$V_{\min}$, D_k becomes forward biased and closes the feedback loop. Y_k will stay at $SITE_k - V_{DF}$, where V_{DF} is the forward drop of D_k. As a result $V_{\min} = \min_k(SITE_k)$. With SW1 open, the *CE* output V_{out} is given by the initialization voltage V_{init}.

When switch SW1 closes, similar argument applies, and $V_{out} = V_{\min} = \min_k(SITE_k, V_{init})$.

3) *The Network*: With the circuit for the sites and units defined, a network is constructed according to Fig. 1. Since the minimum spanning tree problem is defined on undirected graphs, the number of required *CE*'s is $N(N-1)/2$ for an N-node graph.

Fig. 5 shows the simulated response of the network, which corresponds to the problem defined on the graph in Fig. 4.

With START driven active, the output of *CE*'s that represent arcs in A are initialized to the corresponding values. Output of the other *CE*'s, which represent initially disconnected arcs, are initialized to an arbitrary value (1.7 V in Fig. 5) that is larger than all the known values. After initialization, the network begins converging to the solution by driving START inactive at $t = 1$ μs. When the network settles at $t \approx 6.5$ μs, *CE*'s having output values the same as the initial value indicate that the corresponding arcs are in the minimum spanning tree. If there are more than one minimum spanning tree in the graph, all *CE*'s corresponding to arcs in the minimum spanning trees will have the output value the same as the initial value. The settling time of the network is determined by the large signal behavior of the op-amps. The op-amps in the maximizer circuit should not be allowed to go into saturation, otherwise erratic behavior may occur. A faster op-amp is often needed in the maximizer circuit. Fig. 6 shows a correct network response when a fast op-amp is used in the maximizer circuits.

V. Conclusion

The closed semiring structure provides a systematic way in formulating a problem to be solved on the binary relation inference network. The extension operator $\oplus$ in a closed semiring corresponds to the site function while the summary operator $\circledcirc$ corresponds to the unit function. As an example and a further application of the binary relation inference network, the minimum spanning tree problem is solved with a closed semiring structure on the inference network architecture.

A specific implementation of the inference network using basic max/min analog circuit blocks is described, and simulation results show that the convergence time of the network is in the range of microseconds.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* Cambridge, MA: The MIT Press, 1990.

[2] R. W. Floyd, "Algorithm 97—shortest path," *Commun. ACM,* vol. 5, p. 345, 1962.

[3] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik 1,* pp. 269–271, 1959.

[4] G. Rote, "A systolic array algorithm for the algebraic path problem (shortest paths; matrix inversion)," *Computing,* vol. 34, pp. 191–219.

[5] T. Takaoka and K. Umehara, "An efficient VLSI algorithm for the all pairs shortest path problem," *J. Parallel Distributed Computing,* vol. 16, pp. 265–270, 1992.

[6] Y. Robert and D. Trystram, "Systolic solution of the algebraic path problem," in *Systolic Arrays: Paper Presented at the First International Workshop on Systolic Arrays,* W. Moore, A. McCabe, and R. Urquhart, Eds. Oxford: Adam Hilger, July 2–4, 1986, pp. 171–180.

[7] F. J. Núñez and M. Valero, "A block algorithm for the algebraic path problem and its execution on a systolic array," in *Int. Conf. Systolic Arrays: Proc.,* May 25–27, 1988, pp. 265–274.

[8] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms.* Reading, MA: Addison–Wesley, 1974.

[9] K. P. Lam and C. J. Su, "On a binary relation inference network," in *Proc. IEEE 5th Int. Parallel Processing Symp.,* 1991, pp. 250–255.

[10] C. J. Su, "A binary relation inference network for constrained optimization," Ph.D. dissertation, Univ. British Columbia, 1992.

[11] C. W. Tong and K. P. Lam, "Analog implementations of a binary relation inference network for minimum cost path problems," in *Proc. IEEE Int. Conf. Neural Networks,* San Francisco, CA, 1993, pp. 1081–1085.

[12] J. B. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem," *Proc. Amer. Math. Soc.,* vol. 7, pp. 48–50, 1956.

