An Efficient Multivalued Hopfield Network for the Traveling Salesman Problem

E. MÉRIDA-CASERMEIRO1, G. GALÁN-MARÍN3 and J. MUÑOZ-PÉREZ2
1Department of Applied Mathematics, 2Department of Computer Science, University of Málaga, Campus de Teatinos s/n, 29071-Málaga, Spain; e-mail: munozp@lcc.uma.es.
3Department of Electronics and Electromechanical Engineering, University of Extremadura, Escuela de Ingenierías Industriales, Avda. de Elvas s/n, 06071-Badajoz, Spain; e-mail: gloriagm@unex.es

Abstract. In this Letter we show that discrete multivalued Hopfield-type neural networks enable a relatively easy formulation of the Traveling Salesman Problem compared to the traditional Hopfield model. Thus, with the multivalued representation the network can be easily confined to feasible solutions, avoiding the need to tune any parameter. An investigation into the performance of the network has led us to define updating rules based on simple effective heuristic algorithms, a technique that can not be usually incorporated into standard Hopfield models. Simulation results for Euclidean Traveling Salesman Problems taken from the data library TSPLIB [11] indicate that this multivalued neural approach is superior to the best neural network currently reported for this problem.

Key words: combinatorial optimization, discrete Hopfield model, multivalued neurons, neural networks, traveling salesman problem

1. Introduction

The traveling salesman problem (TSP) is a combinatorial optimization problem that results in many technical applications in such areas as routing robots through automatic warehouses and drilling holes in printed circuit boards [12]. Computational time required to find an exactly optimum solution grows faster than any finite power of some appropriate measure of the problem size as long as \( P \neq NP \) [7–9]. Because of the complexity of this problem, efficient approximate algorithms for finding a near-optimum solution within a reasonable computational time have been searched for.

In addition to the classical heuristic algorithms of Operations Research [10], there have also been several approaches based on artificial neural networks which solve the TSP. The first neural network for combinatorial optimization problems was the analog Hopfield model introduced by Hopfield and Tank [1] in 1985. The Hopfield network has demonstrated that a distributed system of simple processing elements can collectively solve optimization problems. The goal of neural network approaches to combinatorial optimization is to formulate the desired objective function being optimized, such that it can be viewed as a ‘natural’ energy minimization problem.
Although the Hopfield networks implement a gradient descent method they should not be viewed as naive gradient descent machines, but as an ensemble of interconnected processing units with simple computational requirements that can implement complex computation (inspired in many natural phenomena). Thus, the true advantage of using Hopfield-type neural networks to solve difficult optimization problems relates to speed considerations. Due to their inherently parallel structure and simple computational requirements, neural network techniques are especially suitable for direct hardware implementation, using analog or digital integrated circuits [4], or parallel simulations [5]. Moreover, the Hopfield neural networks have very natural implementations in optics [6]. Thus, they are considered to hold much potential for rapid execution speed through their hardware implementation.

Basically, the Hopfield model has two versions: analog [3] and discrete [2]. For the analog version, neurons have continuous values within (0,1), and for the discrete version the neurons have only two possible values, 0 or 1. Of the two versions, the analogue is superior to the discrete in terms of the local minima problem, because of its smoother energy surface. Hence, the most famous neural approach to combinatorial optimization problems is the analog Hopfield network. However, due to its continuous change in the state variable, its convergence is slow and this network does not always converge to a valid state.

Some of the problems on the use of the analog Hopfield model have been highlighted by Wilson and Pawley [18], and one of them is the need to tune the parameters in the energy function. These and some other deficiencies of the model have led researchers to modify the original model. Over the past fifteen years, a significant amount of work has been reported on improving the analog Hopfield network [20–23, 30, 33]. Subsequent efforts have resulted in methods that can ensure that invalid solutions are never found [16]. However, it is not still as clear how the networks’ solutions compare in terms of quality to those obtained using other optimization techniques. For the TSP, it can be stated as a conclusion that both the original analog Hopfield model as well as its variants are slow and provide good solutions for TSPs which consist of 200 cities or fewer.

