>0.71</td>
<td>-</td>
</tr>
<tr>
<td>Iterations (average)</td>
<td>38</td>
<td>44</td>
<td>41</td>
<td>74</td>
<td>49</td>
<td>4,970</td>
</tr>
</tbody>
</table>

Table 2. Results for 53 matrices with multiple optimal solutions

The pure Wang’s neural network has slower convergence when the adjustments described by (15)-(17) and (19)-(21) are applied for the parameters \( \lambda \) and \( \tau \), respectively. Better results are found with combination of parameters (16) and (21), as shown in Tables 1, 2 and 3. This combination is used to solve the Traveling Salesman Problem.

These results shows that the “Winner Takes All” principle, applied to the Wang’s neural network, produces good results to Assignment Problem, mainly in matrices with multiple optimal solutions. The parameters to Wang’s neural network presented in section 5 show the efficiency of this technique for great scale problems, because the average number of iterations necessary to find feasible solutions for the Assignment Problem was considerably reduced, compared to the pure Wang’s neural network.

The application of “Winner Take All” principle to Wang’s recurrent neural network to solve the Traveling Salesman Problem is presented on next sections.
7. The traveling salesman problem

The formulation of Traveling Salesman Problem is the same of Assignment Problem, with the additional constraint of Hamiltonian circuit, i.e., the feasible route must form a cycle which visits each city exactly once, and returns to the starting city:

\[
\text{Minimize } C = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{Subject to } \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n \\
\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n \\
x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n
\]

where the vector \( \vec{x} \) has the whole sequence of the route that was found, i.e., the solution for the Traveling Salesman Problem.

The Traveling Salesman Problem is a classical problem of combinatorial optimization in the Operations Research area. The purpose is to find a minimum total cost Hamiltonian cycle (Ahuja et al., 1993). There are several practical uses for this problem, such as Vehicle Routing (Laporte, 1992) and Drilling Problems (Onwubolu & Clerc, 2004).

This problem has been used during the last years as a basis for comparison in order to improve several optimization techniques, such as Genetic Algorithms (Affenzeller & Wagner, 2003), Simulated Annealing (Budinich, 1996), Tabu Search (Liu et al., 2003), Local Search (Bianchi et al., 2005), Ant Colony (Chu et al., 2004) and Neural Networks (Leung et al., 2004; Siqueira et al., 2007).

The main types of Neural Network used to solve the Traveling Salesman Problem are: Hopfield’s Recurrent Networks (Wang et al., 2002) and Kohonen’s Self-Organizing Maps (Leung et al., 2003). In a Hopfield’s Network, the main idea is to automatically find a solution for the Traveling Salesman Problem by means of an equilibrium state of the equation system defined for the Traveling Salesman Problem. By using Kohonen’s Maps for the Traveling Salesman Problem, the final route is determined through the cities corresponding to those neurons that have weights that are closest to the pair of coordinates ascribed to each city in the problem.

Wang’s recurrent neural network with the “Winner Takes All” principle can be applied to solve the Traveling Salesman Problem on this way: solving this problem as if it were an Assignment Problem by means of the Wang’s neural network, and, furthermore, using the “Winner Takes All” principle on the solutions found with the Wang’s neural network, with the constraint that the solutions found must form a feasible route for the Traveling Salesman.
A Recurrent Neural Network to Traveling Salesman Problem

Problem. The parameters used for the Wang’s neural network are those that show the best solutions for the Assignment Problem, as shown on Tables 1, 2 and 3 of previous section. The solutions found with the heuristic technique proposed in this work are compared with the solutions from the Self-Organizing Maps (SOM) and the Simulated Annealing (SA) for the symmetrical TSP, and with other heuristics for the asymmetrical TSP. The 2-opt Local Search technique (Bianchi et al., 2005) is used to improve the solutions found with the technique proposed in this work. The data used for the comparisons are from the TSPLIB database (Reinelt, 1991).

