An asynchronous recurrent linear threshold network approach to solving the traveling salesman problem

E.J. Teoh\textsuperscript{a,*}, K.C. Tan\textsuperscript{a}, H.J. Tang\textsuperscript{b}, C. Xiang\textsuperscript{a}, C.K. Goh\textsuperscript{a}

\textsuperscript{a}Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore
\textsuperscript{b}Queensland Brain Institute, Faculty of Biological and Chemical Sciences, University of Queensland, Australia

Received 10 October 2006; received in revised form 4 June 2007; accepted 9 June 2007
Communicated by Z. Wang
Available online 12 July 2007

Abstract

In this paper, an approach to solving the classical Traveling Salesman Problem (TSP) using a recurrent network of linear threshold (LT) neurons is proposed. It maps the classical TSP onto a single-layered recurrent neural network by embedding the constraints of the problem directly into the dynamics of the network. The proposed method differs from the classical Hopfield network in the update of state dynamics as well as the use of network activation function. Furthermore, parameter settings for the proposed network are obtained using a genetic algorithm, which ensure a stable convergence of the network for different problems. Simulation results illustrate that the proposed network performs better than the classical Hopfield network for optimization.

© 2007 Elsevier B.V. All rights reserved.

Keywords: Linear threshold neurons; Recurrent neural networks; Hopfield model; Traveling salesman problem; Genetic algorithms

1. Introduction

This article is primarily focused on the mapping of combinatorial optimization problems [25] onto a linear threshold (LT) network, where an approach to solving the classical Traveling Salesman Problem (TSP) using a recurrent network of LT neurons is proposed. Although LT-type neurons were first examined in Hartline and Ratliff’s Nobel prize-winning work in 1958 during a study of the limulus in the anatomical make-up of the eye [15], only in recent years has interest in such neurons been growing. The lack of an upper saturating state in neurons has been shown to exclude spurious ambiguous states by Feng and Hadeler [9], as well as to guarantee the presence of a unique stationary state. However, this non-saturating characteristic of a neuron increases the likelihood that the network dynamics may be unbounded, as well as the possibility that no equilibrium point might exist [10].

Nevertheless, further work by Feng [8] using the theory of supermartingales in developing a Lyapunov-based function for asynchronous neural networks with non-differentiable input–output characteristics, has demonstrated the existence of a convergence condition by proving that a Lyapunov function for such a network is equivalent to one that is differentiable. Hahnloser [12] subsequently analyzed the computational abilities and dynamics of LT networks, particularly in explaining Winner-Take-All (WTA) multistability of the network with self-excitation and global inhibition, as well as oscillating behavior of the dynamics when the inhibition delay of the network is increased in an asymmetrical network.

Of more recent interest, Hahnloser et al. [13] demonstrated a functional silicon-based circuit design for a LT network that is able to verify the theory of digital selection (groups of active and inactive neurons) and analog amplification (active neurons are amplified in magnitude, as represented by the gain in neuronal activity). Necessary and sufficient conditions for generalized networks of LT neurons, for both symmetrical and asymmetrical weight matrices were established in works by Wersing et al. [34]
and Yi et al. [36]. Meanwhile, Xie [35] and Hahnloser[14]
put forward the underlying framework for analyzing
networks of LT neurons by examining the system
eigenvalues and classifying neurons according to permitted
and forbidden sets, where subsets of permitted groups of
neurons are permitted, and supersets of forbidden groups
of neurons are forbidden. This is a consequence of the
digital selection abilities of the network. Further analysis of
the dynamics of LT neurons in a recurrent network is
addressed in Tan[31] with application in analog associative
memories pursued in Tang[33].

The original TSP requires that each city and hop taken is
passed through exactly once. This means that there can
only be a single active neuron in each row and column. Any
city cannot be in more than one hop in a valid tour.
Conversely, in any hop (column), only one city can be
visited. The TSP can be formulated as follows: given a
finite number of cities, together with the distance (cost)
between city x and y (denoted by \(d_{xy}\)), find the optimal
sequence of cities that the salesman should visit such that
the total distance taken to travel to all cities exactly once
(and returning to the starting city) is the minimum. For the
problem being considered here, and consistent with the
focus of this paper on symmetrical networks, the symme-
trical-TSP is considered, i.e. \(d_{xy} = d_{yx}\). A more
general formulation is to treat the cost of traversing a pair of cities
unequally, i.e. the cost of traveling from city x to y is
different from that of traveling from city y to x, \((d_{xy} \neq d_{yx})\).
Mathematically, this is similar to a quadratic 0–1
programming problem with lineal constraints, i.e.

\[
\text{Minimize} \quad \frac{1}{2} \sum_{x} \sum_{y \neq x} \sum_{j=1}^{n} \delta(v_{x,j})d_{xy}(v_{x,j}+1) + d_{y,j+1}) \quad (1)
\]

subject to

\[
\begin{align*}
\sum_{y \neq x} \delta(v_{x,j}) &= 1 & \forall i \in \{1, 2, \ldots, n\}, \\
\sum_{j=1}^{n} \delta(v_{x,j}) &= 1 & \forall x \in \{1, 2, \ldots, n\} \quad (3)
\end{align*}
\]

and a redundant constraint \(\sum_{j=1}^{n} \delta(v_{i,j}) = n\), where
\(\delta(v_{i,j}) = 1\) if \(v_{i,j} > 0\) and \(\delta(v_{i,j}) = 0\) otherwise. The matrix
\(v \in \mathbb{R}^{n \times n}\) is the solution, or assignment matrix consisting of
the entries \(v_{x,j} \in [0, \infty)\), which in turn represents the
activation of the neuron in the xth row and ith column.
An active neuron, defined as a nonnegative activation in
the xth row and ith column corresponds to city x being
assigned to the ith hop.

Eq. (1) defines the tour length, the quantity which is to be
minimized since it is the length of the tour a salesman
would make when visiting the cities in the order specified
by the permutation, returning at the end to the initial city.
For n cities, this tour would involve n hops, where each hop
is the route taken (a straight line for a Euclidean problem)
between one city and the next. Alternatively, using
nomenclature from graph theory, each hop is also known
as an edge or arc, and the cities known as a vertices
(singular: vertex) or nodes. The final sequence of cities is
known as a tour. A valid tour solution satisfies the
constraints of Eqs. (2) and (3). A sufficiently good solution
would in turn be a tour which minimizes Eq. (1). See Fig. 1
for a simple illustration of the weight connections between
cities in a 6-city example.

