Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 3, Issue 1 (June 2008), pp. 149 – 164
(Previously, Vol. 3, No. 1)

Neural network models for solving the maximum flow problem

S. Effati
Department of Mathematics
Tarbiat Moallem University
Sabzevar, Iran
effati@sttu.ac.ir

M. Ranjbar
Department of Mathematics
Payam Noor University (PNU)
Quchan, Iran
m.ranjbart@gmail.com

Abstract

In this paper, two new neural network models for solving the maximum flow problem are presented. The maximum flow problem in networks is formulated as a special type of linear programming problem and it is solved by appropriately defined neural networks. The nonlinear neural networks are able to generate optimal solution for maximum flow problem. We solve neural network models by one of the numerical method. Finally, some numerical examples are provided for the sake of illustration.

Key Word: Linear programming, neural network, maximum flow.

MCS: 90C05, 90C08, 90C35, 90C90 90B06, 92B20

1. Introduction

The maximum flow problem is one of the classic combinatorial optimization problems with much application such as electrical power, traffic communication and transportation computer network. This problem is one of the most fundamental problems with a wide variety of scientific and engineering application. The problem is to find a flow of maximum value on a network from a source to a sink, See Ahuja, et al. (1993, 1995). Maximum flow problem is a kind of linear programming problem. The linear programming problem was first solved by Dantzig with

simplex method sixty years ago. The simplex method developed by him, is still the most widely used numerical algorithm. Although the simplex method is efficient and elegant, but the modern numerical algorithms are very efficient and useful to solve maximum flow problem. The classical augmenting path method to find a maximum flow through a network was developed by Ford and Fulkerson, see Ford, et al. (1956). This adaptation of the simplex method for networks is 200 - 300 times faster than the simplex method applied to general linear programs of the same dimensions.

However they do not lend themselves to problems which require solution in real time. One promising approach to solve optimization problems in real time is to use the neural network approach. In the last fifty years, researchers have proposed various dynamic solutions for constrained optimization problems. This approach was first proposed by Pyne in 1956 and developed by Dennis Rybashov, Karpinskay and others. Several new dynamic solvers using artificial neural network models have been developed; see Hopfield, et al. (1985), Kennedy, et al. (1987), Xia, et al. (2002), Effati, et al. (2004). The numerical algorithms are also used like genetic algorithm to solve the network problems, see Leung, et al. (1998). Two neural network models for maximum flow problem are presented in this paper. They have a much faster convergence rate. These models are based on a nonlinear dynamical system.

2. Problem Formulation

Consider a network with m nodes and n arcs. We associate with each arc (i, j), a lower bound on flow of, $l_{ij} = 0$ and an upper bound on flow u_{ij}. We shall assume throughout the development that u_{ij}'s are finite integers. In such a network, we wish to find the maximum amount of flow from node 1 to node m. Let f represent the amount of flow in the network from node 1 to node m. Then the maximum flow problem may be stated as follows:

Maximize f

Subject to

$$\sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} f & if \quad i = 1 \\ 0 & if \quad i \neq 1 \ or \ m \\ -f & if \quad i = m \end{cases} \tag{1}$$

$$x_{ij} \leq u_{ij} \quad i = 1,2,..., m$$

$$x_{ij} \geq 0 \quad j = 1,2,..., m$$

Noting that f is a variable and denoting the node-arc incidence matrix by A, maximum flow problem can be presented in matrix form as:

Maximize cx

Subject to

$$\begin{aligned} &(e_m - e_1) f + Ax_1 = 0 \\ &x_1 \le U \\ &f, x_1 \ge 0 \end{aligned} \tag{2}$$

wherever, u_{ij}'s are components for $U \in R^n$ and containing network arc capacities. x_{ij}'s are components for $X_1 \in R^n$ and show flow on network arc. A is a matrix $m \times n$, and elements are the coefficients of x_{ij}'s. Above problem can be converted as follows:

Maximize cx

Subject to

$$\begin{aligned} &A_1 x = 0 \\ &A_2 x \le 0 \\ &x \ge 0 \end{aligned} \tag{3}$$

wherever, $c = [1,0,0,...,0] \in R^{n+1}$, $x = [f, x_1] \in R^{n+1}$, $b = U$, $A_1 = [e_m - e_1, A] \in R^{m \times (n+1)}$ and

$A_2 = [0, I_n] \in R^{n \times (n+1)}$.