[13] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.,* vol. 36, pp. 1389–1401, 1957.

[14] K. P. Lam, "A continuous-time inference network for minimum cost path problems," in *Proc. IEEE/INNS Int. Joint Conf. Neural Networks,* Seattle, WA, 1991, vol. 1, pp. 367–372.

[15] C. W. Tong and K. P. Lam, "VLSI implementation of binary relation inference network in solving shortest path problems," in *Proc. IEEE Int. Conf. Neural Networks,* New Orleans, LA, 1994, pp. 2143–2148.

A Recurrent Neural Network for Solving the Shortest Path Problem

Jun Wang

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 recurrent neural network is able to generate optimal solutions to the shortest path problem. The performance of the recurrent neural network is demonstrated by means of three illustrative examples. The recurrent neural network is shown to be capable of generating the shortest path and suitable for electronic implementation.**

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 communication networks [2]–[4], and path planning in robotic systems [5].

The shortest path problem has been investigated extensively. The well-known algorithms for solving the shortest path problem include the $O(n^2)$ Bellman's dynamic programming algorithm for directed

Manuscript received November 15, 1993; revised February 1, 1995. This paper was recommended by Associate Editor C. Jutten.

The author is with the Department of Industrial Technology, University of North Dakota, Grand Forks, ND 58202 USA and the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong.

Publisher Item Identifier S 1057-7122(96)03903-7.

acycle networks, the $O(n^2)$ Dijkstra-like labeling algorithm for networks with only nonnegative cost coefficients, and the $O(n^3)$ Bellman-Ford successive approximation algorithm for networks without negative cost coefficients, where n denotes the number of nodes in the network. For large-scale and real-time applications such as traffic routing and path planning, the existing series algorithms may not be effective and efficient due to the limitation of sequential processing in computational time. Therefore, parallel solution methods are more desirable.

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 a very few direct attacks of the shortest path problem using neural network [8], [9], 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 brief, a recurrent neural network for solving the shortest path problem is proposed. The recurrent neural network is demonstrated capable of generating the shortest path for a network with mixed positive and negative cost coefficients. The neural network possesses many regularity properties and is suitable for circuit implementation.

II. Problem Formulation

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 coefficient 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. 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. It is admittedly more difficult to find the shortest path for a network with mixed positive and negative cost coefficients.

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

$$v_{ij} = \begin{cases} 1 & \text{if the arc from node } i \text{ to node } j \text{ is in the path;} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Each row and each column in the edge representation 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 $i = 1, 2, \cdots n$), 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} $(i = 1, 2, \cdots, n)$. Consequently, the simplified edge representation has only $n(n-1)$ binary elements. Because the edge path representation can represent the shortest path with more than n nodes in a general network with mixed positive and negative cost coefficients, it is desirable.

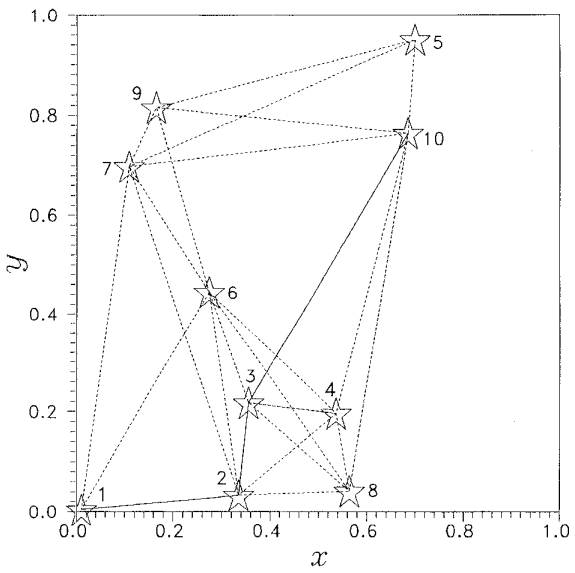

Fig. 1. Network topology and the shortest path in Example 1.

Based on the edge path representation, the shortest path problem can be formulated as a linear integer programming problem minimizing the sum of the costs on the arcs in the path [10]

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

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

$$v_{ij} \in \{0, 1\}, \quad i \neq j, i, j = 1, 2, \cdots, n \quad (4)$$

where s and e denote the starting and ending node, respectively.