For a symmetrical network, where the distance from city
x to city y is equal in both directions, i.e. \(d_{xy} = d_{yx}\), there are
\(2(n-1)!\) possible tours (a specified tour describes
the order in which the n cities are visited, thus a problem
with n cities would have n tours of equal length in one
direction, and another n tours of equal length in the other
direction, leaving 2n degenerate tours). Conversely, for an
asymmetrical network, \((d_{xy} \neq d_{yx})\), there are \((n-1)!\)
possible tours. An optimum solution would be a tour
solution of the least (minimum) length, while an arbitrarily
good tour solution is one whose distance traversed is within
a range of the optimum distance.

It is apparent that an exhaustive search is not possible
for large n in practice. It is interesting, however, to note
that near-optimal tours can be vastly different, especially
for larger-sized problems. To improve on a good solution,
many modifications to the problem approach need to be
done, hence the intractability of the TSP. To many, solving
the TSP seems simple enough, but the key difference here is
that humans perceive the problem differently, from a 2-D
point of view that provides a perspective that allows a
rather trivial solution (which might be significantly harder
for large n). Notions of distances vary according to how the
problem is set up. It is thus not necessary for distances to
be defined strictly in a Euclidean sense. Any appropriate
measures of cost such as time, economical cost, etc., are
possible. Practical applications of the TSP include con-
straint satisfaction, packing, assignment, scheduling, and
routing problems [27,16].

The use of neural networks in solving combinatorial
optimization problems have been largely inspired by
Hopfield and Tank’s implementation of a single-layered
recurrent neural network [19,20], which was in turn drawn
from Hopfield’s earlier work on content-addressable
memory (CAM) [17,18]. The TSP is one of the more
representative polytopes of problems in combinatorial optimization, and is often used as a test-bed for evaluating many proposed solutions and approaches for combinatorial optimization [16], making the TSP a particularly interesting optimization problem for neural networks. The iterative nature of solving such problems necessitates the use of recurrency in the network architecture, in which Hopfield and Tank’s original approach was based on the use of penalty terms with an inhibitory weight matrix, to solve the mapped constrained optimization problem of the TSP. Constraints exist in the form of permissible numbers of active neurons per row and column of the solution matrix. The dynamics of the network is such that its energy function is minimized. Analogously, the energy function is equivalent to the cost or objective function of the original problem. In the TSP, the single objective is to minimize the cost represented by the total distance of the resulting tour. While the initial method proposed by Hopfield and Tank was fraught with some difficulties particularly with the number of invalid solutions found, many improvements using a variety of techniques have been used to obtain improved results.

Previous neural network methods for solving TSPs often involved the use of a network of saturating neurons with bounded activities [19]. However, it is believed that the use of a class of continuous-valued, neurons without an upper saturation state is able to exclude the problem of spurious ambiguous states in WTA dynamics [12,26,34]. Self-organizing neural networks inspired by Kohonen [22], using an elastic net approach [7] have also been employed.

In solving the TSP using a neural network approach, there is an inherent trade-off between a network that has parameters tuned to obtain very good quality solutions and one that has been tuned to obtain mostly valid solutions [23,30]. For the interested reader, Smith (and references therein) [27] provides a fairly comprehensive overview on the evolution of neural networks in solving a variety of combinatorial optimization problems such as the TSP, while Ramanujam [25] discusses the general approach of mapping combinatorial type problems onto neural networks. As Smith has concluded, research is still continuing to improve the many approaches that have been inspired from Hopfield’s seminal work [27], and further asserts that hybridization of these approaches, such as that between neural networks and genetic algorithms attempted by this article, could possibly yield better results.

The organization of this paper is as follows: Section 2 will present the proposed approach to solving the TSP using a recurrent LT network as well as its corresponding state update dynamics. Optimization of the network parameters using a genetic algorithm is then described in Section 3. A comparative study of the proposed method, together with its GA-optimized parameters is subsequently drawn with the classical Hopfield network [19] in Section 4. Consequently in Section 5, a discussion on the network dynamics of the proposed approach, pertinent to the scalability and setting of the parameters, is presented. A summary, in Section 6 then concludes this article.

2. Solving TSP using a recurrent LT network

For an n-city problem, there are \(n^3\) possible connections in the network; \(n^2\) connections between neurons in a single layer, and connections between neurons between the \(n\) layers. In other words, an \(n\)-city problem has \(n^2\) neurons with a weight matrix of \(n^3\) connections. According to [16], the use of binary threshold neurons does not provide good results as the dynamics will rapidly converge onto a local minimum with a poor tour length; hence either stochastic units with simulated annealing, or continuous-valued neurons should be used. This can be explained in terms of granularity of binary neurons versus that of analog neurons. While binary neurons are more nonlinear in the sense that they arrive at decisions more readily (either 1 or 0 states with no values in-between), they do not represent transitional states of the network well. Analog neurons, on the other hand are continuous-valued, hence provide more informative and thus useful values for intermediary computation.

Often, the optimization problem at hand needs to be mapped onto a neural network, such that the dynamics or activities of the network can be interpreted as the result, or solution to the originally posed problem. This mapping process usually involves the setting up of an appropriate cost, or objective function which in turn can be represented as an energy function in the dynamics of the neural network. The solution in turn is most often a discrete answer, although the neurons perform analog computations [19]. In the original formulation of the TSP [19], negative weights (inhibitive connections) were used between neurons in the network. However, in the proposed network of LT neurons, a positive weight matrix \(W \in \mathbb{R}^{n \times n}\) (with zero entries along the diagonal) and zero inputs \((b_{x,i} = 0 \ \forall x, i \in n)\) is used.

2.1. LT neurons

In recent years, a class of neurons which behaves in a linear manner above a threshold, known as linear threshold (LT) or threshold linear neurons, have received increased attention in the realm of neural computation. Previously, saturation points were modeled in activation functions to account for the refractory period of biological neurons when it ‘recharges’, and cannot be activated. Computational neuroscientists believe that artificial neurons based on an LT activation function are more appropriate for modeling actual biological neurons. This is quite a radical departure from the traditional line of thought which assumed that biological neurons operate close to their saturating points, as conventional approaches to neural networks were usually based on saturating neurons with upper and lower bounded activities. Studies have demonstrated that cortical neurons rarely operate close to their
saturating points, even in the presence of strong recurrent excitation \[6\]. This suggests that although saturation is present, most (stable) neural activities will seldom reach these levels.

An LT neuron is a non-differentiable, unbounded activation function that behaves linearly above a certain threshold, similar in functionality to (half-wave) rectification in circuit theory. The general form of the LT activation function is defined as

\[ \sigma(x) = [x]^+ = k \times \max(\theta, x), \]  

where \( k \) and \( \theta \), respectively, denotes the gain (usually 1) and threshold (usually 0), of the activation function (see Fig. 2). This form of nonlinearity is also known as threshold, or rectification nonlinearity.