3. The first neural network model

In this section, we use the penalty method to solve the linear programming problem (3), and then construct a neural network model. In general, if we apply penalty method to solve (3), following unconstrained problem can be obtained:

$$\text{Maximize} \quad P(x) = cx - \frac{\mu}{2}\left[\sum_{i=1}^{n}\left(g_i^+(x)\right)^2 + \sum_{j=1}^{m}\left(h_j(x)\right)^2\right] \qquad x \ge 0, \tag{4}$$

wherever μ is a positive number and $g_i(x) = a^i x - b$, $h_j(x) = a^j x$ and $g_i^+(x) = \max\{0, g_i(x)\}$ ($i = 1,2,...,n$ *and* $j = 1,2,...,m$), n is number inequality constraints, m is number equality constraints, a^i is i'th row of matrix A_2 and a^j is j'th row of matrix A_1. So, the necessary condition for optimality of (4) is $\frac{\partial P(x)}{\partial x} = 0$, i.e.,:

$$c-\mu\left[\sum_{i=1}^{n}\left(g_i^{+}(x)\frac{\partial g_i^{+}(x)}{\partial x}\right)+\sum_{j=1}^{m}\left(h_j(x)\frac{\partial h_j(x)}{\partial x}\right)\right]=0 \qquad x\geq 0,$$

whenever

$$\frac{\partial g_i(x(t))}{\partial x}=\left[\frac{\partial g_i(x(t))}{\partial f},\frac{\partial g_i(x(t))}{\partial x_1},...,\frac{\partial g_i(x(t))}{\partial x_n}\right]=a^i,$$

$$\frac{\partial h_j(x(t))}{\partial x}=\left[\frac{\partial h_j(x(t))}{\partial f},\frac{\partial h_j(x(t))}{\partial x_1},...,\frac{\partial h_j(x(t))}{\partial x_n}\right]=a^j,$$

The neural network model for the maximum flow problem can be described by the following nonlinear dynamical system:

$$\frac{dx}{dt}=c-\mu\left[\sum_{i=1}^{n}\left(a^i\right)^t\left(a^i x-b_i\right)^{+}+\sum_{j=1}^{m}\left(a^j\right)^t\left(a^j x\right)\right], \qquad x\geq 0, \tag{5}$$

3.1. Stability analysis of the first neural network model

In this part, the stability of the equilibrium state and convergence of the neural network (5) to optimal solution are discussed. For nonlinear system, the most common method to show that a system is asymptotically stable is to use the Lyapunov function method. See Sontag (1998), Forti, et al. (1995). Assume that $v(x)=-P(x)$ and based on Theorem 3.1.1, $v(x)$ is as Lyapunov function and dynamical system is asymptotically stable at equilibrium state. So with respect to Theorem (3.1.2), obtain where optimal solution of maximum flow problem is equal to equilibrium state of (5).

Theorem 3. 1. 1: Under the penalty method, $v(x)$ of (4) is a Lyapunov function of system (5).

Proof:

$v(x)$ is a differentiable and positive definite on some neighborhood of equilibrium state, because $v(0)=0$ and μ is an arbitrary positive number so $v(x)>0$, for $x\neq 0$. It is sufficient for $x\neq 0$ show that, $\frac{\partial\left(v(x(t))\right)}{\partial t}<0$.

For this purpose with taking the derivative of $v(x)$ with respect to time t, for $x\neq 0$ we have:

$$\frac{d(v(x(t)))}{dt} = -c\dot{x} + \mu\left[\sum_{i=1}^{n}\left(a^i\right)^t\left(a^i x - b_i\right)^+ \dot{x} + \sum_{j=1}^{m}\left(a^j\right)^t\left(a^j x\right)\dot{x}\right] = \left(-\dot{x}\right)^t \dot{x} = -\left\|\dot{x}\right\|^2 < 0.$$

Thus $v(x)$ is a Lyapunov function.

Now in Theorem 3.1.2, we prove that the optimal solution of (4) is the equilibrium state of (5).

Theorem 3. 1. 2: If for any μ (4) has an optimal solution, and if for system (5) we can find a state variable $x(t)$, so that the neural network is asymptotically stable at x^*, then the optimal solution to (4) would be the equilibrium state of (5).