8. Wang’s neural network and “Winner Takes All” principle to traveling salesman problem

The algorithm presented on section 4 to Assignment Problem can be easily modified to solve the Traveling Salesman Problem:

Step 1: Determine a maximum number of routes $r_{\text{max}}$. Find a solution $x$ to Assignment Problem using the Wang’s neural netowrk. If $Wx(t) - \theta \leq \phi$, then go to Step 2. Otherwise, find another solution $x$.

Step 2: Given the decision matrix, consider matrix $\bar{x}$, where $\bar{x} = x, m = 1$ and go to Step 3.

Step 3: Choose a row $k$ in decision matrix $\bar{x}$. Do $p = k$, $\bar{x}$ $(m) = k$ and go to Step 4.

Step 4: Find the biggest element of row $k$, $\bar{x}_{kl}$. This element’s value is given by the half of the sum of all elements of row $k$ and of column $l$ of matrix $x$, i.e.,

$$\bar{x}_{kl} = \frac{1}{2} \left( \sum_{i=1}^{n} x_{il} + \sum_{j=1}^{n} x_{kj} \right). \tag{27}$$

The other elements of row $k$ and column $l$ become null. So that sub-routes are not formed, the other elements of column $k$ must also be null. Do $\bar{x}$ $(m + 1) = l$; to continue the Traveling Salesman Problem route, make $k = l$ and go to Step 5.

Step 5: If $m < n$, then make $m = m + 1$ and go to Step 4. Otherwise, do

$$\bar{x} (n + 1) = p$$

determine the route’s cost, $C$, and go to Step 6.

Step 6: If $C < C_{\text{min}}$ then do $C_{\text{min}} = C$ and $x = \bar{x}$. Make $r = r + 1$. If $r < r_{\text{max}}$ then run the Wang’s neural network again and go to Step 2, otherwise Stop.

8.1 Illustrative examples applied to problems of TSPLIB

Consider the symmetrical Traveling Salesman Problem with 14-city instances burma14, due Zaw & Win (Reinelt, 1991), as shown in Fig. 1. After 17 iterations, the Wang’s neural network presents the following solution for the Assignment Problem:
In this decision matrix, a city is chosen to start the route, for instance, city 8, this is, \( p = 8 \). In row \( p \) of the decision matrix the biggest element is chosen, thus defining the Traveling Salesman’s destiny when he leaves city \( p \). The biggest element of row \( p \) is in column 1, therefore, \( k = p = 8 \) and \( l = 1 \). After the decision matrix \( \bar{x} \) is updated by means of equation (27), the route goes on with \( k = 1 \):

The biggest element of row 1 in matrix \( \bar{x} \) is in column 2, therefore, \( l = 2 \). This procedure is executed until all rows are updated, thus defining the route: \( \bar{x} = (8, 1, 2, 10, 9, 11, 13, 7, 6, 5, 4, 3, 14, 12, 8) \), as shown in Fig. 1a, with a cost of 34.03, which represents an average error of 10.19%.
This solution is presented to the Wang’s neural network, by making $x = \bar{x}$. After more 16 iterations the neural network the following decision matrix is presented:

$$\bar{x} = \begin{pmatrix}
0 & 0.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Through the “Winner Takes All” principle, an approximation for the optimal solution of this problem is found with the route: $\bar{x} = (2, 1, 10, 9, 11, 8, 13, 7, 12, 6, 5, 4, 3, 14, 2)$, with a cost of 30.88 (Fig. 2b).
Consider the symmetrical Traveling Salesman Problem with 42-city instance by Dantzig (Reinelt, 1991), as shown in Fig. 3 and 4. This problem contains coordinates of cities in the United States, and after 25 epochs the condition $Wx(t) - \theta \leq \phi$ is satisfied with $\phi = 0.01$ and the Wang’s neural network presents the first solution $\tilde{x}_1$ for the Traveling Salesman Problem, as shown in Fig. 3a.

Fig. 2. (a) Feasible solution found to burma14 through the proposed method, with an average error of 10.19%. (b) Optimal solution found through the proposed method

The solution $\tilde{x}_1$ is presented to Wang’s neural network, and after 20 iterations an improved solution is reached, with the average error decreasing from 19.56% to 0.83% as shown in Fig. 3a and 3b.