2.2. Modified formulation with embedded constraints

Unlike the Hopfield model \[19\] where continuous, bounded neurons were used, this paper proposes an approach using a network of LT neurons. It is believed that using LT neurons in a WTA arrangement reduces the possibility of invalid tours as there is more granularity, or resolution in the neural activities; since the neural activities are not bounded, stable network dynamics will ultimately converge to a set of active neurons with a range of analog values. Hence, the criteria for convergence can be stricter to allow for the possibility that the neural activities can assume any real nonnegative value, instead of requiring that the activities of neurons be represented by the neural activity of 0 or 1. See Section 5.4 on conditions for convergence.

Let \( v \) denote the set of \( n \times n \) neurons indicating the tour specified by the network. For an \( n \)-city problem, \( W \in \mathbb{R}^{n \times n} \) and \( v \in \mathbb{R}^{n \times n} \); a valid tour solution would mean that exactly \( n \) neurons are active in \( v \), with the row and column constraints specified by Eqs. (2) and (3). For example, a 6-city problem that has the final tour solution as shown in Fig. 3 has the tour path as follows: \( 1 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow 1 \). Note that each column and row has exactly one active neuron.

An important stage involved in attempting to map a constrained optimization problem onto a neural network implementation is to determine appropriately, how to embed the given problem constraints into the dynamics (motion equation) of the network. Specifically, in addressing this issue for the TSP using a network of LT neurons, the implied idea is to restrict the activation of neurons along a row or column of \( v \) which is already active, unless conditions are favorable (e.g. the support it receives from neighboring hops is greater than the inhibition it receives from active neurons along the same row or column). The next subsection further elucidates this idea.

2.3. State update dynamics

The constraints of the problem is directly embedded into the state dynamics of the network. Let \( x \) and \( y \) denote the identities of the \( n \) cities, and \( i \) and \( j \) represent the \( n \) possible hops taken \((x, y, i, j \in n)\). \([x] = x \) if \( x > \theta \) (where \( \theta \) is the threshold and is usually taken to be 0) and 0 otherwise. This form of an activation function is linear threshold (or threshold linear) and is said to have rectification nonlinearity (linear behavior only occurs above the threshold). Let \( d \in \mathbb{R}^{n \times n} \) represent the square, symmetrical
distance matrix between all \( n \) nodes of the network. Then the weight matrix \( W \) of the network is set-up as follows (\( W \) is a nonnegative matrix, where all off-diagonal entries are strictly positive while the components are zero along the diagonal):

\[
W = \max(d) + \min(d) - d.
\]  

(5)

The equation of motion of the LT network dynamics is as follows (one can assume that the time constant of the dynamics \( \tau = 1 \)), where a neuron \( x \) (in hop/path \( i \)) randomly selected for asynchronous update. Recall that \( u(x,y) \in \mathbb{R}^n \) (all real values) is the pre-activation, and \( v(x,y) \in \mathbb{R}^{0,\infty} \) (all real, nonnegative values) is the activation. Also, \( k_0, k_1, k_2, k_3 \in \mathbb{R}^+ \) are all real, positive values:

\[
u_{x,i} = k_0 \xi_x - k_1 \xi_1 - k_2 \xi_2 + k_3 \xi_3, \]

(6a)

\[
u_{x,i} = [u_{x,i}]^+, \]

(6b)

\[rac{\text{d}u_{x,i}}{\text{d}t} = -\frac{u_{x,i}}{\tau} + v_{x,i}, \]

(6c)

where

\[
\xi_0 = \sum_{j \neq i} w_{x,j} (v_{y,j-1} + v_{y,j+1}), \]

(7a)

\[
\xi_1 = \sum_{j \neq i} v_{x,j}, \]

(7b)

\[
\xi_2 = \sum_{j \neq i} v_{x,j}, \]

(7c)

\[
\xi_3 = \left( 2 - \left( \delta \left( \sum_{j} v_{y,j} \right) + \delta \left( \sum_{j} v_{y,j} \right) \right) \right). \]

(7d)

\( \delta(z) \) represents a step function where \( \delta(z) = 1 \) if \( z > 0 \) and 0 otherwise. The first term (\( \xi_0 \)) represent the support a randomly selected neuron in hop \( i \) receives from neighboring hops (\( i - 1 \) and \( i + 1 \)); the second (\( \xi_1 \)) and third terms (\( \xi_2 \)) denote the row and columnar constraints, respectively, while the fourth term (\( \xi_3 \)) forces the network to have exactly \( n \) active neurons. The inhibitive terms (Eqs. (7b) and (7c)) penalize neurons in the same row or column that are active. Neurons having positive activations at convergence are then selected as the active neurons. This is collectively represented in the \( v \) matrix. As was previously highlighted, feasible or valid solutions require that only one neuron per row and column are active at steady state.

In an ideal situation, we would like to avoid specifying the weighted penalty terms, and let the network attempt to solve the problem to the best of its ability without human intervention solely based on the dynamical update and states of the network itself. However, in our formulation, these penalty terms are often derived heuristically, based on the size of the problem (it becomes substantially more difficult to determine appropriate settings of these penalty terms for larger-sized problems). In particular, a genetic algorithm is used to assist in the process of obtaining a set of penalty terms that is valid for a particular problem size. This approach is outlined in the next section. Note, however, that these penalty terms (also known as network parameters in this article) are not unique.

3. Evolving network parameters using genetic algorithms

Determining the appropriate values for \( k_0, k_1, k_2, k_3 \) in Eq. (6a) is a non-trivial task. A ‘pre-processing’ stage using a simple genetic algorithm approach was used to obtain these values. Genetic algorithms (GAs) are stochastic search methods that simulate the process of biological evolution. The principle of ‘natural selection’ forms the core of such algorithms, enabling it to maintain the necessary evolutionary force to make progressive improvement towards the optima. In contrast to traditional operation research and heuristical methods, GAs are capable of sampling multiple potential solutions simultaneously. Furthermore, operations such as crossover encourage the exchange of information between individuals, giving the GA a global perspective of the optimization process. Intuitively, GA is more effective in dealing with local optimal traps. In addition, the exploitation of both global and local information allows the GA to perform a much more effective optimization process.