Proof:

The necessary condition for optimality of (4) is $\frac{\partial P(x)}{\partial x} = 0$, i.e.

$$c - \mu\left[\sum_{i=1}^{n}\left(a^i\right)^t\left(a^i x - b_i\right)^+ + \sum_{j=1}^{m}\left(a^j\right)^t\left(a^j x\right)\right] = 0.$$

This is equivalent to

$$-c + \mu\left[\sum_{i=1}^{n}\left(a^i\right)^t\left(a^i x - b_i\right)^+ + \sum_{j=1}^{m}\left(a^j\right)^t\left(a^j x\right)\right] = 0.$$

With regard to definition of stability in the equilibrium point we have, $\frac{dx^*}{dt} = 0.$

Using Theorem 3. 1. 1, system (5) is asymptotically stable, thus equilibrium state x^* satisfies (4) and this lead to this fact that optimal solution of (4) can be the same equilibrium state of (5). Now we introduce another neural network model for solving maximum flow problem that with start from any initial point, system has a much faster convergence rate from previous neural network model. In this model, we also obtain optimal solution of the primal and dual problems.

4. The second neural network model

In this model, maximum flow problem (3) is transformed to the following linear programming problem:

$$\begin{aligned} &\text{Maximize} && cx \\ &\text{Subject to} && Bx \le b \\ & && x \ge 0. \end{aligned} \tag{6}$$

$b = [0,0,...,0,U]^t \in R^{2m+n}$, $B = [A_1, -A_1, A_2]^t \in R^{(2m+n)\times(n+1)}$, where c, x, A_1, A_2 are defined in (3). Dual of the problem (6) is defined by:

$$\begin{aligned} &\text{Minimize} && b^t y \\ &\text{Subject to} && y^t B \ge c \\ & && y \ge 0. \end{aligned} \tag{7}$$

Using K. K. T conditions for inequality constrained problems, we have the following theorem:

Theorem 4. 1: x and y are optimal solutions to (6) and (7), respectively, if and only if x and y satisfy the following constraints:

$$\begin{aligned} &Bx - b \le 0 \\ &c - y^t B \le 0 \\ &y^t (Bx - b) = 0 \\ &(c - y^t B)x = 0. \end{aligned}$$

Proof:

See Bazaraa, et al. (1992).

Now the second neural network model can be described by the following nonlinear dynamical system:

$$\frac{dx}{dt} = c - \left(y + k\frac{dy}{dt} \right)^t B, \qquad x \ge 0 \tag{8}$$

$$\frac{dy}{dt} = -b + B\left(x + k\frac{dx}{dt} \right), \qquad y \ge 0. \tag{9}$$

The main property of the above system is stated in the following Theorem.

Theorem 4. 2: When the neural network whose dynamics is described by the differential equations (8) and (9), is asymptotically stable at equilibrium state, then equilibrium state is the optimal solutions for the linear programming problem (6) and its dual problem (7).

Proof:

See Effati, et al. (2004).

5. Numerical Example

Example 5. 1. Consider the maximum flow problem for following network. (See Bazaraa, et al. (1992, p. 565))

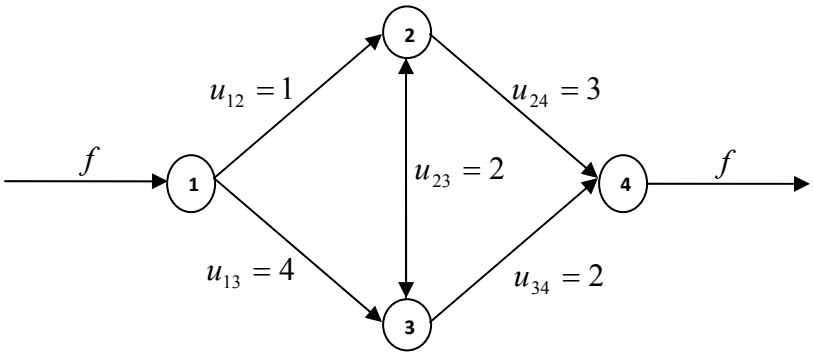

Fig. 1.

wherever f is maximum flow and u_{ij} is arc capacities (for example $u_{12}=1$). The source and destination are respectively vertices 1 and 4. The maximum flow is optimal solution from linear programming problem as

Maximize f

Subject to

$$
\begin{aligned}
& x_{12} + x_{13} = f \\
& x_{23} + x_{24} - x_{12} = 0 \\
& x_{34} - x_{13} - x_{23} = 0 \\
& -x_{24} - x_{34} = -f \\
& x_{12} \le 1 \\
& x_{13} \le 4 \\
& x_{23} \le 2 \\
& x_{24} \le 3 \\
& x_{34} \le 2 \\
& f, x_{12}, x_{13}, x_{23}, x_{24}, x_{34} \ge 0
\end{aligned}
$$

wherever x_{ij}'s are flows that proceed from node i to node j. Optimal solution which obtained by simplex method is:

$$f^* = 3, \quad x_{12}^* = 1, \quad x_{13}^* = 2, \quad x_{23}^* = 0, \quad x_{24}^* = 1, \quad x_{34}^* = 2.$$

