A Recurrent Neural Network for Solving the Shortest Path Problem

Jun Wang
Department of Industrial Technology
University of North Dakota
Grand Forks, ND 58202-7118
USA

***Abstract*— The shortest path problem is the classical combinatorial optimization problem arising in numerous planning and designing contexts. In this paper, a recurrent neural network for solving the shortest path problem is presented. The proposed recurrent neural network is able to generate optimal solutions to the shortest path problem. The performance and operating characteristics of the recurrent neural network are demonstrated by use of illustrative examples.**

I. Introduction

The shortest path problem is concerned with finding the shortest path from a specified starting node (origin) to a specified ending node (destination) in a given network while minimizing the total cost associated with the path. The shortest path problem is a classical combinatorial optimization problem having widespread applications in a variety of settings. The applications of the shortest path problem include vehicle routing in transportation systems [1], traffic routing in telecommunication networks [2,3,4], and path planning in robotic systems [5]. Furthermore, the shortest path problem also has numerous variations such as the minimum weight problem, the quickest path problem, the most reliable path problem, etc.

Since Hopfield and Tank's pioneering work [6,7], neural networks for solving optimization and combinatorics problems have been a major topic in neural network research. Although there have been very few direct attacks of the shortest path problem using neural network [8-10], neural network models for solving the related problems such as the traveling salesman problem have been investigated extensively in recent literature. These investigations have shed light on the neural network approach to the shortest path problem.

In the present paper, a recurrent neural network for solving the shortest path problem is proposed. The proposed recurrent neural network is demonstrated capable of generating the shortest path for a network with mixed positive and negative cost coefficients. The proposed neural network possesses many regularity properties and is suitable for VLSI implementation.

II. Path Representation

Given a direct graph $G = (N, A)$ where N is a set of n nodes (vertices) and A is an ordered set of m arcs (edges), $m \leq n^2$. A fixed cost c_{ij} is associated with each arc a_{ij} in the graph G. In transportation and robotic systems, for example, the physical meaning of the cost can be the distance between the nodes, the time or energy needed for travel from one node to another. In telecommunication systems, the cost can be determined according to the transmission time and the link capacity from one node to another. In general, the cost coefficients matrix $[c_{ij}]$ is not necessarily symmetric; i.e., the cost from node i to node j may not be equal to the cost from node j to node i. Furthermore, the arcs between some nodes may not exist; i.e., m may be less than n^2. The values of cost coefficients for the $n^2 - m$ nonexistent arcs are defined as infinity. More generally, a cost coefficient can be either positive or negative. A positive cost coefficient represents a loss, whereas a negative one represents a gain. For an obvious reason, we assume that there are neither negative cycles nor negative loops in the networks (i.e., no cycle with negative total cost and no loop with negative cost). Hence the total cost of the shortest path is bounded from below.

A path in a given network can be represented in different ways and the way of path representation in turn affects the effectiveness and efficiency of a solution procedure. An edge path representation uses an $n \times n$ binary matrix to represent the edges (segments) in a path. Specifically, the binary matrix of edge path representation $V = [v_{ij}]$ also contains only '0' and '1' elements, and v_{ij} can be defined as

$$v_{ij} = \begin{cases} 1 & \text{if the arc } a_{ij} \text{ is in the path;} \\ 0 & \text{otherwise;} \end{cases} \quad (1)$$

In the edge path representation, each row and each column can contain no more than one '1' element, if each node can be visited at most once. Since the cost coefficients of loops are assumed to be nonnegative (i.e., $c_{ii} \geq 0$ for $i = 1, 2, \ldots n$), the elements in the main diagonal line of the edge path representation are always zero. Since the diagonal elements are always zero, the edge path representation can be simplified by excluding the diagonal elements v_{ii} $(i = 1, 2, \ldots, n)$. Consequently,

the simplified edge representation has only $n(n-1)$ binary elements.