Thus, it is not surprising that GAs have found increasing applications and interest in many practical problems as the tuning of parameters for process controllers, drug scheduling, optimization of neural network weights and design set up in the industry. In fact, GAs have been applied to different problems of neural network design such as the optimization of network parameters, network structures as well as to the training of known architectures. As was highlighted in the previous section, the network parameters of \( k_0, k_1, k_2, k_3 \) are presently determined using a trial-and-error approach. Such an approach becomes infeasible when extended to larger-size problems. With this in mind, a GA-based approach is used to determine the appropriate values of \( k_0, k_1, k_2, k_3 \) for different problems.

This section is solely meant to illustrate the suitability of the parameters that were selected (\( k_i \forall i = 1,2,3,4 \)). In other words, the optimization of the parameters through the use of a simple GA is not meant to be part of the proposed approach, but rather to ‘verify’ the appropriateness of the parameters that were used in the simulation runs. The use of a GA in evolving candidate solutions for solving the TSP has long been a research topic in the literature, and is thus not our objective to solve the TSP using this approach.

3.1. Implementation issues

In this section, a GA approach is used to determine the appropriate values of \( k_0, k_1, k_2, k_3 \), which ensures a stable convergence of the network for different problems.
By stability, we mean that the activation values of the v matrix do not ‘explode’ (recall that LT activation functions is continuous and unbounded). The rationale of such a goal is intuitive since stable activation is a necessary condition before feasible solutions can be found and feasibility is in turn a necessary condition for optimality.

3.2. Fitness function

Considering the implementation of GA, it should be noted that the GA processes a set of encoded parameters and not the parameters itself. Hence, it provides us with the flexibility to design an appropriate representation of the potential solutions. By appropriate representation, we mean that it fulfills some criteria such as ease of implementation or exploitation of the problem structure. In this case, the parameters to be optimized are represented directly by real-number coded chromosomes. The fitness function attempts to minimize the number of epochs required to attain a stable solution. Suppose epoch is the number of epochs required by a NN to converge. Then minimizing epoch is akin to finding the best set of parameter that ensures stability. Then any reasonable solution will be able to provide a reasonable upper bound to the number of epochs required. The fitness function is given as:

\[
\text{epoch} = \min \{\text{num\_epoch}, (\text{MAX\_EPOCH} - \text{state\_stop})\} \tag{8}
\]

num\_epoch is the actual number of epoch required to attain stability, while MAX\_EPOCH is the predefined upper bound on the number of epochs and state\_stop is period of stable-state conditions prior to the end of NN training. Due to the stochastic nature of neural networks, each solution is evaluated and averaged over sample times.

3.3. Genetic operators

Crossover facilitates the exchange of information between selected individuals, allowing the evolutionary process to explore new solutions while retaining past information. In this paper, simulated binary crossover [4] is implemented to produce offspring. The mutation operator employed is the normally distributed mutation, which is a popular method. In addition, the niching mechanism proposed by Goldberg [11] to prevent genetic drift and to promote the sampling of the whole Pareto front by the population is also employed.

3.4. Elitism

The use of the elitist strategy was first introduced by De Jong [5] to preserve the best individuals found into the next generation. The objective of this policy is to prevent the losses of good individuals due to the stochastic nature of the evolution process. De Jong found that elitism could improve the performance of GAs although there is a potential danger of premature convergence. Since then, several variants of the elitist scheme have been employed in GA and showed that elitism can indeed improve the performance of the GA greatly.

In this paper, elitism is implemented using the evolving population and an archive. All the good solutions found from the evolutionary process are copied into the archive while previously archived solutions that are found to be inferior due to the introduction of better solutions are deleted. Binary tournament selection is then performed on a combined population of the evolving population and archive to fill the mating pool. Tournament selection is chosen because it eliminates the need to rank and sort the population.

3.5. Algorithm flow

The algorithm flow of the implemented GA is summarized below. In the experiments, 10 runs are performed for the design problem so as to study the statistical performance, such as consistency and robustness of the methods. Note that a random initial population is created for each of the 10 runs, and for each test problem. See Table 1 for a brief description of the simulation parameters. Actual parameter settings are listed in Table 2.

- Step 1: Initialization—Generate initiate population, and empty archive. Set gen = 0.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Genetic algorithm parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>Pop_size</td>
<td>Population size</td>
</tr>
<tr>
<td>Arc_size</td>
<td>Archive size</td>
</tr>
<tr>
<td>genNum</td>
<td>Maximum number of generations</td>
</tr>
<tr>
<td>P_c</td>
<td>Crossover rate</td>
</tr>
<tr>
<td>P_m</td>
<td>Mutation rate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Genetic algorithm parameter settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Settings</td>
</tr>
<tr>
<td>Chromosome</td>
<td>Real-number representation</td>
</tr>
<tr>
<td>Populations</td>
<td>Population size: 20; archive size: 10</td>
</tr>
<tr>
<td>Selection</td>
<td>Binary tournament selection</td>
</tr>
<tr>
<td>Crossover</td>
<td>Simulated binary crossover</td>
</tr>
<tr>
<td>operator</td>
<td></td>
</tr>
<tr>
<td>Crossover rate</td>
<td>0.95</td>
</tr>
<tr>
<td>Distribution</td>
<td>20</td>
</tr>
<tr>
<td>index</td>
<td></td>
</tr>
<tr>
<td>Mutation rate</td>
<td>0.25</td>
</tr>
<tr>
<td>Mutation (\sigma)</td>
<td>0.1</td>
</tr>
<tr>
<td>Diversity</td>
<td>Niche count with radius 0.1 in the normalized objective space</td>
</tr>
<tr>
<td>Sample</td>
<td>5</td>
</tr>
<tr>
<td>Evaluation number</td>
<td>2000</td>
</tr>
</tbody>
</table>
4. Simulation results

The following results show that the performance of the proposed LT network solves the TSP more efficiently than the classical Hopfield network for optimization. A particularly interesting result is the worst, or maximum distance of the tour solutions solved using the respective approaches, where the proposed LT network produces results that are significantly better (shorter in distance). Good tour solutions are defined to be tours whose distances are within 150% of the optimal tour distance (this performance measure was similarly used in [32,29]), which in the 10-city case is, $d_{optimal} = 2.58325$ units and $d_{150\%} = 3.87488$ units. For the 12-city double-circle problem, the optimum distance is $d_{optimal} = 12.3003$ and $d_{150\%} = 18.045045$. The performance ratio metric measures the consistency of the obtained solutions with respect to the minimum distance obtained, i.e., performance ratio = minimum distance/average distance.