For simplicity, let, $x_1 = x_{12}$, $x_2 = x_{13}$, $x_3 = x_{23}$, $x_4 = x_{24}$, $x_5 = x_{34}$, for the first model, above problem is transformed to the following problem:

Maximize f

Subject to

$$
\begin{aligned}
& h_1(x) = -f + x_1 + x_2 = 0 \\
& h_2(x) = -x_1 + x_3 + x_4 = 0 \\
& h_3(x) = -x_2 - x_3 + x_5 = 0 \\
& h_4(x) = f - x_4 - x_5 = 0 \\
& g_1(x) = x_1 - 1 \le 0 \\
& g_2(x) = x_2 - 4 \le 0 \\
& g_3(x) = x_3 - 2 \le 0 \\
& g_4(x) = x_4 - 3 \le 0 \\
& g_5(x) = x_5 - 2 \le 0 \\
& f, x_1, x_2, x_3, x_4, x_5 \ge 0.
\end{aligned}
$$

The neural network model for solving the maximum flow problem is the following nonlinear dynamical system:

$$\frac{dx}{dt} = c - \mu\left[\sum_{i=1}^{5}\left(a^i\right)^t\left(a^i x - b_i\right)^+ + \sum_{j=1}^{4}\left(a^j\right)^t\left(a^j x\right)\right], \qquad x \geq 0.$$

This is equivalent to:

$$\frac{dx}{dt} = c - \mu\left[A_2^t\left(A_2 x - b\right)^+ + A_1^t\left(A_1 x\right)\right],$$

whenever

$$c = [1,0,0,0,0,0],\ b = [1,4,2,3,2]^t,\ x = [f, x_1, x_2, x_3, x_4, x_5]^t,\ \frac{dx}{dt} = \left[\frac{df}{dt}, \frac{dx_1}{dt}, \frac{dx_2}{dt}, \frac{dx_3}{dt}, \frac{dx_4}{dt}, \frac{dx_5}{dt}\right]^t,$$

we also have:

$$A_1 = \begin{bmatrix} -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & -1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

By selecting $n = 300,\ \mu = 1000,\ dt = 0.0001,\ x_0 = [3,3,3,3,3,3]$ and using Euler method for solving above neural network model, the optimal solution is obtained as follows:

$$x^* = [3.004, 1.001, 2.002, 0.000, 1.002, 2.001]^t.$$

The first component of x^* is flow of maximum value problem, that is, $f^* = 3.004$ (see Fig. 2).

First neural network model which was proposed, for initial point $x_0 = [5,5,5,5,5,5]^t$, after 500 iteration, for $x_0 = [1,1,1,1,1,1]^t$, after 50000 iteration and for $x_0 = [0,0,0,0,0,0]^t$, after 80000 iteration is convergent to the optimal solution. Fig. 3 shows the trajectories of the system with three different initial points are convergent to the optimal solution.