An improvement to heuristic Wang’s neural network is the application of local search 2-opt heuristic on Step 5 of the algorithm shown in this section. This application is made after the expression (27), to the Wang’s neural network solution in the algorithm, just as an improvement. The results of Wang’s neural network with 2-opt on problem dantzig42 is shown in Fig. 4, where after 72 epochs an optimal solution is found.
Fig. 3. Solutions found to dantzig42 data without 2-opt improvement. (a) First feasible tour found through the proposed heuristic, with an average error of 19.56%. (b) Tour with 0.83% of average error, after 29 iterations. Others examples of results found to symmetrical Traveling Salesman problems are (Reinelt, 1991): the 58-city instance of Brazil, due Ferreira, shown in Fig. 5; the 532-city instances of United States due Padberg and Rinaldi, shown in Fig. 6; and the drilling problem u724 due Reinelt, shown in Fig. 7.

Fig. 4. Solutions found to dantzig42 data with 2-opt improvement. (a) First feasible solution found, in 26 epochs and average error of 8.77%. (b) 36 epochs and error 1.5%. (c) 45 epochs and error 0.57%. (d) 72 epochs and optimal solution found.
Fig. 5. Solutions found to brazil58 data. (a) Feasible solution found without local search improvement with 81 epochs and average error of 2.9%. (b) Optimal solution found with 2-opt improvement with 88 epochs.

Fig. 6. Solutions found to att532 data. (a) Feasible solution found without local search improvement with 411 epochs and average error of 14.58%. (b) Feasible solution found with local search improvement with 427 epochs and average error of 1.27%.

Fig. 7. Solutions found to u724 data. (a) Feasible solution found without local search improvement with 469 epochs and average error of 16.85%. (b) Feasible solution found with local search improvement with 516 epochs and average error of 6.28%.
The next Section shows the results of applying this technique to some of the TSPLIB’s problems for symmetrical and asymmetrical Traveling Salesman problems.

9. Comparisons of technique proposed with others heuristics to some TSPLIB’s problems

The results found with the technique proposed to problems of TSPLIB with symmetrical cases are compared with Self Organizing Maps and Simulated Annealing results. In asymmetrical problems of TSPLIB, the technique proposed are compared with heurist of insertion of arcs. In both cases the local search technique was applied to results found with Wang’s Recurrent Neural Network with “Winner Takes All”.

For symmetrical problems, the following methods were used to compare with the technique presented in this work:

- the method that involves statistical methods between neurons’ weights of Self Organizing Maps (Aras et al., 1999) and has a global version (KniesG: Kohonen Network Incorporating Explicit Statistics Global), where all cities are used in the neuron dispersion process, and a local version (KniesL), where only some represented cities are used in the neuron dispersion step;
- the Simulated Annealing technique (Budinich, 1996), using the 2-opt improvement technique;
- Budinich’s Self Organizing Map, which consists of a traditional Self Organizing Map applied to the Traveling Salesman Problem, presented in Budinich, 1996;
- the expanded Self Organizing Map (ESOM), which, in each iteration, places the neurons close to their corresponding input data (cities) and, at the same time, places them at the convex contour determined by the cities (Leung et al., 2004);
- the efficient and integrated Self Organizing Map (eISOM), where the ESOM procedures are used and the winning neuron is placed at the mean point among its closest neighboring neurons(Jin et al., 2003);
- the efficient Self Organizing Map technique (SETSP), which defines the updating forms for parameters that use the number of cities of problem (Vieira et al., 2003);
- and Kohonen’s cooperative adaptive network (CAN) uses the idea of cooperation between the neurons’ close neighbors and uses a number of neurons that is larger than the number of cities in the problem (Cochrane & Beasley, 2003).