4.1. 10-city TSP

A 10-city problem was solved using the proposed LT network approach, with the optimal solution of 2.58325 units shown in Fig. 4. Empirical results obtained for the proposed LT network and the classical Hopfield network are summarized in Table 3. For all runs, stopping condition of $\Delta v_{i,j} = 0$ $\forall x_i, i \in n$ for 80 iterations or $\Delta \delta (v_{i,j}) = 0$ $\forall x_i, i \in n$ for 2000 iterations is used.

Random valid tours are solution tours where at each hop, the destination city is randomly selected, without taking into account the distance matrix $d$. This is pseudo-random, as purely random tours are not considered because of the possibly large set of invalid solution tours. The use of more complex problem sets, in terms of size ($n$) and/or positioning of cities (see the 12-city double-circle TSP in the next subsection) would further differentiate the results obtained using a random approach (as explained above), the Hopfield network method and the proposed LT network. Throughout the simulations carried out for the 10-city TSP, the default values for the LT network parameters are $k_0 = \frac{1}{x}, k_1 = 1, k_2 = 1, k_3 = 1$. The values obtained for these parameters using a GA was found to be $k_0 = 0.6183, k_1 = 1.244, k_2 = 0.8531, k_3 = 1.3311$. Note the closeness of these values to the values that were found using a trial-and-error method. These GA-evolved parameter settings were shown to obtain solutions substantially faster. The difference between the LT and LT+GA approaches is that for the latter, the LT network parameters are optimized using the GA.

Fig. 5 shows a comparison of the total distance resulting from the different methods used to solve the 10-city TSP (random, Hopfield, LT, LT+GA). The distribution of the results is represented in boxplot format to visualize the distribution of the simulated results efficiently (the results being the distances of the solutions found by the different methods). The vertical axis represents the distances while the horizontal axis denotes the solution approach used. Each boxplot represents the distribution of the 100 results (runs), where a thick horizontal line within the box encodes the median while the upper and lower ends denote the upper and lower quartile, respectively. Dashed appendages illustrate the spread and shape of distribution and dots represent outside values. This boxplot clearly illustrates the improvement in the results obtained for the 10-city TSP using the proposed LT and LT+GA approach. In this particular example, the results obtained using the LT+GA method is similar to the results obtained using the LT approach, the only improvement being the shorter time required for obtaining solutions using the GA optimized parameters in the LT + GA method (see Table 3).

4.2. 12-city double-circle TSP

The solutions to a double-circle TSP, though perceptually is easy to solve, is very difficult for the original
Hopfield network to solve. For this case, 12 cities were arranged in a circle such that there are 6 cities on the inside circle and another 6 cities located just on the outside (see Fig. 6 for the optimal tour solution). Throughout the simulations carried out for the 12-city double-circle TSP, the default values for the LT network parameters are $k_0 = \frac{1}{2}$, $k_1 = \frac{1}{2}$, $k_2 = \frac{1}{2}$, $k_3 = 1$. The values obtained for these parameters using a GA was found to be $k_0 = 0.8555, k_1 = 1.5002, k_2 = 1.6078$. Again, note the closeness of these values to the values that were found using a trial-and-error method. Recall that the difference between the LT and LT + GA approaches is that for the latter, the LT network parameters are optimized using the GA. Similar to the 10-city problem, these GA-evolved parameter settings were shown to obtain solutions substantially faster (and in this case, slightly better results were obtained). The weight matrix $W$ is also scaled by a factor of $\frac{1}{4}$ to ensure that the network dynamics is stable, since unlike the previous 10-city TSP, the positions $(x, y)$-coordinates of the cities are not normalized within the unit box $[0, 1]$. The definitions for the various performance measures are the same as for the 10-city TSP explained in the previous section. Empirical results obtained for the proposed LT network and the classical Hopfield network are summarized in Table 4. For all runs, stopping condition of $\Delta d(v_i) = 0 \forall v_i, i \in n$ for 80 iterations or $\Delta d(v_i) = 0 \forall v_i, i \in n$ for 2000 iterations is used.

Histograms of the distribution of tour distances, for the four cases: the LT network approach, LT Network with GA evolved parameters the random and Hopfield methods are shown in Figs. 7–10, respectively, again illustrating the noticeable spread of tour distances using the various approaches. Fig. 8 shows the noticeable improvement obtained from using the parameters obtained from the GA.

Fig. 11 shows a comparison of the total distance resulting from the different methods used to solve the 12-city double-circle TSP (random, Hopfield, LT, LT + GA). This boxplot clearly illustrates the improvement in the results obtained for the 12-city double-circle TSP using the proposed LT and LT + GA approach. In this case, the results obtained using the LT + GA method is significantly improved.

### Table 3

Simulation results for the 10-city TSP

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>Random valid tours</th>
<th>Hopfield network</th>
<th>Proposed LT network</th>
<th>Proposed LT network w/GA evolved $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average distance</td>
<td>4.6534</td>
<td>3.8601</td>
<td>3.2608</td>
<td>3.3955</td>
</tr>
<tr>
<td>Maximum distance</td>
<td>5.6504</td>
<td>5.2216</td>
<td>4.0468</td>
<td>4.1489</td>
</tr>
<tr>
<td>Minimum distance</td>
<td>3.3153</td>
<td>2.8978</td>
<td>2.5832</td>
<td>2.6018</td>
</tr>
<tr>
<td>Valid tours</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Invalid tours</td>
<td>n/a</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Good tours</td>
<td>6</td>
<td>64</td>
<td>95</td>
<td>91</td>
</tr>
<tr>
<td>Performance ratio</td>
<td>0.7124</td>
<td>0.7507</td>
<td>0.7979</td>
<td>0.7662</td>
</tr>
<tr>
<td>Time taken (s)</td>
<td>6.219</td>
<td>54.375</td>
<td>156.078</td>
<td>82.531</td>
</tr>
</tbody>
</table>
Table 4
Simulation results for the 12-city double-circle TSP

<table>
<thead>
<tr>
<th>Performance metric</th>
<th>Random valid tours</th>
<th>Hopfield network</th>
<th>Proposed LT network</th>
<th>Proposed LT network w/GA evolved k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average distance</td>
<td>31.6149</td>
<td>22.9828</td>
<td>18.6660</td>
<td>16.1568</td>
</tr>
<tr>
<td>Maximum distance</td>
<td>41.7054</td>
<td>32.1454</td>
<td>24.4208</td>
<td>25.4873</td>
</tr>
<tr>
<td>Valid tours</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Invalid tours</td>
<td>n/a</td>
<td>6</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Good tours</td>
<td>0</td>
<td>13</td>
<td>42</td>
<td>74</td>
</tr>
<tr>
<td>Performance ratio</td>
<td>0.6879</td>
<td>0.5354</td>
<td>0.6386</td>
<td>0.7613</td>
</tr>
<tr>
<td>Time taken (s)</td>
<td>10.906</td>
<td>47.062</td>
<td>770.406</td>
<td>95.094</td>
</tr>
</tbody>
</table>

Fig. 7. Histogram of tour distances for the 12-city double-circle TSP from the proposed LT network.

