Solving Nonlinear Continuous Constrained Optimization Problems Using Neural Networks

[1] Dr. Thabit Sultan Mohammed, and [2] Mr. Essa Ibrahim Essa
[1]Al-Zaytoonah University, P.O.Box-130, Amman 11733-Jordan, email: thabitsm@yahoo.com
[2] College of Science, University of Kirkuk, Iraq, email: essaibrahimessa@yahoo.com

ABSTRACT

Nonlinear Programming (NP) problems have enormous practical applications, and as such they have been the subject of extensive research from both the theoretical and the practical aspects. Such applications require solutions to satisfy a set of non-linear constraints over real numbers or to optimize a non-linear function subject to non-linear constraints. Many researches are made and numerous nonlinear programming algorithms have been developed. The main goal of this paper is to provide illustrations of how neural networks can be used as a computationally efficient and relatively simple tool for implementing some of the well-known nonlinear programming techniques, namely, penalty function method and, augmented Lagrange multiplier methods.

Key words: nonlinear programming, neural networks, penalty function method, augmented Lagrange method, steepest descent approach.

1. Introduction

There are several deterministic, stochastic, and cooperative solvers in solving both constraint satisfaction and constraint optimization problems. The methods include sequential quadratic programming (SQP), static penalty method, dynamic penalty method, augmented Lagrangian method, Neldermead simplex method and simulated annealing (SA).

Before we mention some of the previous work in the area and before addressing the neural implementation of NP algorithms, let us revisit the definition of the problem. Based on the form the constraints, we can define three different forms of NP minimization problems [1]:

- NP1 (NP problem with equality constraints);

Minimize $f(x) = f(x_1, x_2, ..., x_n)$ … (1)

Subject to $h_i(x) = 0$

where $i = 1, 2, \ldots, m$

- NP2 (NP problem with inequality constraints):

Minimize $f(x) = f(x_1, x_2, ..., x_n)$ … (2)

Subject to $g_i(x) \leq 0$

where $i = 1, 2, \ldots, m$

- NP3 (NP problem with mixed constraints):

Minimize $f(x) = f(x_1, x_2, ..., x_n)$ … (3)

Subject to $h_i(x) = 0$

where $i = 1, 2, \ldots, p$ and

$g_j(x) \leq 0$ where $j = p+1, p+2, \ldots, m$.

where $x \in \Re^{n\times 1}$ is the vector of the independent variables, $f(x) : \Re^{n\times 1} \to \Re$ is the objective function, and functions $h_i(x), g_i(x) : \Re^{n\times 1} \to \Re$ represent constraints. To simplify the derivations of the algorithms we will assume that both the objective function and the constraints are smooth differentiable functions of independent variables.

The study of algorithms for solving constrained optimization problems is difficult unless the problems can be represented in a unified way. Using surplus variables, the inequality constraints in problems NP2 and NP3 can be converted to equality constraints. Similarly, each of the equality constraints can be converted to a pair of inequality constraints according to:

$$h_i(x) = 0 \Leftrightarrow h_i(x) \leq 0 \quad and \quad h_i(x) \geq 0 \qquad \ldots (4)$$

Earlier versions of the augmented Lagrange method suggested different update rates for the solution of x and Lagrange multipliers [2, 3]. In these early versions a complete unconstrained minimization was performed relative to x before each update of the Lagrange multipliers.

There is also a class called Lagrangian relaxation [4, 5, 6, 7], which are based on

Lagrangian Duality theory [8]. These algorithms aims to find an optimal solution given an optimal dual solution, or vice versa. This approach is simple in the case of linear functions but does not work well for non-linear functions.

Penalty transformations is an approach to transform a continuous constrained problem into unconstrained problem, consisting of a sum of its objective and its constraints weighted by penalties, before solving the penalty function by unconstrained methods [9,10, 11, 12].