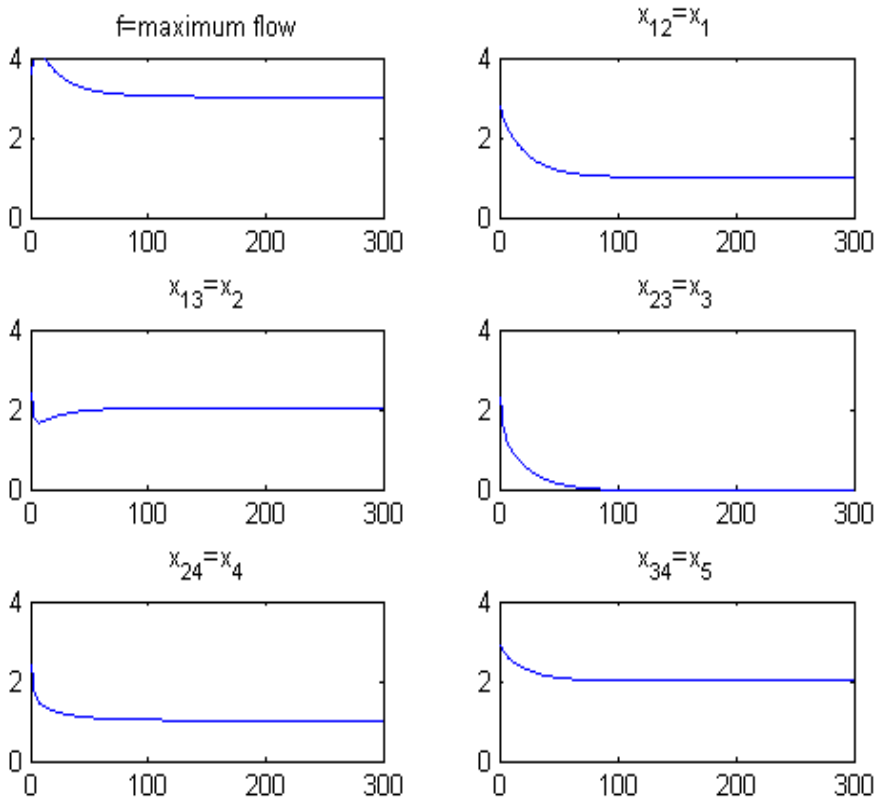

Fig. 2. State-variables obtained for the first neural network model.

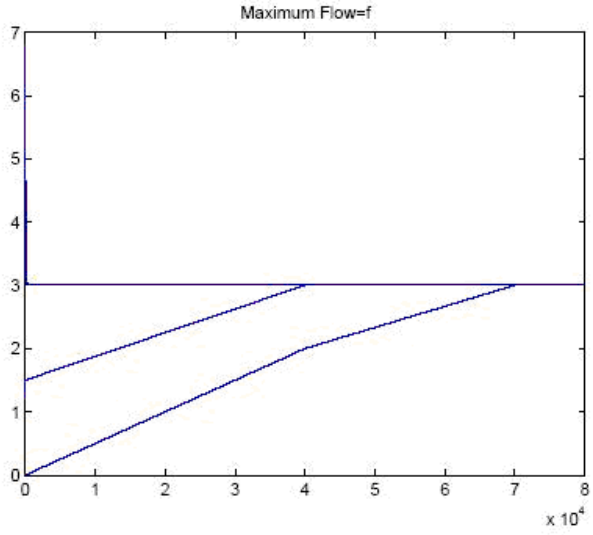

Fig. 3. Three trajectories with different initial points.

Now consider the second model of the neural network for example 5. 1, we have:

Maximize f

Subject to

$$
\begin{aligned}
&-f + x_1 + x_2 \le 0\\
&-x_1 + x_3 + x_4 \le 0\\
&-x_2 - x_3 + x_5 \le 0\\
&f - x_4 - x_5 \le 0\\
&f - x_1 - x_2 \le 0\\
&x_1 - x_3 - x_4 \le 0\\
&x_2 + x_3 - x_5 \le 0\\
&-f + x_4 + x_5 \le 0\\
&x_1 \le 1\\
&x_2 \le 4\\
&x_3 \le 2\\
&x_4 \le 3\\
&x_5 \le 2\\
&f, x_1, x_2, x_3, x_4, x_5 \ge 0
\end{aligned}
$$

Duality of above problem is:

Minimize $y_9 + 4y_{10} + 2y_{11} + 3y_{12} + 2y_{13}$

Subject to

$$
\begin{aligned}
&-y_1 + y_4 + y_5 - y_8 \ge 1\\
&y_1 - y_2 - y_5 + y_6 + y_9 \ge 0\\
&y_1 - y_2 - y_5 + y_6 + y_{10} \ge 0\\
&y_2 - y_3 - y_6 + y_7 + y_{11} \ge 0\\
&y_3 - y_4 - y_7 + y_8 + y_{13} \ge 0\\
&y_1, y_2, ..., y_{13} \ge 0.
\end{aligned}
$$

The second model for solving primal and dual problem is the following nonlinear dynamical system:

$$\frac{dx}{dt} = c - \left(y + k\frac{dy}{dt} \right)^t B, \qquad x \ge 0$$

$$\frac{dy}{dt} = -b + B\left(x + k\frac{dx}{dt} \right), \qquad y \ge 0.$$

wherever $c=[1,0,0,0,0,0]$, $b=[0,0,0,0,0,0,0,0,1,4,2,3,2]^t$, $B=[A_1,-A_1,A_2]^t$. By selecting $k=dt$, $n=2000$, $dt=0.1$, $x_0=[0,0,0,0,0,0]^t$, $y_0=[0,0,0,0,0,0,0,0,0,0,0,0,0]$ and using Euler method for solving above neural network model, primal and dual optimal solution is obtained as follows:

$$x^*=[3.011,1.004,2.006,0.000,1.006,2.004]^t$$

$$y^*=[0.084,0.648,0.207,0.630,0.577,0.141,0.701,0.123,0.999,0.000,0.000,0.000,0.999]^t$$

The first component of x^* is flow of maximum value, that is, $f^*=3.011$ (see fig. 4).