Fig. 8. Histogram of tour distances for the 12-city double-circle TSP from the proposed LT network with GA-evolved parameters.

Fig. 9. Histogram of tour distances for the 12-city double-circle TSP from the random case.

Fig. 10. Histogram of tour distances for the 12-city double-circle TSP from the Hopfield case.
better than the results obtained using the LT approach, in all measured performance metrics (see Table 4).

5. Discussion

5.1. Energy function

The objective of any mapped optimization problem onto a neural network is to minimize the computational energy function of the network. The energy function not only determines the number of neurons used in the system but also the strength of the synaptic connections (weights) between these neurons. Such an energy function is constructed based on the constraints and the cost function of the original problem. For the TSP, the constraints are specified by Eqs. (2) and (3), while the cost function is given by Eq. (1). The change in the (input) state of a neuron in the network is given by the partial derivatives of the computational energy function $E$ with respect to the (output) activity of the particular neuron $(v_{i,j})$ in question. Observe that $E$ is a $n \times n$-variable function of the $n \times n$ neurons present in the system. The motion equation of the $x$th neuron in the $t$th hop is given by

$$\frac{du_{x,j}}{dt} = -u_{x,j} + \left( -\frac{\partial E}{\partial v_{x,j}} + u_{x,j} \right)^+ .$$

The total energy of the system, denoted by $E$, can be separated into four terms,

$$E = E_0 + E_1 + E_2 + E_3.$$

Individually, these computational energy terms correspond to

$$E_0 = -\frac{k_0}{2} \sum_x \sum_{i \neq j}^n w_{xy} v_{x,i} (v_{y,j-1} + v_{y,j+1}),$$

$$E_1 = \frac{k_1}{2} \sum_x \sum_{i \neq j}^n v_{x,i} \sum_{j \neq i}^n v_{x,j},$$

$$E_2 = \frac{k_2}{2} \sum_x \sum_{i \neq j}^n v_{x,i} \sum_{j \neq i}^n v_{x,j},$$

$$E_3 = -\frac{k_3}{2} \sum_x \sum_{i \neq j}^n \left( 2 - \delta \left( \sum_{j \neq i}^n v_{j,i} \right) \right) v_{x,j}.$$

The first term ($E_0$) is always non-negative for a valid tour solution since $v_{x,i} \forall x, i \in n$ is non-negative and $w_{xy}$ is positive (with zero diagonals). $E_1$ and $E_2$ is, in a similar manner, non-negative following the same argument that $v_{x,i} \forall x, i \in n$ is non-negative. The last term, $E_3$ is zero if exactly $n$ neurons are active in the whole network. The inclusion of such terms in the energy function gives rise to dynamics that is inclined toward (i) valid tour solutions, and (ii) tours with short distances.

$$\frac{\partial E}{\partial v_{x,j}} = \frac{\partial E_0}{\partial v_{x,j}} + \frac{\partial E_1}{\partial v_{x,j}} + \frac{\partial E_2}{\partial v_{x,j}} + \frac{\partial E_3}{\partial v_{x,j}},$$

$$\frac{\partial E}{\partial v_{x,j}} = k_0 v_{x,j} + k_1 v_{x,j} + k_2 v_{x,j} + k_0 v_{x,j}.$$
nonnegative value.

\[
\frac{dE}{dt} = \sum_i \sum_j \left( \frac{\partial E}{\partial u_{ij}} \right) \left( \frac{\partial u_{ij}}{ct} \right) a,
\]

(16)

\[
\frac{dE}{dt} = \sum_i \sum_j \left( \frac{\partial E}{\partial v_{ij}} \right) \left( -u_{ij} + \left[ -\frac{\partial E}{\partial v_{ij}} + u_{ij} \right] \right)^+ a.
\]

(17)

Now, two cases are possible, depending on whether the LT function (in square brackets) results in a positive activation, or because of its rectification, thresholds the activation at zero.

\[
u_{ij} > 0 : \frac{\partial E}{\partial v_{ij}} \left( \frac{\partial u_{ij}}{ct} \right) = -\frac{\partial E}{\partial v_{ij}} \Rightarrow \frac{dE}{dt} = -\sum_i \sum_j \left( \frac{\partial E}{\partial v_{ij}} \right)^2 a,
\]

(18)

\[
u_{ij} \leq 0 : \frac{\partial E}{\partial v_{ij}} \left( \frac{\partial u_{ij}}{ct} \right) = -u_{ij} \Rightarrow \frac{dE}{dt} = -\sum_i \sum_j \left( \frac{\partial E}{\partial v_{ij}} \right) u_{ij} a.
\]

(19)

In the first case, when the activation is positive, results in \( \frac{dE}{dt} \leq 0 \). However, for the second case, when the activation is thresholded at zero, \( \frac{dE}{dt} \) can be either positive or negative; this in turn depends on the value of the pre-activation \( u_{ij} \). Since \( u_{ij} \) obeys the state update equation (Eq. (15)), \( u_{ij} \) will always tend to a positive value after some time, allowing the assumption that \( u_{ij} \) is nonnegative. Because we have \( u_{ij} \leq \frac{\partial E}{\partial v_{ij}} \geq 0 \) for \( u_{ij} \geq 0 \). Then \( \frac{dE}{dt} \leq 0 \). Essentially, this means that as \( u_{ij} \) tends to a nonnegative value, the energy of the system remains stable. While \( \frac{dE}{dt} \) initially increases (this can be seen as escaping a local minimum), \( \frac{dE}{dt} \) will eventually tend to zero for appropriate settings of \( k_0, k_1, k_2, k_3 \). Typically, we choose \( k_0, k_1 \gg k_2, k_3 \), such that \( E(>0) \) is bounded from below. This has been empirically verified from the simulation results obtained.