Simulated annealing [13,14,15] and genetic algorithms [16, 17,18] are two stochastic optimization algorithms for solutions. Both approaches require a good choice of penalties in a penalty formulation for the search to converge to a constrained global minimum or otherwise may end up with infeasible solutions.

There are number of software solvers. Some of them are available on the world wide web (WWW). Packages such as; (LGO, BARON,SBB, DICOPT) can be called from the GAMS modeling system [19]. Other examples of software solvers are Genocop, and COBYLA2 [20].

2. Neural Networks for Penalty Function NP Methods

Methods using penalty functions make an attempt to transform the NP problem to an equivalent unconstrained optimization problem, or to a sequence of constrained optimization problems. This transformation is accomplished through modification of the objective function so that it includes terms that penalize every violation of the constraints. In general, the modified objective function takes the following form:

$$f_A(x) = f(x) + \sum_{i=1}^{p} K_i^{(1)} \Phi_i^{(1)} \left[h_i(x) \right] + \sum_{i=p+1}^{m} K_i^{(2)} \Phi_i^{(2)} \left[g_i(x) \right] \quad \ldots (5)$$

Functions $\Phi_i^{(1)}$ *and* $\Phi_i^{(2)}$ are called *penalty functions*, and they are designed to increase the value of the modified objective function $f_A(x)$ whenever the vector of independent variables violates a constraint, or in other words whenever it is outside the *feasible* region. Penalty functions are commonly selected as at least one-time differentiable. Table-I- shows the requirements to satisfy the conditions of the penalty functions as well as their common choices.

Table-I-: Constraints and common choices for the penalty functions.

$\Phi_i^{(1)}$	
Constraints	$\Phi_i^{(1)} > 0 \quad \text{for } h_i(x) \neq 0$ $\Phi_i^{(1)} = 0 \quad \text{for } h_i(x) = 0$
Common choices	$\frac{1}{2}u^2$
	$\frac{1}{r}\lvert u \rvert^r \qquad r > 0$
	$\cosh u - 1$
	$\ln\left[\frac{1}{2}(e^u + e^{-u})\right]$

$\Phi_i^{(2)}$	
Constraints	$\Phi_i^{(2)} > 0 \quad \text{for } g_i(x) > 0$ $\Phi_i^{(2)} = 0 \quad \text{for } g_i(x) \leq 0$
Common choices	$\max\{0, u\}$
	$(\max\{0, u\})^2$

For example, the typical modified objective function for the NP3 problem can be written as

$$f_A(x) = f(x) + \sum_{i=1}^{p} \frac{K_i^{(1)}}{r_1} \lvert h_i(x) \rvert^{r_1} + \sum_{i=p+1}^{m} K_i^{(2)} \max\{0, g_i(x)\}^{r_2} \ldots (6)$$

where $r_1, r_2 \geq 0$. Parameters $K_i^{(1)}, K_i^{(2)} \geq 0$ are commonly referred to as *penalty parameters* or *penalty multipliers*, and in (6) we have assumed that a separate penalty parameter is associated with each of the penalty functions. In practice this is rarely the case, and commonly there is only one parameter multiplying the entire penalty term, that is,

$$f_A(x) = f(x) + K\left[\sum_{i=1}^{p} \frac{1}{r_1}\left|h_i(x)\right|^{r_1} + \sum_{i=p+1}^{m} \max\{0, g_i(x)\}^{r_2}\right] = f(x) + KP(x) \quad \ldots (7)$$

where $P(x)$ represents the penalty term.

From the form of the augmented objective function in (7), it should be obvious that the solution resides in the region where the value of the penalty function $P(x)$ is small. As a matter of fact, if K is increased towards infinity, the solution of the unconstrained problem will be forced into the feasible region of the original NP problem. Remember that if the point is in the feasible region, all the constraints are satisfied and the penalty function equals zero. In the limiting case, when $K \to \infty$, the two problems become equivalent. The following theorem summarizes the equivalence of the two problems.