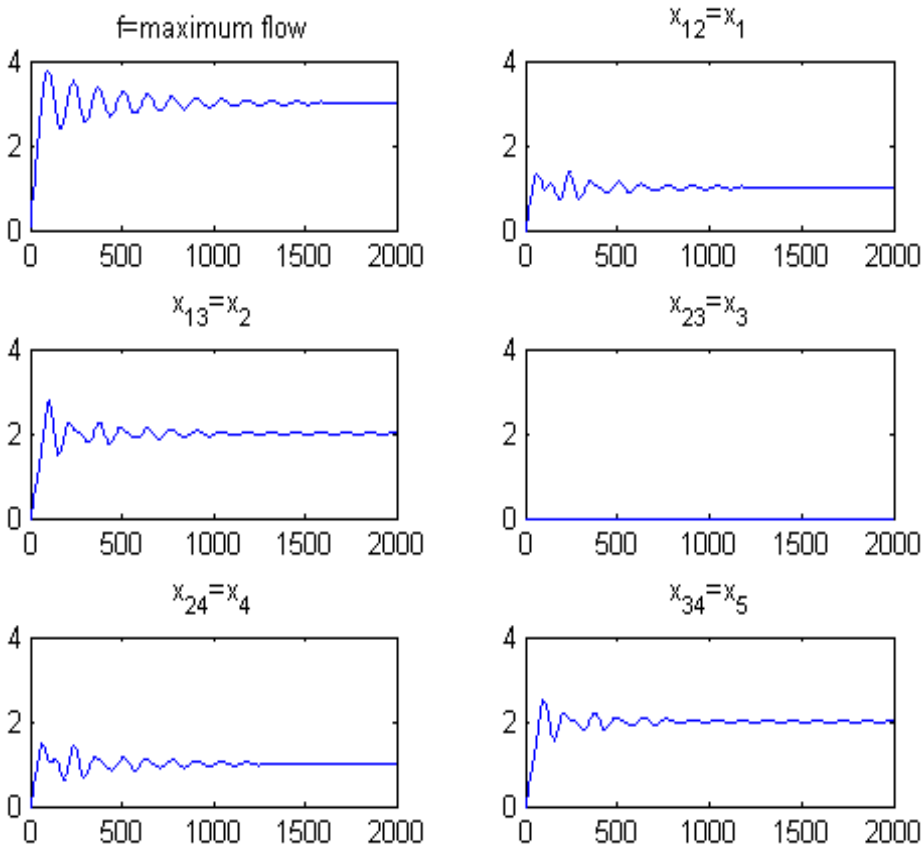

Fig. 4. State-variables obtained for the second neural network model.

Second neural network model for $dt=0.1$ and initial point $x_0=[0,0,0,0,0,0]^t$ after 2000 iteration, for $x_0=[1,1,1,1,1,1]$ after 2000 iteration and for $x_0=[5,5,5,5,5,5]$ after 2000 iteration is convergent to the optimal solution. Fig. 5 shows the trajectories of the system with three different initial points is convergent to the optimal solution. It is showed for a linear programming problem with 6 variable and 9 constrains, the first model is convergent after about 500 iteration and period of, 0.0058 seconds. Similarly, the second model is convergent after

about 2000 iteration and period of, 0.0210 seconds. In comparison, Xia model takes more than 6000 iterations to solve the same problem, using Xia model in period of, 0.1864 seconds (see Table 1).

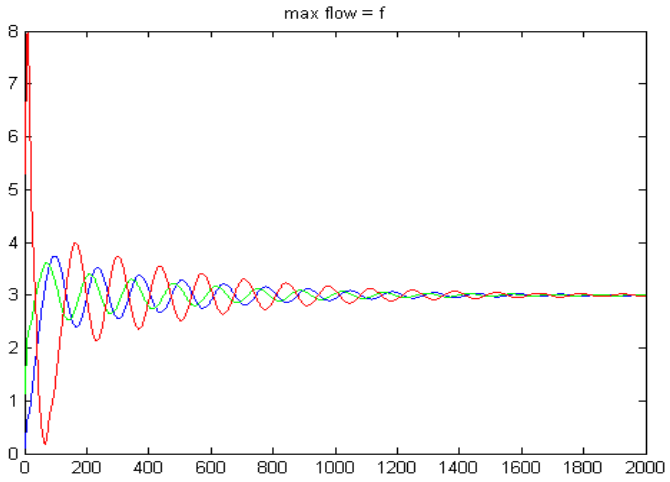

Fig. 5. Three trajectories with different initial points.