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 equality constraint coefficients and the right-hand sides (RHS's) are -1, 0, or 1. The first constraint, (3), ensures that a continuous path starts from a specified origin and ends at a specified destination. The second constraint, (4), is the integrality constraint.

Because of the total unimodulity property of the constraint coefficient matrix defined in (3) [10], the integrality 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 [10]. The integrality constraint of (4) can be equivalently replaced by

$$v_{ij} \geq 0, \quad i \neq j; \ i, j = 1, 2, \cdots, n. \quad (5)$$

If each arc can be visited no more than once, then the assumption of the absence of negative cycles and negative loops can be relaxed without incurring unbounded total costs. In this case, an upper bound of 1 for each decision variable v_{ij} can be added to prevent v_{ij} increases to infinity. Namely, (4) can be changed to

$$0 \leq v_{ij} \leq 1, \quad i \neq j; \ i, j = 1, 2, \cdots, n. \quad (6)$$

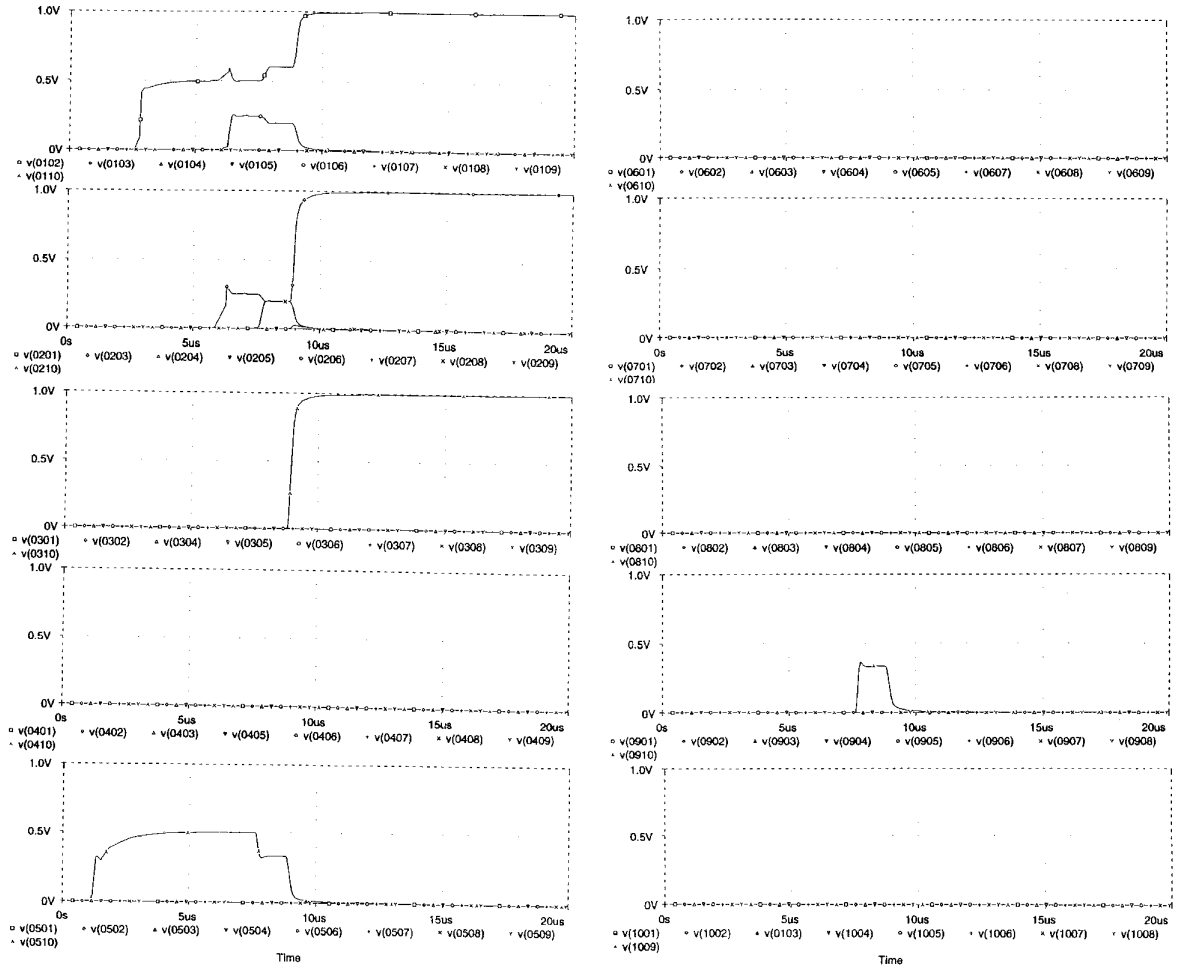

Fig. 2. Transient states of the simulated recurrent neural network in Example 1.

III. Dynamical Equation

Because the formulated shortest path problem is a linear programming problem, the shortest path problem can be solved by the neural networks for linear programming. In [11] and [12], a recurrent neural network called deterministic annealing neural network is presented 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 deterministic annealing neural network [11], [12].