Theorem 1 [21]***:-***

Consider an NP problem given as follow:

Minimize $f(x) = f(x_1, x_2, ..., x_n)$ … (8)

subject to

$$x \in S, \qquad S \subset \Re^{n \times 1}$$

where S is constrained set (i.e., the feasible region) defined by a number of either equality or inequality constrains. Define a sequence of unconstrained optimization problems as follows:

Minimize $q(K_j, x) = f(x) + K_j P(x)$ … (9)

where P is a penalty function satisfying:

$$P(x) \geq 0 \quad \text{for } x \in \Re^{n \times 1}$$

$$P(x) = 0 \quad \text{if and only if } x \in S$$

and $K_j, j = 1,2,...$ is a sequence of real numbers satisfying :

$(K_j > 0 \quad \forall j, \; K_{j+1} > K_j \quad \forall j,$

and $K_j \to \infty$ as $j \to \infty$)

The above equations show that a sequence of unconstrained optimization problems is generated, and the solution to the generated sequence converges to the solution of the original NP problem. When adopting neural networks as a tool for solution, some modifications on the implementation of the penalty methods are usually accomplished in either of following two ways:

1. Penalty parameter K is made time-varying, and it is increased over the course of network training.
2. Penalty parameter K is selected as a sufficiently large positive number to ensure that the unconstrained problem represents a close approximation to its NP counterpart.

Once the modified objective function is specified, any one of many gradient techniques can be used to perform the optimization task. For the sake of simplicity, we will demonstrate the use of steepest descent method. However, the conjugate gradient, Newton's, and quasi-Newton methods can offer significantly faster convergence rates [4].

Applying the steepest descent approach, we can generate the update equations in accordance with

$$x(k+1) = x(k) - m\frac{\partial f_A(x)}{\partial x} \quad \ldots (10)$$

where $m > 0$ is the *learning rate parameter*, and the gradient term on the right-hand side of (10) depends on the penalty function selection. For example, when the form of the objective function given in (7), is considered, with $r_1 = 2,$ and $r_2 = 1$ we have:

$$\frac{\partial f_A(x)}{\partial x} = \frac{\partial f(x)}{\partial x} + K\left[\sum_{i=1}^{p} \frac{\partial h_i(x)}{\partial x} h_i(x) + \sum_{i=p+1}^{m} \frac{\partial}{\partial x} \max\{0, g_i(x)\}\right] \quad \ldots (11)$$

After substituting (11) into (10), we have for the learning rule

$$x(k+1) = x(k) - m\left[\frac{\partial f(x)}{\partial x} + K\sum_{i=1}^{p} \frac{\partial h_i(x)}{\partial x} + K\sum_{i=p+1}^{m} \frac{\partial}{\partial x} \max\{0, g_i(x)\}\right] \quad \ldots (12)$$

The neural network architecture realization of this process in equation (12) is presented in Figure (1).

3. Augmented Lagrange Methods for NP2 Problems

There are several important properties of augmented Lagrange multiplier method that need to be taken into consideration [2]:

- *Local-minimum property.* Similar to the penalty function approach, the augmented Lagrange multiplier method guarantees convergence to the local minimum of the augmented Lagrangian. The local minimum of the Lagrangian converges to the constrained minimum of the objective function only in the limiting case when the penalty parameters in *k* are sufficiently large.
- *Choice of the penalty parameter.* In general, the penalty parameters need to be chosen so that the Hessian matrix of the augmented Lagrangian is positive definite. If the values of the penalty parameters are too small the algorithm may fail to converge or it may converge to a value that is a local minimum of the augmented Lagrangian but dose not minimize the objective function itself. On the other hand, if the parameters are chosen too large, the algorithm may exhibit oscillatory in the vicinity of solution.
- *Convergence of the Lagrange multipliers.* To find the optimal solution it is necessary that both x and l converge to their optimal values $\tilde{x}$ and $\tilde{l}$. In some cases the augmented Lagrangian can be very sensitive to the values for the multipliers, and it may take a considerable number of iterations before convergence is achieved.