III. Problem Formulation

The shortest path problem is to find the shortest (least costly) possible directed path from a specified starting node (origin) to a specified ending node (destination). The cost of the path is the sum of the costs on the arcs in the path.

Based on the edge path representation, the shortest path problem can be formulated as a linear integer programming problem as follows [11]:

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{j\neq i} c_{ij}v_{ij}; \tag{2}$$

$$\text{subject to} \quad \sum_{k=1,k\neq i}^{n} v_{ik} - \sum_{l=1,l\neq i}^{n} v_{li} = \begin{cases} 1, & \text{if } i = s, \\ 0 & \text{if } i \neq s \,\&\, i \neq e, \\ -1, & \text{if } i = e, \end{cases} \tag{3}$$

$$v_{ij} \in \{0,1\},\ i \neq j; i,j = 1,2,\ldots,n; \tag{4}$$

where v_{ij} denotes the decision variable associated with the arc from node i to node j.

In this shortest path problem formulation, the number of decision variables is $n(n-1)$. The arc from node i to node j is in the shortest path if decision variable $v_{ij} = 1$. The objective function to be minimized, eqn. (2), is also the total cost for the path. The equality constraint coefficients and the right-hand sides are -1, 0, or 1. The first constraint, eqn. (3), ensures that a continuous path starts from a specified origin and ends at a specified destination. The second constraint, eqn. (4), is the integrality constraint.

Because of the total unimodulity property of the constraint coefficient matrix defined in eqn. (3) [11], the interality constraint in the shortest path problem formulation can be equivalently replaced with the nonnegativity constraint, if the shortest path is unique. In other words, the optimal solutions of the equivalent linear programming problem are composed of zero and one integers if a unique optimum exists [11]. The equivalent linear programming problem can be described as follows.

$$\text{minimize} \quad \sum_{i=1}^{n}\sum_{j\neq i} c_{ij}v_{ij}; \tag{5}$$

$$\text{subject to} \quad \sum_{k\neq i} v_{ik} - \sum_{l\neq i} v_{li} = \delta_{is} - \delta_{ie},\ \forall i; \tag{6}$$

$$v_{ij} \geq 0, \quad \forall i \neq j; \tag{7}$$

where δ_{pq} is the Kronecker delta function defined as $\delta_{pq} = 1$ if $p = q$ and $\delta_{pq} = 0$ if $p \neq q$.

IV. Dynamical Equation

Because the formulated shortest path problem is a special form of linear programming problems, the shortest path problem can be solved by the neural networks proposed for solving linear programming problems in [7,12-14]. In [13], a recurrent neural network is proposed and demonstrated to be capable of solving linear programming problems. The proposed recurrent neural network for solving the shortest path problem is tailored from the recurrent neural network [13].

Let the decision variables of the shortest path problem be represented by the activation states, the neural network for solving the shortest path problem consists of $n(n-1)$ neurons arranged spatially in an $n \times n$ array without diagonal elements. For simplicity of notations, the same symbol $V = [v_{ij}]$ is used to denote both the decision variables and activation states.

An energy function can be defined as follows:

$$E[t,v(t)] = \beta\sum_{i=1}^{n}\sum_{j\neq i} c_{ij}\exp(-t/\tau)v_{ij}(t) + \frac{w}{2}\sum_{i=1}^{n}\left[\sum_{k\neq i} v_{ik}(t) - \sum_{l\neq i} v_{li}(t) - \delta_{is} + \delta_{ie}\right]^2, \tag{8}$$

where β, w and τ are positive scaling constants. The role of $\exp(-t/\tau)$ in eqn. (8) is explained in [13,14].