The discrete Hopfield model is not usually found in the literature as a solver of combinatorial optimization problems, but as a content-addressable memory. However, recent models based on the binary sequential Hopfield model [35, 36] and on the binary competitive Hopfield model [34, 38] have been shown to provide powerful approaches for combinatorial optimization, by solving very large scale problems with a high speed of convergence. The problem is that the quality of the solutions found using these discrete Hopfield models in the TSP is still unlikely to be comparable to those obtained using traditional techniques. As it has been recently pointed out [17], the poor performance of the Hopfield network on the TSP when compared to traditional operations research techniques may be explained by the fact that Hopfield and Tank used an integer quadratic formulation of the TSP.
However, the operations research community generally uses an integer linear programming formulation.

Other connectionist approaches different from those based on the Hopfield network have also been used to solve the TSP. Kohonen’s self organizing map algorithm (SOM) [15, 31] and elastic net algorithms [32] have successfully solved large-scale TSP problems. The most accurate of all reported neural solutions for the TSP available in the literature has been recently obtained by applying the Kohonen network incorporating explicit statistic (KNIES) [13]. However, in this paper we propose a discrete multivalued Hopfield-type network with dynamics derived from heuristic methods that provides comparable or better solutions than KNIES in very short processing times. Moreover, our network does not demand fine-tuning of parameters, as most SOM based methods do.

Multivalued neural networks were firstly introduced by Erdem and Ozturk [19]. They proposed a multivalued, recurrent, nonlinear associative memory structure (MAREN) inspired from Hopfield’s binary associative memory. This model can also be used in solving optimization problems. Although a very interesting formulation of the TSP is applied by [19], results reported in up to the 80-city TSP problem are not very accurate. In this paper we propose a multivalued neural network with a different formulation of the TSP. Moreover, since it is a very easy task generating only feasible solutions with the multivalued representation, we consider the search space as the set of valid tours. Thus our energy function is the tour distance and we avoid the need to tune any parameter. An investigation into the performance of the network brings us to define updating rules completely different to the ones of [19]. Instead of inspiring from Hopfield’s binary memory, our updating rules are based on topological information of the tour defined by the states of the neurons. In this way we derive the dynamics of the network from the well-known 2-opt and 3-opt moves [28].

This Letter is organized as follows. Section 2 briefly describes the formulations of the TSP by the traditional Hopfield model, by the multivalued MAREN model and by the proposed multivalued model. Section 3 describes 2-opt and 3-opt, the simplest and most famous of the classical local optimization algorithms, and discusses some of their implementations. Section 4 describes the proposed multivalued network and its dynamics based on \( k \)-opt moves. Finally, Section 5 evaluates the performance of the network in solving the TSP compared with the best reported neural network methods. Conclusions are given in Section 6.

2. Neural Network Formulations of the TSP

2.1. MAPPING THE TSP ON TO THE HOPFIELD NETWORK

The TSP is a problem that is easy to describe but hard to solve. Given a \( N \times N \) symmetric matrix \( D = (d_{ij}) \) of distances between a set of \( N \) cities, \((i, j = 1, 2, \ldots, N)\), find a minimum-length tour that visits each city exactly once.
In the Hopfeld and Tank approach [1], a solution for the TSP is described in terms of an \( N \times N \) matrix \( V = (v_{ij}) \) with \((0,1)\)-binary elements, such that if \( v_{ij} = 1 \) then city \( i \) is in \( j \)th place in the tour found by the network. Preference for the matrix \( V \) as a solution of the TSP can be measured by the following energy function

\[
E = A \frac{1}{2} \sum_{i=1}^{N} \left( \sum_{k=1}^{N} v_{ik} - 1 \right)^2 + A \frac{1}{2} \sum_{k=1}^{N} \left( \sum_{i=1}^{N} v_{ik} - 1 \right)^2 +
\]

\[
+ B \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} d_{ij} v_{ik} (v_{j,k+1} + v_{j,k-1})
\]

where the subscripts in the last term are cyclic, such that \( v_{j,N+1} = v_{j,1} \) and \( v_{j,1} = v_{j,N} \). The first two terms represent the constraint terms to satisfy the feasibility condition of TSP, and the third term represents a total path length of a complete tour of feasible TSP solutions.