The method of the augmented Lagrange multipliers can be extended to NP problems with inequality constrains. To accomplish this, the augmented Lagrange is modified according to

$$L(x, l) = f(x) + \sum_{i=1}^{m} l_i \max\{0, g_i(x)\} + \sum_{i=1}^{m} \frac{K_i}{2} \max\{0, g_i(x)\}^2 \quad \text{.... (13)}$$

where l_i, i =1,2,…,m are *Lagrange multipliers* and $,K_i$,i = 1,2,…,m, are the penalty parameters. As can be seen from (13), any violation of the constraints increases the value of the Lagrangian; that is, only violated constraints are considered to be active. In a more compact form, (13) can be written as

$$L(x, l) = f(x) + \sum_{i=1}^{m} S_i \left[l_i g_i(x) + \frac{K_i}{2} g_i(x)^2 \right] \quad \text{… (14)}$$

where

$$S_i = \begin{cases} 0 & if \;\; g \;\; (x) \le 0 \\ 1 & if \;\; g \;\; (x) > 0 \end{cases}$$

To derive a corresponding neural network, the updated equations can be obtained according to

$$x(k+1) = x(k) - m_x \frac{\partial L(x, l)}{\partial x} \quad \text{… (15)}$$

and

$$l(k+1) = l(k) + m_l \frac{\partial L(x, l)}{\partial l} \quad \text{… (16)}$$

After substitution of the appropriate gradients of (14) into (15) and (16), we have

$$x_j(k+1) = x_j(k) - m_x \left\{ \frac{\partial f(x(k))}{\partial x_j} + \sum_{i=1}^{m} S_i \left[l_i + K_i g_i(x(k)) \right] \frac{\partial g_i(x(k))}{\partial x_j} \right\} \quad \text{… (17)}$$

and

$$l_j(k+1) = l_j(k) + m_l S_j g_j(x(k)) \quad \text{… (18)}$$

A neural network architecture realization of this process is shown in figure (2).

4. Implementation of Neutral Networks and Results

The following examples show how neural networks are adopted in solving nonlinear continuous constrained problems. In both examples the Matlab package is used, where the neural networks of figures (1), and (2) are simulated, and the results are shown in this section.

Example.1:- Consider the following NP problem:

Minimize $f(x) = \exp\left[(x_1 - 1.5)^2 + x_2^2\right]$

subject to $x_1^2 + x_2^2 - 1 \le 0$

The above problem is obviously an NP problem with inequality constraints (i.e., NP2). The modified penalty function can be formed as:

$$f_A(x) = \exp\left[(x_1 - 1.5)^2 + x_2^2\right] + \frac{K}{2}\max\left\{0, x_1^2 + x_2^2 - 1\right\}$$

By using the steepest descent method, the update equations can be computed as

$$x_1(k+1) = x_1(k) - m\Big\{2(x_1 - 1.5) \exp\left[(x_1 - 1.5)^2 x_2^2\right] + K\left[\operatorname{sgn}(x_1^2 + x_2^2 - 1)\right]x_1\Big\}$$

and

$$x_2(k+1) = x_2(k) - m\Big\{2x_2 \exp\left[(x_1 - 1.5)^2 + x_2^2\right] + K\left[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1\right]x_2\Big\}$$

where $\operatorname{sgn}(u) = {}^{u}\!/_{|u|}$ is the sign function. The neural network architecture shown in Figure (1) was used to determine the solution of the NP problem. The neural network architecture realization of this process is presented in the neural network architecture consists of three separate modules, the first one module performs the objective function, the second module performs the equality constraints, and the last one module performs the inequality constraints. The results was tested, and verified in Matlab. Parameters of the network were chosen as k=5, and p =0.01, and initial solution was set as $x = \begin{bmatrix}0 & 1.5\end{bmatrix}^T$. The network converged in approximately 900 iterations.