Let $du_{ij}(t)/dt = -\partial E[t,v(t)]/\partial v_{ij}$, the state dynamics of the recurrent neural network can be described as follows: For $i \neq j; i,j = 1,2,\ldots,n$;

$$\frac{du_{ij}(t)}{dt} = -w\sum_{k\neq i} v_{ik}(t) + w\sum_{l\neq i} v_{li}(t) + w\sum_{p\neq j} v_{jp}(t) - w\sum_{q\neq j} v_{qj}(t) + w(\delta_{is} - \delta_{ie} - \delta_{js} + \delta_{je}) - \beta c_{ij}\exp(-t/\tau); \tag{9}$$

$$v_{ij}(t) = \frac{v_{\max}}{1+\exp(-\xi u_{ij}(t))}; \tag{10}$$

where $f_{ij}(\cdot)$ is a nonnegative activation function.

The first four terms in the right-hand side of eqn. (9) define the connectivity of the recurrent neural network. The fifth and sixth terms in the right-hand side of eqn. (9) define the constant and decaying thresholds (biases), respectively. Let the neuron in the jth row and ith column be termed as the mirror neuron of the neuron in the ith row and jth column. Eqn. (9) shows that (i) inhibitory connections exist between neuron $v_{ij}(t)$ and every neuron in the present row and column including itself, $v_{ik}(t)$ $(k \neq i)$ and $v_{qj}(t)$ $(q \neq j)$; (ii) excitatory connections exist between neuron $v_{ij}(t)$ and every neuron in the same row and column of its mirror neuron $v_{ji}(t)$, $v_{jp}(t)$ and $v_{li}(t)$ $(p \neq j, l \neq i)$; (iii) all self-feedback connection weights are equal to $-2w$; (iv) all

connection weights between mirror neurons are $2w$; (v) the absolute values of all other connection weights are w. That is, among the $n(n-1)(2n-3)$ connection links, there are $n(n-1)(2n-3)/2$ inhibitory connections and $n(n-1)(2n-3)/2$ excitatory connections for a nontrivial case ($n \geq 2$). Eqn. (9) also shows that (i) the constant biasing thresholds unrelated to starting or ending nodes are zero, (ii) the constant biases in the sth row and eth column are positive, and (iii) the constant biases in the eth row and sth column are negative. Specifically, the constant bias of neuron $v_{se}(t)$ is $2w$, the constant bias of neuron $v_{es}(t)$ is $-2w$, the constant biases of neurons $v_{sj}(t)$ and $v_{ie}(t)$ ($j \neq e, i \neq s$) are w, and the constant biases of neurons $v_{sj}(t)$ and v_{is} ($j \neq s, i \neq e$) are $-w$.

One salient advantage of the proposed recurrent neural network is the independency of the connection weight matrix upon specific problems. Specifically, only the constant biases are different for different origins and/or destinations of the same network, and only the initial values of the decaying biases are different for different networks with the same number of nodes, and the same origin and destination. By biasing different nodes, the proposed recurrent neural network can be used to generate all-pair shortest paths. Furthermore, the recurrent neural network can be modulized with a large number of neurons. In specific applications, the unused neurons can be disabled by assigning very large cost coefficients to penalize the selection of the arcs. These desirable features facilitate the VLSI implementation of the proposed recurrent neural network.

Because the average convergence rate of the recurrent neural network is nondecreasing as the network size increases and the convergence rate can be expedited by properly selecting design parameters τ, β, and $w_{,,}$ the proposed recurrent neural network has $O(1)$ time complexity. The spatial complexity of the recurrent neural network is characterized by $O(n^2)$ neurons and $O(n^3)$ connections.