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. An energy function can be defined as follows:

$$E[t, v(t)] = \beta \sum_{i=1}^{n} \sum_{j \neq i} c_{ij} \exp(-t/\tau) v_{ij}(t) + \frac{w}{2} \sum_{i=1}^{n} \left[\sum_{k \neq i} v_{ik}(t) - \sum_{l \neq i} v_{li}(t) - \delta_{is} + \delta_{ie} \right]^2 \quad (7)$$

where β, w and τ are positive scaling constants, δ_{pq} is the Kronecker delta function. The role of $\exp(-t/\tau)$ in (7) is explained in [11] and [12].

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 $i \neq j$; $i, j = 1, 2, \cdots, n$;

$$\frac{du_{ij}(t)}{dt} = -w \sum_{k \neq i} v_{ik}(t) + w \sum_{l \neq i} v_{li}(t) + w \sum_{p \neq j} v_{jp}(t) - w \sum_{q \neq j} v_{qj}(t) + w(\delta_{is} - \delta_{ie} - \delta_{js} + \delta_{je}) - \beta c_{ij} \exp(-t/\tau); \quad (8)$$

$$v_{ij}(t) = f_{ij}[u_{ij}(t)] \quad (9)$$

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

The first four terms in the RHS of (8) define the connectivity of the recurrent neural network. The fifth and sixth terms in the RHS of (8) 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. Equation (8) 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 $2w$; and v) the