Example.2:-Let us consider the same problem in the above example, but using the method of augmented Lagrange multipliers. For the NP problem with equality constraints:

Maximize	$f(x) = f(x_1, x_2, ..., n)$
subject to	$h_i(x) = 0$
	where i = 1,2,…, m

The augmented Lagrangian can be formed as:

$$L(x, l) = \exp\left[(x_1 - 1.5)^2 + x_2^2\right] + l \max\left\{0, x_1^2 + x_2^2 - 1\right\} + \frac{K}{2}\max\left\{0, x_1^2 + x_2^2 - 1\right\}^2$$

The appropriate derivatives can be computed as

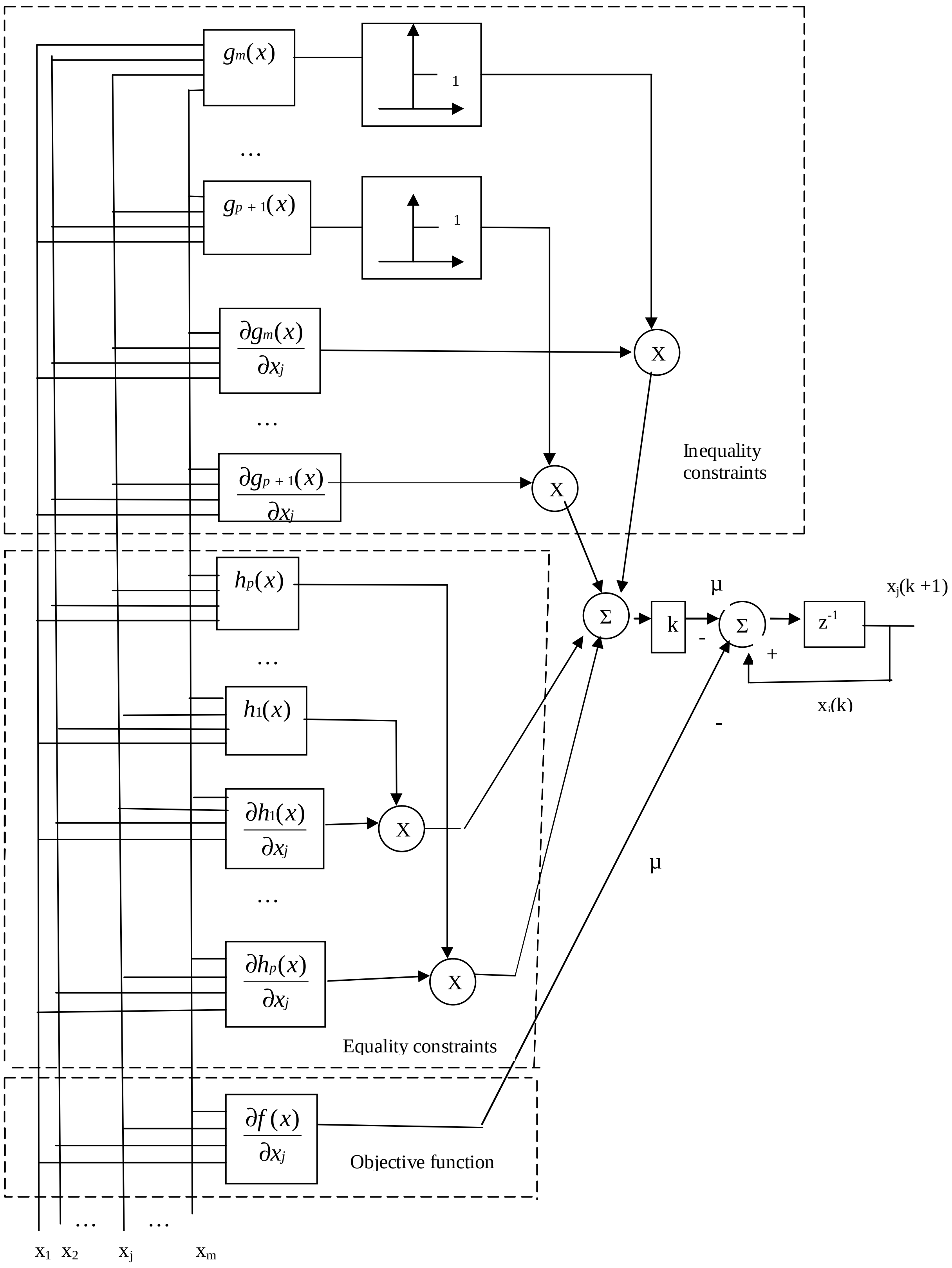
$$\frac{\partial L(x, l)}{\partial x_1} = 2(x_1 - 1.5)\exp\left[(x_1 - 1.5)^2 + x_2^2\right] + 2x_1\left[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1\right]\left[l + K(x_1^2 + x_2^2 - 1\right]$$