V. Simulation Results

Example 1: Consider a shortest path problem with n being ten. Without loss of generality, let's assume that the starting and ending nodes are nodes 1 and 10, respectively. Figure 1 depicts the network topology which is generated randomly in the unit square, where the solid lines indicate the shortest path and the dashed lines indicate the existing arcs. Euclidean distances are used as the cost coefficients. The shortest path of this problem is $\{n_1, n_2, n_3, n_{10}\}$; i.e., $\{a_{12}, a_{23}, a_{3,10}\}$. The total cost of the shortest path is 1.149896. Note that the second shortest path, $\{n_1, n_2, n_4, n_{10}\}$, has a very close total cost of 1.172729. Let $w = \beta = 10^7, \tau \approx 10^{-6}$, $\xi = 10, v_{\max} = 10$. Figure 2 depicts the transients of the first three rows in the state matrix of the recurrent neural network simulated using PSpice, where $V(ij)$ denotes

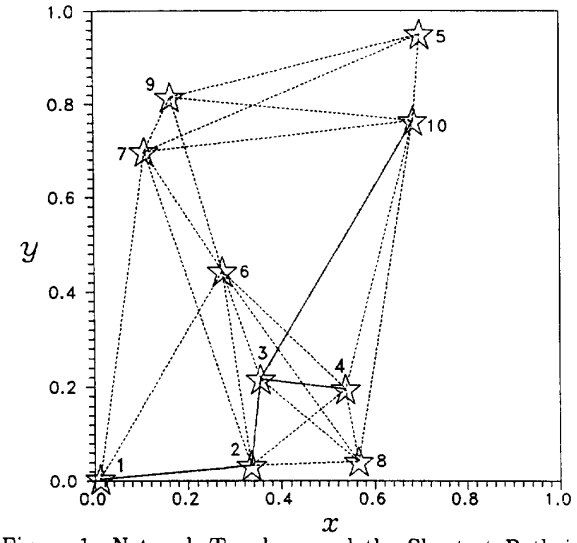

Figure 1: Network Topology and the Shortest Path in Example 1.

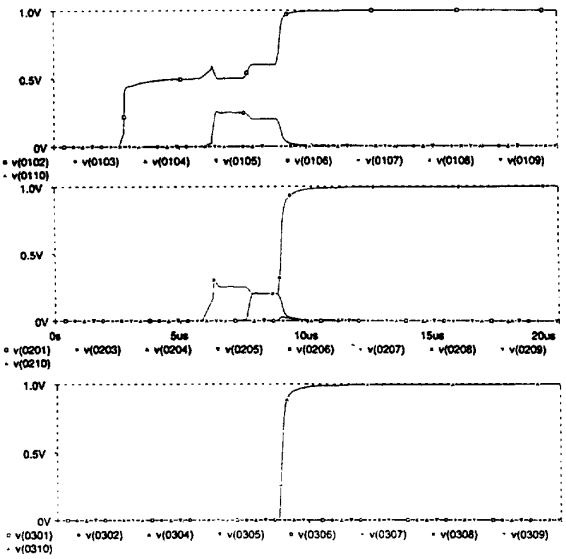

Figure 2: Transient States of the Simulated Recurrent Neural Network in Example 1.

v_{ij} ($i, j = 1, 2, \ldots, n$). Obviously, the neural network solution to the problem represents the shortest path. The simulated recurrent neural network takes about 10 microseconds to converge. The simulation result in this example also indicates that the proposed recurrent neural network can distinguish the shortest path and the very close second shortest path.

Example 2: Consider solving different sizes of shortest path problems. One hundred random cost coefficient matrices of each size are generated uniformly on

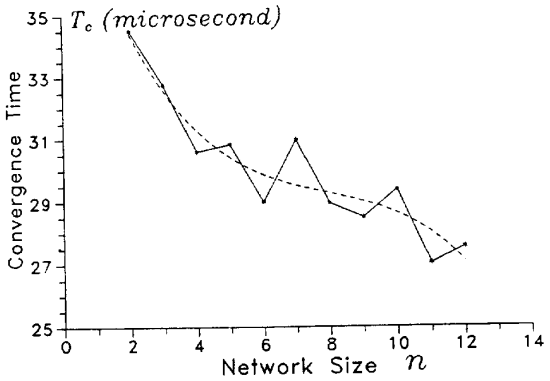

Figure 3: Network Size versus Average Convergence Time in Example 2.