Another interesting formulation of the TSP for the Hopfield model based on an ‘adjacency representation’ can be found in [33]. The problem is that it generates an optimum combination of optimum subtours, which is not always an optimum Hamiltonian tour.

2.2. MAPPING THE TSP ON TO THE MULTIVALUED MAREN NETWORK

The architecture of the MAREN network [19] is identical to that of the discrete Hopfield model. It consists of a single layer of \( N \) discrete multivalued neurons or processing elements where every neuron is connected to all others. The main difference is that each neuron can take one of \( M \) discrete levels, instead of the two only possible values 0 or 1. As Erdem and Ozturk pointed out [19], multivalued representation is a much relevant and direct approximation to real world data than binary or bipolar representations.

Because of the multivalued representation, mapping the TSP on to the network is relatively easy compared to the Hopfield model. In an \( N \)-city TSP network there are \( N \) neurons and the discrete levels are \( M = \{1, 2, \ldots, N\} \). Every neuron denotes a city and the state of this neuron represents the visiting order of that city, e.g., if \( v_B = 3 \), city \( B \) will be the third city to be visited. Erdem and Ozturk considered for this problem the energy function

\[
E = A \sum [\text{sgn}(v_i - v_j)]^2 - B \sum d_{ij} [1 - [\text{sgn}(\text{abs}(v_i - v_j) - 1)]] -
\]

\[
- B \sum d_{ij} [1 + [\text{sgn}(\text{abs}(v_i - v_j) - N + 1)]]
\]

where the signum function is defined as \( \text{sgn}(x) = 1, -1, 0 \) if \( x > 0, x < 0 \) and \( x = 0 \) respectively. The first term \( E_1 \) ‘punishes’ the states which have some equal valued neurons, since in a valid tour all neurons will have distinct values. The second term \( E_2 \) guarantees that valid tours have higher energies than invalid ones. The third term
$E_3$ is added to include the distance between the first and the last cities, where $E_2 + E_3$ is exactly (minus) the tour length times $B$. An updating rule is also defined in order to monotonically increase the energy function. It is observed that $E$ is a rather difficult energy function for solving the TSP, clearly inspired from a binary associative memory. Although this is a very interesting formulation, results are not very accurate. We propose now a different formulation of the TSP for a multivalued network with a very simple energy function.

### 2.3. Mapping the TSP on to the Proposed Multivalued Network

In our approach to TSP the neural network is organized by the $1 \times N$ vector $V$, such that $v_i = k$ if city $k$ is in position $i$ in the tour. Then, a solution $V = (v_1, \ldots, v_k, \ldots, v_N)$ represents that the first city to be visited is the value of $v_1$ and the $k$th city to be visited is the value of $v_k$. The last city to be visited before going back to the city $v_1$ is the city $v_N$. In the multivalued neural network approach the network has $N^N$ possible states. However, since a valid tour is represented by a permutation of $M = \{1, 2, \ldots, N\}$, the number of neural states corresponding to feasible tours is $(N - 1)!$ if we consider a fixed first city. Notice that with this representation the direction is not arbitrary. For instance, although the neural state $(A, B, C, D)$ is different from the neural state $(A, D, C, B)$, they represent the same tour.