$$\frac{\partial L(x, l)}{\partial x_2} = 2x_2 \exp\left[(x_1 - 1.5)^2 + x_2^2\right] + 2x_2\left[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1\right]\left[l + K(x_1^2 + x_2^2 - 1\right]$$

and

$$\frac{\partial L(x, l)}{\partial l} = \left[\operatorname{sgn}(x_1^2 + x_2^2 - 1) + 1\right](x_1^2 + x_2^2 - 1)$$

The network shown in Figure (2) was used to perform the optimization task. The neural network architecture consists of two separate modules. The first module performs the update of the solution, and the second module updates the augmented Lagrange multiplier. The parameters of the network were set to the values used in example1, that is k=5, and $m = 0.01$. Initial conditions were chosen as $x_1(0) = 0$, $x_2(0) = 0$, and $l(0) = 1$. The network converged in approximately 700 iterations to the solution $\tilde{x} = [1.0088, 0.0002]^T$.



$g_m(x)$
1
…
$g_{p+1}(x)$
1
$\frac{\partial g_m(x)}{\partial x_j}$
X
…
$\frac{\partial g_{p+1}(x)}{\partial x_j}$
X
Inequality constraints
$h_p(x)$
…
$h_1(x)$
$\frac{\partial h_1(x)}{\partial x_j}$
X
…
$\frac{\partial h_p(x)}{\partial x_j}$
X
Equality constraints
$\frac{\partial f(x)}{\partial x_j}$
Objective function
Σ
k
μ
Σ
-
+
z^{-1}
$x_j(k+1)$
$x_j(k)$
-
μ
x_1 x_2 … x_j … x_m

Figure (1): Discrete-time network for NP3 problems implementing penalty function method; implementation of Equation (12).