However, to guarantee boundedness of \( E \), such that it does not tend to \(-\infty\), one needs to restrict the support term (\( \xi_0 = \sum_{y \neq x} w_{xy} (v_{y,x-1} + v_{y,x+1}) \)) of the dynamics. This in turn is dependent on the values of the weight matrix \( W \), which is a matrix with positive entries and zero diagonals. Sufficient conditions for stability of \( \frac{dE}{dt} \) is based upon the original requirements for stability in an LT network, that is \( \sum_{y \neq x} w_{xy} < 1 \). This is thus discussed subsequently. This then leads us to consider conditions for which the dynamics of Eq. (14a) will be stable. To be more specific, one needs to consider the conditions for which \( w_{xy} \) in Eq. (14b) will result in a stable value of \( \xi_0 \). Such sufficient conditions can be derived from

\[
w_{xx} + \sum_{y \neq x} w_{xy} < 1.
\]

(20)

But since \( w_{xx} = 0 \) and \( w_{xy} > 0, \forall x, y \in n \), this condition is reduced to

\[
\sum_{y \neq x} w_{xy} < 1.
\]

(21)

In other words, the network dynamics of the proposed approach can be guaranteed to be stable (but not necessarily result in valid tour solutions) if the weight matrix \( W \) derived from the distances between the cities are normalized. Hence when different scales (feet vs. meters) or magnitude of measure (1 m vs. 1 km) are used in calculating the distances between the cities prior to creating the weight matrix \( W \), attention needs to be given to ensure that \( W \) does not give rise to unstable dynamics. It is surmised that normalization would aid in scalability of the problem particularly in determining appropriate values of the network parameters. It is not the absolute values but the relative values of these distances that are important. A simple method of synaptic weight normalization is to divide the individual weight entries in \( W \) with the sum of the weights leading from, or to a particular neuron. Mathematically, this corresponds to

\[
w_{xy}' = \frac{w_{xy}}{\sum_{y \neq x} w_{xy}}.
\]

(22)

A caveat to this point is that not all the entries in \( W \) contribute to the stability (instability) of the network dynamics. Only the components of \( W \) that correspond to active neurons in \( v \) at any time step are significant to the dynamics. For example if a candidate tour solution at a time step for a 5-city TSP is \( 1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 1 \), this means neurons in \( v \) that correspond to \( v_{1,1}, v_{3,2}, v_{2,3}, v_{5,4}, v_{4,5} \) are active (of which the non-zero magnitudes are unknown but positive). Hence, only the entries of \( W \) that correspond to \( (x, y) \) values of \( (1, 3), (3, 2), (2, 5), (5, 4), (4, 1) \) as well as \( (3, 1), (2, 3), (5, 2), (4, 5), (1, 4) \), and because of symmetry, contribute to the stability (instability) of the network dynamics. Thus, to further extend Eq. (21),

\[
\sum_{y \neq x} \sigma(x, y) w_{xy} < 1,
\]

(23)

where \( \sigma(x, y) = 1 \) if a path exists between city \( x \) and \( y \) and \( \sigma(x, y) = 0 \) otherwise. The issue of stability and convergence will be a focal point for future work.

5.2. Constraints

The problem constraints, mapped onto the recurrent LT network manifests itself as constraints upon the dynamics of the network, usually in the form of inhibition. These embedded constraints in the proposed dynamics pose an interesting problem. A trade-off exists between the selection of a small inhibitory term with that of a large inhibitory term. If the constraints, mapped as an inhibitory term in the neural network implementation are excessively large, an optimum solution is less likely to be found, as previously activated neurons are more likely to inhibit the activation of later neurons. Intuitively, this means that the search space, or candidate sets of neurons is limited and that the possible candidates are less diverse. Conversely, too small an inhibition would result in the network perpetually switching between active and inactive neurons,
since the support a neuron receives from previously
activated neurons may be greater than the inhibition it
receives from active neurons in the same row or column.
A corollary of this is that convergence of the network
dynamics will be much slower as neuron activity is less
likely to settle to a steady state. Invalid tour solutions
would then result.

5.3. Network parameters

In the proposed LT approach, the 4-network parameters
\((k_0, k_1, k_2, k_3)\) can be modified to alter the relative
contribution of each term \((\xi_0, \xi_1, \xi_2, \xi_3, \) respectively) to
the state update dynamics of the LT network. These
parameters are critical in determining the stability,
feasibility and optimality of the network dynamics,
particularly because the architecture is based on a recurrent
structure, which, together with the use of unbounded LT
activation function can easily lead to unstable dynamics.

Under most circumstances, \(k_0\) is modified when the
distances of the cities are not normalized (i.e. the Cartesian
\((x, y)\) coordinates of the cities are not within the unit box
with range \([0, 1]\)). This is to prevent the divergence of
network activity since \(\xi_0\) is always non-positive. \(k_1\) and \(k_2\)
controls the magnitude of inhibition of the rows and
columns of neurons. We believe that prior normalization of
the weight matrix \(W\) would aid in scalability of the problem
particularly in determining appropriate values of the
network parameters. As mentioned in the previous
sections, a greater inhibitory value restricts the search
space and the number of possible candidate solutions in the
network dynamics. Lastly, \(k_3\) accounts for the magnitude
of the penalty term whenever the sum of active neurons is
less than \(n\) (positive penalty) or more than \(n\) (negative
penalty).

These parameter values \((k_0, k_1, k_2, k_3)\) influences the size
of the search space that is being considered by the
dynamics of the LT network. Too strict values will lead
to a small search space being considered and is heavily
dependent on the initial conditions of the network due to
premature convergence, while values which are too relaxed
will often lead to non-convergence of the dynamics. Proper
setting of these values (the use of stochastic optimization
approaches such as GA, as was outlined in Section 3) will
enable better results to be obtained.

5.4. Conditions for convergence

In a similar manner, the conditions for convergence are
also of considerable significance. A requirement that is too
strict would lead to long convergence times, while on the
other hand, convergence conditions that are not sufficiently
strong would result in solutions that are not particularly
good (longer distances are obtained), although the
corresponding convergence times might be favorable.
Worse yet, invalid tour solutions occur more frequently
when weaker conditions for stopping are used. This can be
interpreted as the search space being stuck in a local
minimum.

It is for this reason that analog neurons are used instead
of binary neurons, as the lack of granularity in a binary
threshold neuron does not provide a good basis for
intermediary computation (since only one of the two states
are stored). The finer resolution in an analog neuron in this
sense allows greater versatility in calculation of transitory
state values prior to convergence. The use of an LT neuron,
which is not only analog-valued but also non-saturating,
allows a greater range of values to be represented, without
having to be overly concerned with the possibility of
intermediate state values saturating. As long as the LT
network is stable (using the previously mentioned sufficient
conditions), the dynamics are guaranteed to converge.

In the simulations that were carried out, unless otherwise
specified, the condition requirements were that there is no
change in the energy function (see next section) for
successive iterations, i.e. \(\Delta E = 0\) for at least \(k\) time steps.
Alternatively, other stopping conditions can be one that
requires that there be no change in either (i) the analog
values of the neurons (i.e. the activations of the neurons
remain steady or constant), or (ii) the states of the neurons
(i.e. active neurons remain active, inactive neurons remain
likewise inactive).