Instead of applying a penalty method (as it is usual in Hopfield-type networks), we decide to consider the search space as the set of valid tours in order to obtain a high speed of convergence. Therefore, since the only possible states for the network are the permutations of $M$, the TSP can be solved by minimizing the simple energy function

$$E = \sum_{i=1}^{N} D(v_i, v_{i+1})$$

where $D$ is the distance function defined as $D(i,j) = d_{ij}$ and the subscripts are cyclic, such that $v_{j,N+1} = v_{j,1}$.

### 3. 2-opt and 3-opt Algorithms

The simplest and most famous of the classical local optimization algorithms for the TSP are 2-opt and 3-opt moves. These algorithms provide the essential building blocks used by many researchers in adapting tabu search [29] and simulated annealing [25–27] to the TSP. The 2-opt algorithm was first proposed by Croes [28] and is motivated by the fact that if two edges cross in an Euclidean TSP, the tour can be improved by removing the edges that cross and reconnecting the resulting two paths by edges that do not cross. Improvements can also be made using this move if edges are not crossed and for non-euclidean problems. To have more flexi-
bility for modifying the current tour we can break the tour into three parts instead of only two and combine the resulting paths in the best possible way. Such a modification is called 3-opt move. In general, we can remove $k$ edges and reconnect the resulting $k$ paths in a new manner. The rapid growth, both in the number of ways to form the $k$ paths and in the number of ways to reconnect them, shows the impossibility of guaranteeing that a tour is $k$-optimal for even moderate values of $k$.

In order to obtain shorter running times for 2-opt and 3-opt there are simple and effective ways to avoid looking all possible moves in the search for one that shortens the tour, first noticed by Lin and Kernighan [10]. In order to exploit these techniques a data structure is needed to allow us to quickly identify the allowable candidates. However, the drawback to this approach is that it takes much time to set up the lists and much space to store them. Even with some speedups, the construction time for neighbor lists dominates the overall running time for the usual neighbor list implementations of 2-opt and 3-opt, as pointed by Johnson and McGeoch [24]. In Euclidean TSPs using truncated neighbor lists can significantly reduce the running time with a very little loss of tour quality [24]. Thus the running time bottleneck becomes usually the time to perform the moves that are found, which depends intimately on the method used for representing the current tour.

We present the discrete multivalued neural network as an effective representation of the TSP and a very easy way to implement the moves involved in 2- and 3-opt algorithms. Without the neighbor lists we obtain for the TSP very short running times as shown in Section 5.

4. The Proposed N-City TSP Multivalued Network

The proposed neural network consists of a single layer of $N$ interconnected neurons or processing elements. The state of neuron $i$ is denoted by $v_i(k)$, where $k$ denotes discrete time. Each neuron can take one of the discrete states $M = \{1, 2, \ldots, N\}$. $\omega_{ij}$ is a real number that represents the interconnection strength between neurons $i$ and $j$, for $i, j = 1, 2, \ldots, N$.

As a generalization of the energy function that characterizes the binary discrete Hopfield network

$$ E(k) = -\frac{1}{2} \sum_i \sum_j \omega_{ij} v_i(k) v_j(k) $$

we introduce the following new energy function for the proposed discrete multivalued network

$$ E = -\frac{1}{2} \sum_i \sum_j \omega_{ij} f(v_i(k), v_j(k)) \quad (2) $$

where $f(v_i(k), v_j(k))$ is a ‘similarity function’ that depends on the problem.
Consider now that we apply this multivalued network for solving the N-city symmetric TSP and only moves between feasible states, i.e., permutations of $M$, are allowed. By comparing the general energy function (2) of the multivalued network and the energy function (1) defined for solving the TSP, we derive the ‘similarity function’ $f(v_i, v_j) = -2D(v_i, v_j)$, where $D$ is the distance function. It is also derived the weight matrix

$$W = (\omega_{ij}) = \begin{pmatrix}
0 & 1 & 0 & 0 & \ldots & 0 & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \ldots & 1 & 0 \\
0 & 0 & 0 & 0 & \ldots & 0 & 1 \\
1 & 0 & 0 & 0 & \ldots & 0 & 0
\end{pmatrix}$$

From a given initial state $V(0) \in \theta$ (the set of permutations of $M = \{1, 2, \ldots, N\}$) that represents a feasible tour we are going to define updating rules that generate improved feasible tours.