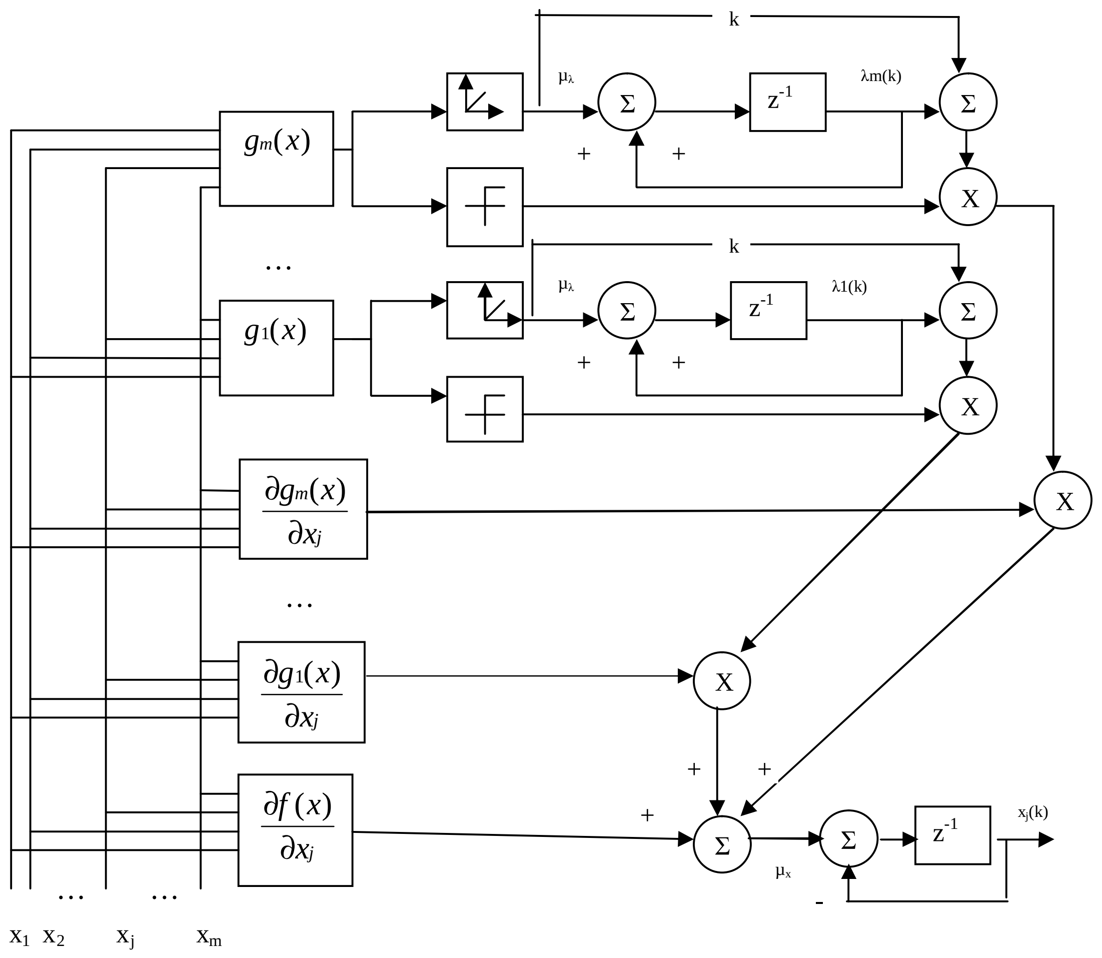


Figure (2): Discrete-time realization of the neural network for implementation of augmented Lagrange multiplier method for NP problems with inequality constrains. Equations (17) and (18) show the two learning rules.

5- Conclusions

Among the many methods used for solving nonlinear continuous constrained optimization problems, neural networks is proposed as an efficient method. The main idea of solution is to transform a nonlinear problem to an equivalent unconstrained optimization problem. Both the penalty function methods and the augmented Lagrange method problems are modeled using neural networks. The network converged to give the solution of a penalty function problem after about 900 iterations, while about 700 iterations are required for the network to converge for a solution for an augmented Lagrange problem.

Frankly, if the coefficients in k are sufficiently large, the Hessian Matrix of the Lagrangian can be forced to have all eigenvalues greater than zero, which is of great importance from the neural network implementation standpoint, since the solution has to be sought in an iterative manner. The fact that the Hessian matrix is positive definite ensures existence of a neighborhood of the solution in which the function convex; and therefore, if the initial point in the optimization process is properly chosen, the algorithm will converge to global minimum. The accuracy of the solution depends on the accuracy of the Lagrange multiplier estimates, and computing will not converge to the optimal value of $x = \tilde{x}$ until the multipliers converge to optimal value $l = \tilde{l}$.