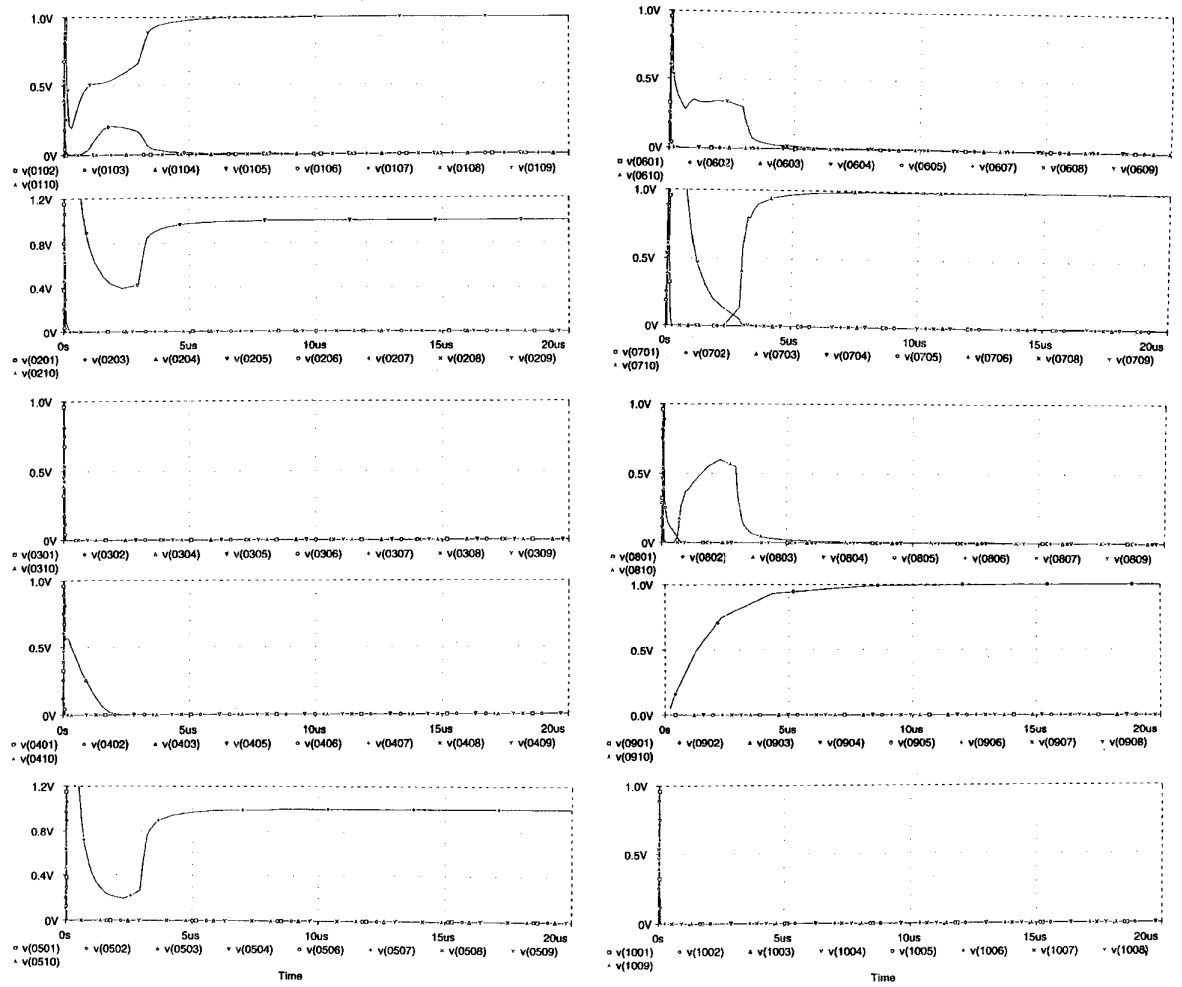

Fig. 3. Transient states of the simulated recurrent neural network in Example 2.

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 ($n \geq 2$). Equation (8) 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 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 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 recurrent neural network.

IV. Illustrative Examples

Example 1: Consider a shortest path problem with n being 10. Without loss of generality, let's assume that the starting and ending nodes are nodes 1 and 10, respectively. Fig. 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 1.149 896. Note that the second shortest path, $\{n_1, n_2, n_4, n_{10}\}$, has a very close total cost of 1.172 729. The optimal shortest path of this problem is given by the following binary matrix of edge representation.

$$V^* = [v_{ij}^*] = \begin{pmatrix} - & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & - & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - \end{pmatrix}.$$

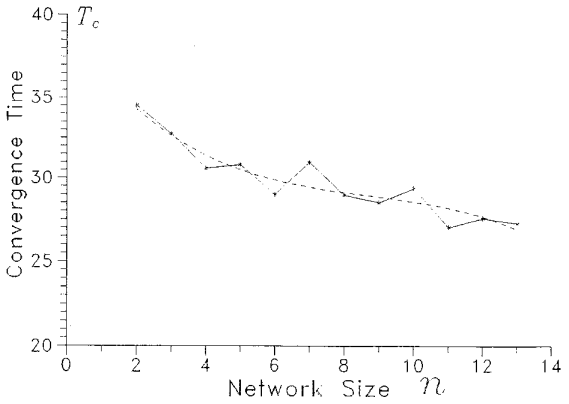

Fig. 4. Network size versus average convergence time in Example 3.

Fig. 2 depicts the transient behavior of the activation states of the recurrent neural network simulated using PSpice, where $V(ij)$ denotes v_{ij} $(i, j = 1, 2, \cdots, n)$, and i, j are in two digits. Obviously, the neural network solution to the problem represents the shortest path as shown in V^*. The simulated recurrent neural network takes about 10 μs to converge. The simulation result in this example also indicates that the recurrent neural network can distinguish the shortest path and the very close the second shortest path.