DEFINITION. Given a feasible state of the N-city TSP multivalued network $V(k) \in \theta$, we define the reverse factor between neurons $m$ and $p$ at time $k$ as

$$RF_{mp}(k) = D(v_{m-1}(k), v_m(k)) + D(v_p(k), v_{p+1}(k)) - D(v_m(k), v_{p+1}(k)) - D(v_{m-1}(k), v_p(k))$$

Observe that this factor represents the variation of the tour length generated if the path between the cities $v_m$ and $v_p$ is reversed and reconnected in the other possible way.

THEOREM (one-path updating rule). Let us consider a sequence of neural index subsets $I(k) = \{m, m+1, \ldots, p\}$, $p > m$ and a feasible initial state of the N-city TSP multivalued network $V(0) \in \theta$. If the neural states are updated in the following form:

$$v_i(k+1) = \begin{cases} 
  v_{m+p-i}(k) & \text{if } RF_{mp}(k) < 0 \\
  v_i(k) & \text{otherwise}
\end{cases}$$

for $i \in I(k)$

then a new feasible state $V(k+1) \in \theta$ is obtained. Moreover, it is guaranteed that $\Delta E(k) \leq 0$ for any $k \geq 0$ and after a finite number of updates the network converges to a fixed point.

Proof. Observe that the sequence of neural index subsets $I(k) = \{m, \ldots, p\}$ represents the path given by the neuron states $\{v_m(k), \ldots, v_p(k)\}$ where an orientation of the tour is naturally given by the neural representation. If we apply the updating rule given by (3) we are actually removing the existing path from the city $v_{m-1}(k)$ to the city $v_m(k)$ and the existing path from $v_p(k)$ to $v_{p+1}(k)$. We also add
a new path from city \( v_m(k) \) to city \( v_p(k) \) and another new path from \( v_m(k) \) to \( v_{p+1}(k) \).

Observe that (3) also involves a reversal of the direction of the current path \{\( v_m(k), \ldots, v_p(k) \)\}.

Since (3) generates a new permutation of the neural states, from a feasible state \( V(k) \in \theta \) we obtain a new state \( V(k+1) \in \theta \) that also represents a valid tour. It is straightforward to verify that if \( V(k+1) = V(k) \) then \( \Delta E(k) = E(k+1) - E(k) = 0 \), and if \( V(k+1) \neq V(k) \) then \( \Delta E(k) = RF_{mp}(k) < 0 \). Thus given an initial state that represents a feasible tour, the network is updated so long as it reduces the energy function (the length of the current tour), until a feasible state is reached for which no updating yields an improvement (a locally optimal tour).

The '1-path updating rule' dynamics proposed in the theorem involve the updating of the neurons \( I(k) = \{m, \ldots, p\} \) considered as a single directed path of the current tour. If we divide this path into two directed paths we are actually considering two sequence of neural index subsets \( I_1(k) = \{m, \ldots, q\} \) and \( I_2(k) = \{q+1, \ldots, p\} \), where \( m < q < p \). Therefore the dynamics (3) can be easily generalized to a '2-path updating rule' if we update simultaneously both groups of neurons \( I_1 \) and \( I_2 \) by the 1-path updating rule. For simplicity let us call \( I \) a sequence of neurons \( I \) updated by (3), when \( V(k+1) \neq V(k) \). Thus we can update both groups of neurons in one of the eight alternatives arising \( \{I_1, I_2, I_1T_2, I_1T_2, I_1T_1, I_2T_1, I_2T_1, I_2T_1\} \). We shall say that the \( N \)-city TSP multivalued network is updated by a 2-path updating rule if the energy change generated by every alternative is computed and the network updated to the best state found. In this way the network will converge to a stable state in the fewest number of steps. These dynamics actually represent a 3-opt move, in which three nonadjacent edges are broken and the resulting three paths are reconnected in the best of the eight possible alternatives. It is clear that each of the more elaborated updating rules is capable of producing higher-quality solutions, but at the same time the computational time increases.