The computational complexity of the proposed heuristic is $O(n^2 + n)$ (Wang, 1997), considered competitive when compared to the complexity of mentioned Self Organizing Map, which have complexity $O(n^2)$ (Leung et al., 2004). The CAN technique has a computational complexity of $O(n^2 \log(n))$ (Cochrane & Beasley, 2003), while the Simulated Annealing technique has a complexity of $O(n^4 \log(n))$ (Liu et al., 2003).

The results for the proposed heuristic in this paper, together with the 2-opt improvement, presented an average error range from 0 to 3.31%, as shown in the 2-opt column of Table 4. The methods that use improvement techniques to their solutions are Simulated Annealing, CAN and Wang’s neural network with “Winner Takes All”. The technique proposed in this paper, with 2-opt, present better results that Simulated Annealing and CAN methods in almost every problem, with the only exception in the lin105 problem. Without the improvement 2-opt, the results of problems eil76, eil51, eil101 and...
rat195 are better than the results of the other neural networks that do not use improvement techniques in its solutions.

In Table 4 are shown the average errors of the techniques mentioned above. The "pure" technique proposed in this work to Traveling Salesman Problem, the proposed technique with the 2-opt improvement algorithm, as well as the best (max) and worst (min) results of each problem considered are also shown.

<table>
<thead>
<tr>
<th>TSP’s name</th>
<th>n</th>
<th>optimal solution</th>
<th>average error (%) for 8 algorithms presented on TSPLIB</th>
<th>WRNN with WTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>eil51</td>
<td>51</td>
<td>430</td>
<td>2.86 2.86 2.33 3.10 2.10 2.56 2.22 0.94 1.16 1.16 0</td>
<td></td>
</tr>
<tr>
<td>st70</td>
<td>70</td>
<td>678.6</td>
<td>2.33 1.51 2.14 1.70 2.09 NC 1.60 1.33 4.04 2.71 0</td>
<td></td>
</tr>
<tr>
<td>eil76</td>
<td>76</td>
<td>545.4</td>
<td>5.48 4.98 5.54 5.32 3.89 NC 4.23 2.04 2.49 1.03 0</td>
<td></td>
</tr>
<tr>
<td>gr96</td>
<td>96</td>
<td>514</td>
<td>4.12 2.09 1.03 NC NC NC 6.61 4.28 0</td>
<td></td>
</tr>
<tr>
<td>rd100</td>
<td>100</td>
<td>7,910</td>
<td>2.62 2.09 3.26 3.16 1.96 NC 2.60 1.23 7.17 6.83 0.08</td>
<td></td>
</tr>
<tr>
<td>eil101</td>
<td>101</td>
<td>629</td>
<td>5.63 4.66 5.74 5.24 3.43 3.59 NC 1.11 7.95 3.02 0.48</td>
<td></td>
</tr>
<tr>
<td>lin105</td>
<td>105</td>
<td>14,383</td>
<td>1.29 1.98 1.87 1.71 0.25 NC 1.30 0 5.94 4.33 0.20</td>
<td></td>
</tr>
<tr>
<td>pr107</td>
<td>107</td>
<td>44,303</td>
<td>0.42 0.73 1.54 1.32 1.48 NC 0.41 0.17 3.14 3.14 0</td>
<td></td>
</tr>
<tr>
<td>pr124</td>
<td>124</td>
<td>59,030</td>
<td>0.49 0.08 1.26 1.62 0.67 NC NC 2.36 2.63 0.33 0</td>
<td></td>
</tr>
<tr>
<td>bier127</td>
<td>127</td>
<td>118,282</td>
<td>3.08 2.76 3.52 3.61 1.70 NC 1.85 0.69 5.08 4.22 0.37</td>
<td></td>
</tr>
<tr>
<td>pr136</td>
<td>136</td>
<td>96,772</td>
<td>5.15 4.53 4.90 5.20 4.31 NC 4.40 3.94 6.86 5.99 1.21</td>
<td></td>
</tr>
<tr>
<td>pr152</td>
<td>152</td>
<td>73,682</td>
<td>1.29 0.97 2.64 2.04 0.89 NC 1.17 0.74 3.27 3.23 0</td>
<td></td>
</tr>
<tr>
<td>rat195</td>
<td>195</td>
<td>2,323</td>
<td>11.92 12.24 13.29 11.48 7.13 NC 11.19 5.27 8.82 5.55 3.31</td>
<td></td>
</tr>
<tr>
<td>kroa200</td>
<td>200</td>
<td>29,368</td>
<td>6.57 5.72 5.61 6.13 2.91 1.64 3.12 0.92 12.25 8.95 0.62</td>
<td></td>
</tr>
<tr>
<td>lin318</td>
<td>318</td>
<td>42,029</td>
<td>NC NC 7.56 8.19 4.11 2.05 NC 2.65 8.65 8.35 1.90</td>
<td></td>
</tr>
<tr>
<td>pcb442</td>
<td>442</td>
<td>50,784</td>
<td>10.45 11.07 9.15 8.43 7.43 6.11 10.16 5.89 13.18 9.16 2.87</td>
<td></td>
</tr>
<tr>
<td>att532</td>
<td>532</td>
<td>27,686</td>
<td>6.8 6.74 5.38 5.67 4.95 3.35 NC 3.32 15.43 14.58 1.28</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Results of the experiments for the symmetrical problems of TSP, with techniques presented on TSPLIB: KniesG, KniesL, SA, Budinich’s SOM, ESOM, EISOM, SETSP, CAN and a technique presented on this paper: WRNN with WTA. The solutions presented in bold characters show the best results for each problem, disregarding the results with the 2-opt technique. (NC = not compared)