Condition (i) is equivalent to requiring that \(\Delta e_{x,y} = 0\)
\(\forall x, y; i \in n \Rightarrow \sum_x \sum_y (v_{x,y}^{new} - v_{x,y}^{old})^2 = 0\) or
\(\sum_x \sum_y |v_{x,y}^{new} - v_{x,y}^{old}| = 0\). Condition (ii), on the other hand, is a less strict
criteria, only requiring that the set of active neurons remain
active, and the set of inactive neurons remain inactive for a
certain number \((k)\) of time steps, i.e. \(\sum_x \sum_y (\delta(v_{x,y}^{new}) -
\delta(v_{x,y}^{old})) = 0\) for \((k)\) time steps. Again, \(\delta(z) = 1\) for \(z > 0\)
and \(\delta(z) = 0\) otherwise.

Fig. 12 shows a typical Pareto front for the solutions (tour
distances), as a trade-off between convergence time (a strict
stopping criteria takes a longer time to converge; conversely,
a more relaxed stopping criteria would converge more
rapidly) and the tour distances obtained. The Pareto front in
this case is based on the 10-city TSP where the optimal tour
distance is 2.59 units. The solutions are represented by the
circles lying on a curved line (since two competing objectives
are being examined here). In an ideal situation, we would
like to minimize both these objectives, that is the tour
distance that is obtained, as well as the time required to
arrive at the solution. A particular solution is a member of
the Pareto front if there are no other solutions that are better
than this solution both in tour distance and time taken.
These are known as the \textit{non-dominated} solutions. On the
other hand, solutions that do not lie on this Pareto front
(not shown) are the \textit{dominated} solutions.

5.5. Open problems

In combinatorial optimization, or for any problems in
general optimization theory, LT networks are an interesting
possibility particularly because of its non-saturating
activation function which allows greater competition and hence selectivity to be achieved as compared to archetypal classes of neurons with discrete or bounded activation functions. The study into identifying appropriate parameters when mapping combinatorial optimization problems onto neural networks of LT neurons warrants further investigation. Better insight into the role played by these parameters would be expected to result in better quality solutions. The determination of the network parameters, can be analogously compared to as determining a balance between the size of the search space and restricting the search within the confines of the problem constraints. The use of a GA that was used here provide a more appropriate setting of the network parameters \((k_0, k_1, k_2, k_3)\) that was shown to obtain better results. This issue of stability and convergence of the energy dynamics will be a focal point for future work. Aside from the above parameters, the solution approach also involves the prior determination of various settings, such as convergence conditions, network weights; all of which are less explicit but nonetheless critical in the set of solutions that are obtained using the proposed approach.

6. Conclusions

This paper has presented an alternative formulation to solving the TSP, where a recurrent network of non-saturating linear threshold (LT) neurons has been used, instead of the classical Hopfield model. Results that were obtained from utilizing the proposed network of LT neurons using asynchronous update dynamics consistently produced better results than its Hopfield equivalent in solving certain instances of the Traveling Salesman Problem (TSP). Particularly important is that the worst (maximum) distances obtained for the TSP using the proposed LT network is significantly better (lesser in distance) than the Hopfield network (see Tables 3 and 4). The improved results are believed to be attributed to the ability of the LT neurons to avoid saturating levels, characteristic of many archetypical neuronal activating functions. Larger-size problems were much slower to converge because of the difficulty in determining the appropriate parameters, which in turn is due to the exponentially greater number of iterations and candidate solutions, and hence were not quantitatively pursued. However, the dynamics for higher dimensional problems exhibit qualitatively similar dynamics.

Clearly, heuristical and mathematical approaches using a priori information would give rise to better performance and scalability. Notwithstanding, the objective of this article is not to demonstrate improved results for the TSP, but rather, to put forward the viability of using neurons with non-saturating properties, specifically of the LT type, as a candidate activation function for solving such combinatorial optimization problems using Hopfield-like recurrent networks. We believe that further fine-tuning of the architecture and parameter settings would increase the practicability of such networks in solving a more general class of optimization problems.

References

Huajin Tang received his B.S. degree in engineering from Zhe Jiang University, in 1998, his M.S. degree in engineering from Shanghai Jiao Tong University, in 2001, and his Ph.D. degree in electrical and computer engineering from National University of Singapore, Singapore, in 2005. His research interests include neural networks, neuro-dynamics, intelligent computation, and optimization. He has coauthored a number of journal and conference papers, as well as one monograph.

K.C. Tan received his B. Eng. degree with first class honors in electronics and electrical engineering and his Ph.D. degree from the University of Glasgow, Glasgow, Scotland, in 1994 and 1997, respectively. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, National University of Singapore. He has authored or coauthored 6 books and more than 140 journal and conference publications. He is an international program committee member for over 50 conferences and served in the organizing committee for over 15 international conferences. He currently serves as an Associate Editor/Editorial Board member of IEEE Transactions on Evolutionary Computation, Journal of Scheduling, European Journal of Operational Research, and International Journal of Systems Science. His current research interests include computational intelligence, evolutionary computation, multiobjective optimization, and engineering design optimization.

C. Xiang received his B.S. degree in mechanical engineering from Fudan University, China, in 1991, his M.S. degree in mechanical engineering from the Institute of Mechanics, Chinese Academy of Sciences, in 1994, and his M.S. and Ph.D. degrees in electrical engineering from Yale University, New Haven, CT, in 1995 and 2000, respectively. From 2000 to 2001, he was a Financial Engineer at Fannie Mae, Washington, DC. Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering, National University of Singapore. His research interests include computational intelligence, adaptive systems, and pattern recognition.

C.K. Goh received his B.Eng. degree in electrical engineering from the National University of Singapore, in 2003. He is currently working toward his Ph.D. degree at the Centre for Intelligent Control, National University of Singapore on a research and graduate fellowship. His current research interests include evolutionary computation and neural networks, specifically in the application of evolutionary techniques for multiobjective optimization.

E.J. Teoh received his B.Eng. degree in electrical engineering from the National University of Singapore, in 2004, together with a minor in management of information technology, with first-class honors. Since then, he has been working toward a Ph.D. degree in electrical engineering at the same university on a research scholarship and graduate fellowship at the Centre for Intelligent Control. His research interests include, but are not limited to neural networks, neural and evolutionary computation and their applications. He plays the classical guitar and cello, in addition to building them, as his other interests.