Therefore, every NP problem can be transformed into either the NP1 or NP2 form. However, it is preferred from the computational standpoint to consider the NP problem in its original form. Finally, note that the NP1 or NP2 form of the problem can be regarded as special cases of a more general NP3.

References

1- Cichocki, A. and R. Unbhauen, “Neural Networks for Optimization and Signal Processing”, Chichester, England: Wiley, 1993.

2- Gill, P. E., W. Murray, and M. H. Wright, “Practical Optimization”, London: Academic, 1981.

3- Powell, M. J. D.," A Method for Nonlinear Constraints in Minimization Problems", In Optimization, Ed, R. Fletcher, London; Academic, pp.283-288, 1969.

4- Geoflrion, A.M., “Lagrangian Relaxation for Integer Programming”, Mathematical Programming Study, 2:82-114, 1974.

5- Giles, F.R., and W.R. Pulleyblank, ”Total Dual Integrality and Integer Polyhedra”, Vol. 25, Elsevier North Holland Inco., 1979.

6- Gavish, B., “On Obtaining the Best Multipliers for Lagarangian Relaxation for Integer Programming”, Computer & Operational Research, 5:55-71, 1978.

7- Shang, Y. and B.W. Wah,” A Discrete Lagarangian Based Global Search Method for Solving Satisfiability Problems”, J. of Global Optimization, 12(1): 61-99, January 1998.

8- Tind, J., and L.A. Wolsey, ” An Elementary Survey of General Duality Theory in Mathematical Programming”, Mathematical Programming, pp:241-261, 1981.

9- Back, T., F. Hoffmeister, and H.P. Shwefel,” A Survey of Evolution Strategies”, In Proc. of 4th Int. Conf. on Genetic Algorithms.

10- Glover, F. and G. Kochenberger, “Critical Event Tabu Search for Mathematical Knapsack Problems”, In Proc. of Int. Conf. on Metaheuristic for Optimization”, pp: 113-133, 1995.

11- Joines, J., and C. Honck, “On the Use of Non-stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems”, In Proc. of Int. Conf. on Evolutionary Computation, pp: 579-584, 1994.

12- Michalewicz, Z., D. Dagupta, R.G. LeRiche, and M. Schoenauer, “Evolutionary Algorithm for Constrained Engineering Problems”, Computers and Industrial Engineering Journal, 30 (2): 851-870, 1996.

13- Wah, B.W., and T. Wang, “Simulated Annealing with Asymptotic Convergence for Non-linear Constrained Global Optimization”, Principles and Practice of Constrained Programming, 10: 461-475, 1990.

14- Su, H., and R. Hartley, “Fast Simulated Annealing”, Physics Letters, A, 122(3-4): 157-162, 1987.

15- Joong, I. K., and J.J. Lee, “Adaptive Simulated Annealing Genetic Algorithm for System Identification”, Eng. Application of Artificial Intelligence, 9(5): 523-532, 1996.

16- Homaifar, A., S. H. Y. Lai, and X. Qi, “Constrained Optimization via Genetic Algorithms”, Simulation, 62:242-254, 1994.

17- Chen, X., “Optimal Anytime Search for Constrained Nonlinear Programming”, MSc Thesis, University of Illinois at Urbana-Champaign, 2001.

18- Petridis, V., S. Kazarlis, and A. Bakirtzis, “Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting Stock and Unit Commitment Problems”, IEEE Trans. on System, Cybernetic, 28(5): 629-640, 1998.

19- GAMS World, WWW-document 2003. [http://www.gamsworld.org.]

20- Michalewics, Z., and G. Nazhiayath, “Genocop III: A Co-Evolutionary Algorithm From Numerical Optimization Problems with Nonlinear Constraints”, Proceedings of the IEEE Inter. Conf. on Evolutionary Computation, 2:647-651, 1995.

21. Luenberger, D. G., “Linear and Nonlinear Programming”, 2nd ed., Springer, 2003.