For the asymmetrical problems, the techniques used to compare with the technique proposed in this work were (Glover et al., 2001):

- the Karp-Steele path methods (KSP) and general Karp-Steele (GKS), which begin with one cycle and by removing arcs and placing new arcs, transform the initial cycle into a
Hamiltonian one. The difference between these two techniques is that the GKS uses all of the cycle's vertices for the changes in the cycle's arcs;

- the path recursive contraction (PRC) that consists in forming an initial cycle and transforming it into a Hamiltonian cycle by removing arcs from every sub-cycle;
- the contraction or path heuristic (COP), which is a combination of the GKS and PRC techniques;
- the “greedy” heuristic (GR) that chooses the smallest arc in the graph, contracts this arc creating a new graph, and keeps this procedure up to the last arc, thus creating a route;
- and the random insertion heuristic (RI) that initially chooses 2 vertices, inserts one vertex that had not been chosen, thus creating a cycle, and repeats this procedure until it creates a route including all vertices.

Table 5. Results of the experiments for the asymmetrical problems of TSP with techniques presented on TSPLIB: GR, RI, KSP, GKS, PRC, COP and a technique presented on this paper: WRNN with WTA. The solutions presented in bold characters show the best results for each problem, disregarding the results with the 2-opt technique.

<table>
<thead>
<tr>
<th>TSP's name</th>
<th>$n$</th>
<th>optimal solution</th>
<th>average error (%) for 6 algorithms WRNN with WTA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>GR</td>
</tr>
<tr>
<td>br17</td>
<td>17</td>
<td>39</td>
<td>102,56</td>
</tr>
<tr>
<td>ftv33</td>
<td>33</td>
<td>1,286</td>
<td>31,34</td>
</tr>
<tr>
<td>ftv35</td>
<td>35</td>
<td>1,473</td>
<td>24,37</td>
</tr>
<tr>
<td>ftv38</td>
<td>38</td>
<td>1,530</td>
<td>14,84</td>
</tr>
<tr>
<td>pr43</td>
<td>43</td>
<td>5,620</td>
<td>3,59</td>
</tr>
<tr>
<td>ftv44</td>
<td>44</td>
<td>1,613</td>
<td>18,78</td>
</tr>
<tr>
<td>ftv47</td>
<td>47</td>
<td>1,776</td>
<td>11,88</td>
</tr>
<tr>
<td>ry48p</td>
<td>48</td>
<td>14,422</td>
<td>32,55</td>
</tr>
<tr>
<td>ft53</td>
<td>53</td>
<td>6,905</td>
<td>80,84</td>
</tr>
<tr>
<td>ftv55</td>
<td>55</td>
<td>1,608</td>
<td>25,93</td>
</tr>
<tr>
<td>ftv64</td>
<td>64</td>
<td>1,839</td>
<td>25,77</td>
</tr>
<tr>
<td>ft70</td>
<td>70</td>
<td>38,673</td>
<td>38,673</td>
</tr>
<tr>
<td>ftv70</td>
<td>70</td>
<td>1,950</td>
<td>31,85</td>
</tr>
<tr>
<td>kro124p</td>
<td>100</td>
<td>36,230</td>
<td>21,01</td>
</tr>
<tr>
<td>ftv170</td>
<td>170</td>
<td>2,755</td>
<td>32,05</td>
</tr>
<tr>
<td>rbg323</td>
<td>323</td>
<td>1,326</td>
<td>8,52</td>
</tr>
<tr>
<td>rbg358</td>
<td>358</td>
<td>1,163</td>
<td>7,74</td>
</tr>
<tr>
<td>rbg403</td>
<td>403</td>
<td>2,465</td>
<td>0,85</td>
</tr>
<tr>
<td>rbg443</td>
<td>443</td>
<td>2,720</td>
<td>0,92</td>
</tr>
</tbody>
</table>

Table 5 shows the average errors of the techniques described, as well as those of the "pure" technique presented in this work and of the proposed technique with the 2-opt technique.