Model	Iterations	Maximum flow	Time(seconds)
Xia	6000	3.0017	0.1864
First model	500	3.003	0.0058
Second model	2000	3.0002	0.0210

Table. 1.

Example 5. 2. Consider the pipe network shown as in Fig. 6, it shows the flow capacities between various pairs of locations in ways. Determine a $r-s$ flow of maximum value in the following graph.

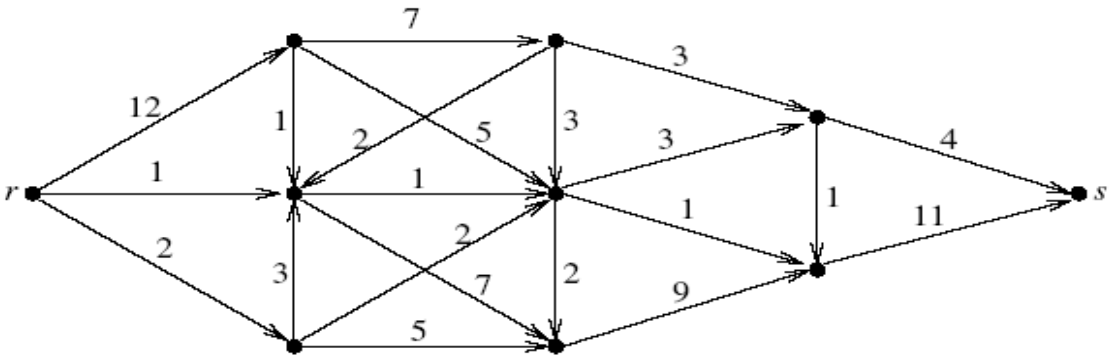

Fig. 6. Network showing pipe capacities.

If we consider maximum flow problem as a linear programming problem, we have a problem with 22 variables, 10 equality constraints and 21 inequality constraints. Flow of Maximum value which obtained by simplex method is $f^* = 14$.

We solve the maximum flow problem with the first, second and Xia model by selecting the initial point:

$$x_0 = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10].$$

By using Euler method, the results in are shown Table 2 and Fig. 7.

Model	Iterations	Maximum flow	Time(seconds)
Xia	2000000	14.026	376.205829
First model	5000	14.012	0.119542
Second model	10000	14.0006	0.140331

Table. 2.