Unlike the popular Dijksdra's labeling algorithm which can solve the shortest path problem with nonnegative cost coefficients only, the recurrent neural network is capable of solving the shortest path problem with mixed positive and negative cost coefficients as will be shown in the following example.

Example 2: Consider a 10-node directed network with mixed positive and negative cost coefficients. The cost coefficient matrix is asymmetric and there are no loops or negative cycles in this network. The starting and ending nodes are assumed again as nodes 1 and 10, respectively. The shortest path of this problem is $\{n_1, n_9, n_2, n_5, n_7, n_{10}\}$; i.e., $\{a_{19}, a_{92}, a_{25}, a_{57}, a_{7,10}\}$, with the total cost of 0.360 952. The optimal shortest path of this problem is given by the following binary matrix of edge representation.

$$V^* = [v_{ij}^*] = \begin{pmatrix} - & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & - & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & - & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & - & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & - & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & - \end{pmatrix}.$$

Fig. 3 depicts the transient behavior of the activation states of the recurrent neural network simulated using PSpice based on the same design parameters in Example 1. Evidently, the neural network solution to the problem represents the shortest path as shown in V^*. It takes less than 10 μs for the simulated recurrent neural network to converge.

The average convergence rate of the recurrent neural network with the same design parameters normally does not decrease as the network size increases. The following example demonstrates this feature.

Example 3: Consider solving different sizes of shortest path problems. One hundred random cost coefficient matrices of each size are generated uniformly on [0, 1]. Let $w = \beta = 10^5$ and $\tau = 0.1$ ms. Using a C simulator based on second-order Runge-Kutta method, the simulation results show that the steady states of the simulated neural network always give rise to correct solutions. Fig. 4 illustrates the network size n versus the average convergence time of the recurrent neural network T_c in μs, 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 μV range of the steady-state $\bar{v}$. It shows clearly that the average convergence time does not increase as the network size increases statistically speaking.

The simulation results have shown that large-scale problems 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 neural network using these small-scale examples.

V. Conclusion

In this brief, a recurrent neural network for solving the shortest path problem has been presented. It has also been shown that the recurrent neural network is capable of determining the shortest paths of a given directed network with positive and/or negative cost coefficients. 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 recurrent neural network suitable for solving large-scale shortest path problems in real-time applications. The recurrent neural network implemented in a VLSI circuit can serve as a co-processor for onboard planning in dynamic decision environments.

References

[1] L. Bodin, B. L. Golden, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: The state of the art," *Comput. Operat. Res.*, vol. 10, no. 2, pp. 63–211, 1983.

[2] A. Ephremides and S. Verdu, "Control and optimization methods in communication network problems," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 930–942, Sept. 1989.

[3] D. M. Topkis, "A k shortest path algorithm for adaptive routing in communication networks," *IEEE Trans. Commun.*, vol. 36, pp. 855–859, July 1988.

[4] J. K. Antonio, G. M. Huang, and W. K. Tsai, "A fast distributed shortest path algorithm for a class of hierarchically clustered data networks," *IEEE Trans. Comput.*, vol. 41, pp. 710–724, June 1992.

[5] S. Jun and K. G. Shin, "Shortest path planning in distributed workspace using dominance relation," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 342–350, June 1991.

[6] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybern.*, 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. Circuits Syst.*, vol. 33, pp. 533–541, May 1986.

[8] H. E. Rauch and T. Winarske, "Neural networks for routing communication traffic," *IEEE Contr. Syst. Mag.*, vol. 8, no. 2, pp. 26–30, 1988.

[9] G. Fahner, "An algorithm-structured neural net for the shortest path problem," in *Proc. Int. Joint Conf. Neural Networks*, 1991, vol. I, pp. 153–158.

[10] M. S. Bazaraa and J. J. Jarvis, *Linear Programming and Network Flows*. New York: Wiley, 1977, pp. 483–492.

[11] J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 613–618, Sept. 1993.

[12] ——, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, no. 4, pp. 629–641, 1994.