5. Simulation Results

We have tested both algorithms described in the preceding section by solving problem instances taken from TSPLIB, the well-known data library collected by Reinelt [11]. The selected test problems are Euclidean TSP instances where the sizes of the problems range between 51 and 532. To facilitate subsequent computation time comparisons all experiments were run on a conventional 233 MHz Pentium II PC with 64 MBytes RAM by Matlab. Table I shows the average tour quality and the best tour quality found with the multivalued neural network using 1-path updating rule. Here we define the quality of a tour as the percentage that its length is above the length of the optimal tour. The first column in Table I specifies the instance as is referred in TSPLIB and the second column shows the optimal solution for each instance. The last column indicates the average running time in seconds. The
results were obtained for a total of 5000 runs performed from randomly generated initial states, i.e., random permutations of $M = \{1, 2, \ldots, N\}$.

Notice that accurate solutions are obtained in very short running times. For instance, an average computation time of 2.3 s is needed to solve a 105-city TSP problem (lin105), where the best tour found is 0.8% longer than the optimal tour and the average tour length is 8.3% longer than the optimal tour. Results for the 2-path updating rule based on 50 runs are also shown in Table VI. It is observed that this updating rule produces better solutions, but at the same time the average running times are increased. Thus an average computation time of 205.8 s is needed to solve the 105-city TSP problem, where the best tour found is the optimal tour and the average tour length is 0.82% longer than the optimal tour.

In comparing our results we have found that in the literature almost all the reported results on the Neural Network algorithms for the TSP are based on randomly generated problems and usually there is no description of computation time. An exception is the investigation into the improvement of local minima of the Hopfield network presented by Peng et al. [20]. Experiments on the TSP show that their local minima escape algorithm (LME) produces better results than Hopfield network with simulated annealing in less time. However, their experiment with a 51-city TSP problem from TSPLIB shows that for a total of 10 runs the average tour length is 25.1% longer than the optimal tour where the average computation time is 4.5 h on a HP7000 workstation. Although these results greatly improve the Hopfield networks’ solution, they can not be compared in terms of quality and computation time to those obtained using the multivalued network, which takes some seconds on a conventional PC to produce accurate solutions. Table II shows the results reported by [20] for the 51-city TSP problem (eil51) by using the Hopfield network with simulated annealing based on 5 runs and by using LME for a total of 10
runs. These results are compared to the results obtained by our multivalued network with 1- and 2-path updating rules based on 10 runs. For a 101-city TSP problem (eil101) the usual Hopfield network provides an average tour length 416.5% longer than the optimal tour as reported by Peng et al. They improve these results by applying the LME algorithm and report an average tour length 43.2% longer than the optimal tour and a shortest tour 37.5% longer than the optimal tour where the computation time is not given. As it takes too much time (ten of days) to use the Hopfield network with simulated annealing no comparison to this method was reported. In contrast, our multivalued network with the 2-path updating rule provides an average tour length 3.7% longer than the optimal tour and takes an average time of 373.7s as shown in Table VI.

Recently, a neural network that improves the results of the algorithm of Peng et al. has been presented by Martín-Valdivia et al. [22]. This method combines the Hopfield-type network based on augmented Lagrange multipliers (ALH) presented by Li [21] and the LME algorithm of Peng et al. [20], improving both of them. Comparison of the results obtained for the 51 and 101 cities TSP problems are presented in Tables III and IV, where all the results were obtained for a total of 10 runs.