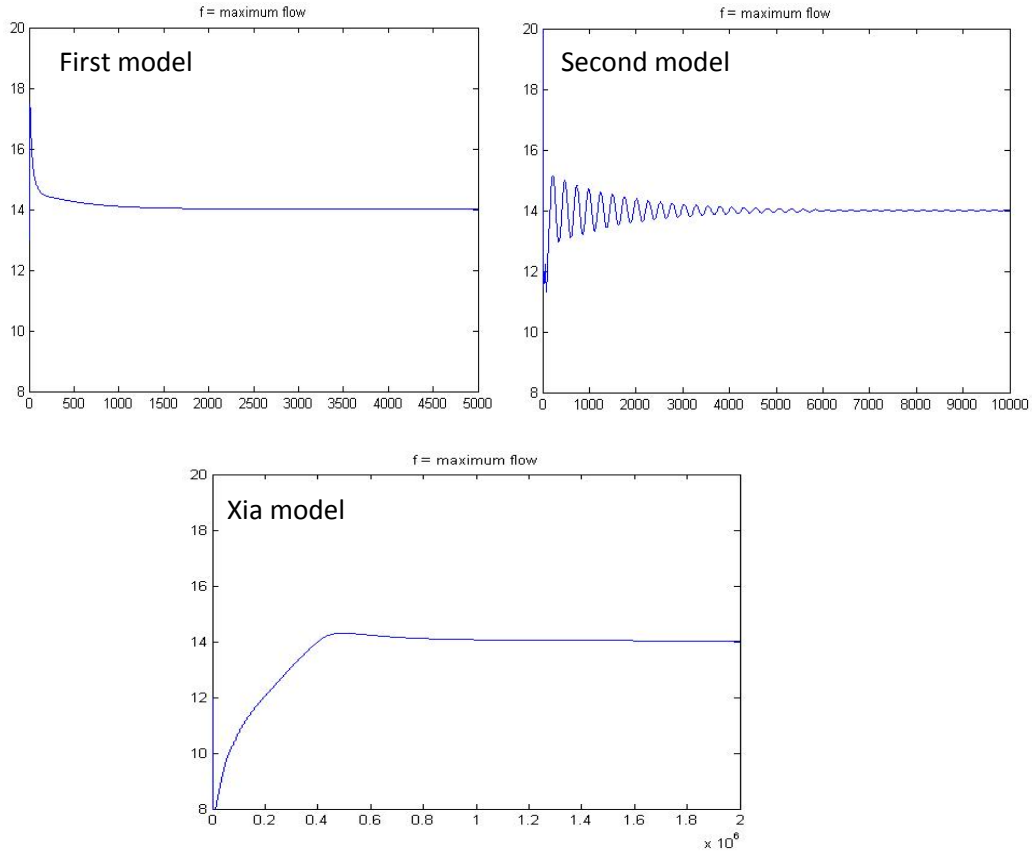

Fig. 7. Three trajectories with different iterations for the First, Second and Xia models.

The convergence rate of the first model depends on initial point, but the second model with the start of any initial point has a fixed convergence rate. By selecting:

$x_0 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]^t$, and using Euler method, you can see the results in Table 3.

Model	Iterations	Maximum flow	Time(seconds)
First model	600000	14.012	14.659978
Second model	10000	13.997	0.144184

Table. 3.

6. Conclusions

This paper presents two neural network models to solve maximum flow problem. To obtain the first neural network model, the first original problem is transformed into an unconstrained optimization problem, then constructed a nonlinear dynamic system. In the second neural network model using the K.K.T conditions, other nonlinear dynamic system is obtained. In this model, optimal solutions of the primal and dual problems are obtained, also with start of any initial point system of the nonlinear dynamic has a much faster convergence rate. Both models with start of any initial point are convergent to the optimal solution.

REFERENCES

Ahuja, R. k, T. L. Magnanti, and J. B. Orline (1993). Network Flows – Theory, Algorithms, and Application, Prentic Hall, Englewood Cliffs, New Jersey.

Ahuja, R. k and J. B. Orline (1995). A capacity scaling algorithm for the constraint maximum flow problem network, 25, pp. 89-98.

Bazaraa, M. S, J. J. Jarvis and H. D. Sherali (1992). Linear programming and network flows, John Wiley and Sons, New York.

Bazaraa, M. S and C. M. Shetty (1979). Nonlinear programming, theory and algorithms, John Wiley and Sons, New York.

Dantzig, G. B and M. N. Thapa (1997). Linear programming Springer, New York.

Effati, S and M. Baymani (2005). A new nonlinear neural network for solving quadratic nonlinear programming problems, Applied Mathematics And Computation 165, 719-729.

Effati, S and M. Baymani (2005). A new nonlinear neural network for solving convex nonlinear programming problems, Applied Mathematics And Computation 168, 1370-1379.

Ford, L. R and D. R. Fulkerson (1956). Maximal flow through a network, Canadian Journal of Mathematics 8, 399-404.

Forti, M and P. Nistri (2003). Global convergence of neural networks with discontinuous neuron activations, IEEE Trans, Circuits Syst. I 50 (11), 1421-1435.

Forti. M and A. Tesi (1995). New condition for global stability of neural networks with application to linear and quadratic programming problems, IEEE Trans. Circuits Syst. I 42 (7), 354-366

Hopfield, J. J and D. W. Tank, *Neural computation of decisions in optimization problems,* Biol. Cybern. 52 (1985) 141-152.

Kennedy, M. P and L. O. Chua, *Neural networks for nonlinear programming*, IEEE Trans. Circuits Syst. 35 (1988) 554-562.

Leung, Y, G. Li and Z. B. Xu*, A genetic algorithm for the multiple destination routing problems*, IEEE Trans. Evol. Comput, vol. 2, 99 (1998) 150-161.

Sontag, E. D, *Mathematical control theory deterministic finite dimensional systems*, Springer, New York, (1998).

Tank, D. W and J. J. Hopfield, *Simple neural optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit*, IEEE Trans. Circuits Syst. 33 (1986) 533-541.

Vazquez, A. R, R. D. Castro, A. Rueda, J. L. Huertas and E. S. Sinencio, *Nonlinear switched-capacitor neural networks for optimization problems*, IEEE Trans. Circuits Syst. 37 (1990) 384-397.

Xia, Y and J. Wang (2000). A projection neural network and its application to constrained optimization problems*,* IEEE Trans. Circuits Syst. 49, 447-457.

Wu, Y, Y. Xia, J. Li and W. Chen (1996). A high-performance neural network for solving linear and quadratic programming problems*,* IEEE Trans. Neural Networks 7, 643-651.