[0,1]. Let $w = \beta = 10^5, \xi = 10$ and $\tau = 0.1ms$. Using a simulator, the simulation results show that the steady states of the simulated neural network always give rise to correct solutions. Figure 6 illustrates the network size (n) versus the average convergence time of the recurrent neural network (T_c) in microseconds, where the solid line connects the data, the dashed line represents a polynomially fitted curve based on the data. T_c is determined using the average time interval from the beginning to all the activation states entering and staying within $\pm 1\mu V$ range of the steady state $\bar{v}$; i.e., $T_c = 0.01 \sum_{p=1}^{100} \min\{t | \lim_{t\to\infty} v_{ij}^{(p)}(t) - 1\mu V \leq v_{ij}^{(p)}(t) \leq \lim_{t\to\infty} v_{ij}^{(p)}(t) + 1\mu V\}$, where $p = 1, 2, \ldots, 100$, representing ten samples for each size of directed network. It shows clearly that the average convergence time does not increase as the network size increases. This implies that large-scale problems will make the spatial complexity of the recurrent neural network increase, but not the temporal complexity from a statistical point of view. Therefore, it suffices to demonstrate the performance of the proposed neural network using small-scale examples.

VI. Conclusions

In this paper, a recurrent neural network for solving the shortest path problem has been proposed. The dynamics and architecture of the recurrent neural network have been described. It has been shown that the proposed recurrent neural network is capable of determining the shortest paths of a given directed network. Since the solution process is inherently parallel and distributed, the convergence rate is nondecreasing with respect to the size of the shortest path problem. Furthermore, the convergence rate of the neural network can be expedited by properly selecting design parameters. These features make the proposed recurrent neural network suitable for solving large-scale shortest path problems in real-time applications.

References

1. L. Bodin, B. L. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: The state of the art," *Computers and Operations Research*, vol. 10, no. 2, pp.'63-211, 1983.
2. A. Ephremides and S. Verdu, "Control and optimization methods in communication network problems," *IEEE Trans. on Automatic Control*, vol. 34, no. 9, pp. 930-942, 1989.
3. D. M. Topkis, "A k shortest path algorithm for adaptive routing in communication networks," *IEEE Trans. on Communications*, vol. 36, no. 7, pp. 855-859, 1988.
4. J. K. Antonio, G. M. Huang, W. K. Tsai, "A fast distributed shortest path algorithm for a class of hierarchically clustered data networks," *IEEE Trans. on Computers*, vol. 41, no. 6, 710-724, 1992.
5. S. Jun and K. G. Shin, "Shortest path planning in distributed workspace using dominance relation," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 3, pp. 342-350, 1991.
6. J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141-152, 1985.
7. D. W. Tank and J. J. Hopfield, "Simple neural optimization networks, an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. on Circuits and Systems*, vol. 33, no. 5, pp. 533-541, 1986.
8. H. E. Rauch and T. Winarske, "Neural networks for routing communication traffic," *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 26-30, 1988.
9. L. Zhang and S. C. A. Thomopoulos, "Neural network implementation of the shortest path algorithm for traffic routing in communication networks," *Proc. of Intl. Joint Conf. on Neural Networks*, vol. II, p. 591, 1989.
10. G. Fahner, "An algorithm-structured neural net for the shortest path problem," *Proc. of Intl. Joint Conf. on Neural Networks*, vol. I, pp. 153-158, 1991.
11. M. S. Bazaraa and J. J. Jarvis, *Linear Programming and Network Flows*, pp. 483-492, John Wiley & Sons, New York, NY, 1977.
12. J. Wang, and Chankong, V., "Recurrent neural networks for linear programming: Analysis and design principles," *Computers and Operations Research*, vol. 19, no. 3/4, pp. 297-311, 1992.
13. J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 9, pp. 613-618, 1993.
14. J. Wang, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, in press, 1994.