### Table II. Results obtained for a 51-city problem using the multivalued network with 1- and 2-path updating rule, using the Hopfield network with simulated annealing and using the LME method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Quality (%)</th>
<th>Maximum Quality (%)</th>
<th>Minimum Quality (%)</th>
<th>Average time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopf. + SA</td>
<td>101.8</td>
<td>88.97</td>
<td>109.6</td>
<td>16 hours</td>
</tr>
<tr>
<td>LME</td>
<td>25.1</td>
<td>16.4</td>
<td>31.9</td>
<td>4.5 hours</td>
</tr>
<tr>
<td>1-path UR</td>
<td>7.18</td>
<td>3.76</td>
<td>11.9</td>
<td>0.5 seconds</td>
</tr>
<tr>
<td>2-path UR</td>
<td>2.7</td>
<td>0.9</td>
<td>3.29</td>
<td>30.3 seconds</td>
</tr>
</tbody>
</table>

### Table III. Comparison of the results obtained using the multivalued network with 1- and 2-path updating rule and using the LME-ALH network for a 51-city problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Quality (%)</th>
<th>Maximum Quality (%)</th>
<th>Minimum Quality (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LME-ALH</td>
<td>12.4</td>
<td>9.6</td>
<td>15.7</td>
</tr>
<tr>
<td>1-path UR</td>
<td>7.18</td>
<td>3.76</td>
<td>11.9</td>
</tr>
<tr>
<td>2-path UR</td>
<td>2.7</td>
<td>0.9</td>
<td>3.29</td>
</tr>
</tbody>
</table>

### Table IV. Comparison of the results obtained using the multivalued network with 1- and 2-path updating rule and using the LME-ALH network for a 101-city problem.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average Quality (%)</th>
<th>Maximum Quality (%)</th>
<th>Minimum Quality (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LME-ALH</td>
<td>31.8</td>
<td>23.7</td>
<td>39.1</td>
</tr>
<tr>
<td>1-path UR</td>
<td>10.33</td>
<td>6.99</td>
<td>13.67</td>
</tr>
<tr>
<td>2-path UR</td>
<td>3.74</td>
<td>2.54</td>
<td>5.4</td>
</tr>
</tbody>
</table>
Observe that the multivalued network with 1-path and 2-path updating rules produces much better results than the very recent LME-ALH algorithm without the fine-tuning of parameters required in these analog Hopfield models.

The largest TSP problem attempted by a Hopfield-type network for which the results are published seems to be the solution for a 318-city TSP problem reported by Mehta and Fulop [30]. However, the result was significantly away from the best known cost since an average tour length 48.6% longer than the optimal tour is reported. As a conclusion it is observed that continuous Hopfield-type models for the TSP are usually very slow and do not provide good solutions for large-scale problems.

We have also compared our multivalued neural network to the discrete multivalued MAREN network presented by Erdem and Ozturk [19]. They reported results for 10, 20, 30, 50 and 80 cities TSP problems, where location of cities were chosen at random on the interior of a square of edge length 1. The second and fourth columns in Table V indicate the best tours of 100 trials with the multivalued MAREN network and with the 1-path updating rule multivalued network respectively. Columns 3 and 5 give the percentage of 100 network simulations that produced valid tours, obviously always 100% in our multivalued network which moves only between valid tours. The simulation results in Table V indicate that our multivalued network with the simple 1-path updating rule produces much better results than MAREN for the TSP.

Of all the neural network methods that are available for solving the TSP, it is believed that the algorithms derived from the Kohonen’s self organizing map are both the most accurate and the fastest. Recently, Aras et al. [13] have presented a new self-organizing neural network for solving the TSP called KNIES. This paper is also of documentary importance because many problem instances taken from TSPLIB are solved by using the Pure Kohonen Network [15], the Guilty Net [14] and the approach of Angéniol et al. [31]. Their experimental results indicate that KNIES is the most accurate neural network strategy for the TSP currently reported.