The results of the "pure" technique proposed in this work are better or equivalent to those of the other heuristics mentioned above, for problems br17, ftv33, ftv44, ft53, ft70 and kro124p, as shown in Table 5. By using the 2-opt technique on the proposed technique, the best results were found for problems br17, ftv33, pr43, ry48p, ftv44, ft53, ft70 and kro124p, with average errors ranging from 0 to 16.14%.

10. Conclusions

This work presented the Wang’s recurrent neural network with the “Winner Takes All” principle to solve the Assignment Problem and Traveling Salesman Problem. The application of parameters with measures of matrices dispersion showed better results to both problems.

The results of matrices to Assignment Problem had shown that the principle “Winner Takes all” solves problems in matrices with multiple optimal solutions, besides speed the convergence of the Wang’s neural network using only 1% of necessary iterations of neural network pure.

Using the best combination of parameters, the average errors are only 0.71% to 100 tested matrices to Assignment Problem. Using these parameters solutions of Traveling Salesman Problem can be found.

By means of the Wang’s neural network, a solution for the Assignment Problem is found and the “Winner Takes All” principle is applied to this solution, transforming it into a feasible route for the Traveling Salesman Problem. These technique’s solutions were considerably improved when the 2-opt technique was applied on the solutions presented by the proposed technique in this work.

The data used for testing were obtained at the TSPLIB and the comparisons that were made with other heuristics showed that the technique proposed in this work achieves better results in several of the problems tested, with average errors below 16.14% to these problems.

A great advantage of implementing the technique presented in this work is the possibility of using the same technique to solve both symmetrical and asymmetrical Traveling Salesman Problem as well.

11. References


www.intechopen.com


The idea behind TSP was conceived by Austrian mathematician Karl Menger in mid 1930s who invited the research community to consider a problem from the everyday life from a mathematical point of view. A traveling salesman has to visit exactly once each one of a list of m cities and then return to the home city. He knows the cost of traveling from any city i to any other city j. Thus, which is the tour of least possible cost the salesman can take? In this book the problem of finding algorithmic technique leading to good/optimal solutions for TSP (or for some other strictly related problems) is considered. TSP is a very attractive problem for the research community because it arises as a natural subproblem in many applications concerning the every day life. Indeed, each application, in which an optimal ordering of a number of items has to be chosen in a way that the total cost of a solution is determined by adding up the costs arising from two successively items, can be modelled as a TSP instance. Thus, studying TSP can never be considered as an abstract research with no real importance.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

© 2008 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike-3.0 License, which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited and derivative works building on this content are distributed under the same license.