Notice that Aras et al. [13] point out that the results that they report are the best values recorded after running their algorithm for a large number of parameter settings, where four parameters have to be defined. Our simulation results show that on every instance of TSPLIB the best solution found using the multivalued network

<table>
<thead>
<tr>
<th>Cities</th>
<th>MAREN network</th>
<th>1-path UR network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best tour</td>
<td>% valid</td>
</tr>
<tr>
<td>10</td>
<td>3.14</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>5.39</td>
<td>72</td>
</tr>
<tr>
<td>30</td>
<td>6.49</td>
<td>64</td>
</tr>
<tr>
<td>50</td>
<td>11.84</td>
<td>83</td>
</tr>
</tbody>
</table>
with 1-path updating rule is better than the best solution found by KNIES, excepting for the KroA200 and pr124 instances. If we compare now the multivalued network with 2-path updating rule to the KNIES network, as shown in Table VI, it is observed that not only our best tour is always better than the best tour reported by Aras et al. on every instance, but even our average tour is usually better than the best tour found by KNIES. Notice that the results reported in Table VI were obtained by performing 50 runs from random initial states of the multivalued network with 2-path updating rule, where no fine-tuning of any parameter is needed. The average running time in seconds on a conventional Pentium PC is reported in column 6, where no comparison with KNIES is given since Aras et al. do not give description of computation times.

### Table VI
Comparison of the results obtained using the multivalued network with 2-path updating rule and using the KNIES network for various TSP instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>KNIES 2-path UR network</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Best Quality</td>
</tr>
<tr>
<td>eil51</td>
<td>2.86</td>
</tr>
<tr>
<td>eil101</td>
<td>4.66</td>
</tr>
<tr>
<td>st70</td>
<td>1.51</td>
</tr>
<tr>
<td>att532</td>
<td>6.74</td>
</tr>
<tr>
<td>bier127</td>
<td>2.76</td>
</tr>
<tr>
<td>eil76</td>
<td>4.98</td>
</tr>
<tr>
<td>kroA200</td>
<td>5.71</td>
</tr>
<tr>
<td>lin105</td>
<td>1.29</td>
</tr>
<tr>
<td>pcp442</td>
<td>10.44</td>
</tr>
<tr>
<td>pr107</td>
<td>0.42</td>
</tr>
<tr>
<td>pr124</td>
<td>0.08</td>
</tr>
<tr>
<td>pr136</td>
<td>4.53</td>
</tr>
<tr>
<td>pr152</td>
<td>0.97</td>
</tr>
<tr>
<td>rat195</td>
<td>11.92</td>
</tr>
<tr>
<td>rd100</td>
<td>2.09</td>
</tr>
</tbody>
</table>

### 6. Conclusion
In this paper a discrete multivalued Hopfield-type neural network for solving the TSP has been proposed. In the traditional Hopfield neural network for the TSP, every possible solution is mapped into a network of $N \times N$ neurons with (0,1)-binary outputs. However, multivalued neurons generate an easier representation, since every solution of the TSP is mapped into a network of $N$ neurons, where each neuron can take one of $N$ discrete outputs. Thus, the multivalued network for the TSP presented in this paper enables an easy implementation of updating rules based on heuristic techniques. Therefore, the dynamics for the network guaranteeing the energy decrease are derived from the efficient well-known 2- and 3-optimal moves.

Experimental results for problems taken from TSPLIB show that it is the most accurate neural network strategy for the TSP currently reported and probably
the fastest. It indicates that for neural network algorithms the hybrids with classical optimization techniques are one of the best ways to go if one wants to handle larger instances and obtain reasonably good tours. In future we plan to show the efficiency of the multivalued network in solving more combinatorial optimization problems.

Acknowledgement

This work was supported in part by the Comisión Interministerial de Ciencia y Tecnología (CICYT) under Grant TAP99-0629.

References