The Journal of
Mathematics and Computer Science

Available online at
http://www.TJMCS.com

The Journal of Mathematics and Computer Science Vol .5 No.3 (2012) 160-166

A Neural Network Approach to solve Semi-Infinite Linear Programming Problems

Alireza Fakharzadeh Jahromi
Department of Mathematics, Faculty of basic Sciences, Shiraz University of Technology, Shiraz, Iran
a_fakharzadeh@sutech.ac.ir

Zahra Alamdar Ghahfarokhi,
Department of Mathematics, Faculty of basic Sciences, Shiraz University of Technology, Shiraz, Iran
zhalamdar@gmail.com

Masoumeh Hosseinipour
Department of Mathematics, Faculty of basic Sciences, Shiraz University of Technology, Shiraz, Iran
m.hosseinipour@sutech.ac.ir

Received: February 2012, Revised: November 2012
Online Publication: December 2012

Abstract
In this article, we present a new algorithm for solving Semi-Infinite Linear Programming (SILP) problems based on an artificial neural network concept. First the local reduction method for solving the SILP problems is introduced. Based on the local reduction method, the Karush-Kuhn-Tucker (KKT) conditions and gradient method are used to convert the SILP problem to an unconstrained optimization problem; then, a neural network model is constructed to solve it. Numerical example has been employed to indicate the accuracy of the new method.

Keywords: Semi-Infinite linear programming; Neural network; Local reduction method; KKT conditions.

2010 Mathematics Subject Classification: Primary 54A40; Secondary 46S40.

1. Introduction.

In programming modeling of real phenomenon, we frequently face to the continuous parameters such as time, space, weight and so on. Since usually the number of variables or constraints in such models is infinite, these problems are pertained to the semi-infinite programming category.

A Semi-Infinite Linear Programming (SILP) problem is a linear optimization problem with a finite number of variables and an infinite number of constraints. Like [4], the primal form of a SILP problem can be formulated mathematically as:

$$(P) \quad \text{Min: } c'x$$
$$\text{S. to: } a_t'x \geq b_t, t \in T,$$

where $c \in \mathbb{R}^n$, $T \neq \emptyset$ is the index set, $a_t \equiv a(t) = (a_1(t), a_2(t), \dots, a_n(t))$ maps T onto $\mathbb{R}^n$ and $b_t \equiv b(t)$ is a scalar function on T. If $T \subseteq \mathbb{R}^m$, be a countably infinite set, then the SILP problem is called countable. Also, if T is taken as a continuous subset of $\mathbb{R}^m$, the problem is called continuous semi-infinite linear program. It is also proved in [4] that the natural dual of (P) is:

$$(D) \quad \text{Max: } \int_T b(t)\, d\mu$$
$$\text{s.to: } \int_T a(t)\, d\mu = c,$$
$$\mu \in \mathcal{M}^+(t),$$

where $\mathcal{M}^+(t)$ is the set of all positive Radon measures on T. In absence of the duality gap (see [4]), different methods are available to solve the primal and the dual form of the SILP problems. Among them, we mention to Local Reduction and Discretization methods ([4]), Three-Phase approach ([2]), Primal and Dual Exchange methods ([11]), Perturbation technique ([16]) and Directions method ([7]).

Regarding the successfulness and effectiveness applications of the neural network in different branch of science, especially in optimization, one possible and very promising approach to real-time optimization is to apply artificial neural networks. Hopfield and Tank ([5] and [14]) first proposed a neural network for solving linear programming problems. Kennedy and Chua [6] proposed a neural network for solving nonlinear programming problems, which employs both gradient method and penalty function method and its equilibrium points correspond to approximate optimal solution only. Applying the gradient method and switched-capacitor technology, Rodriguez-Vazquez et al. [12] proposed a class of neural networks for solving optimization problems, in which their design does not require the calculation of a penalty parameter. Using gradient and projection methods [9], Bouzerdoum and Pattison [1] presented a neural network for solving quadratic optimization problems with bounded variables only. Based on dual and projection methods ([8], [10]) Xia and Xia et al. [17]-[21] presented several neural networks for solving linear and quadratic programming problems with nonunique solutions, which are proved to be globally convergent to exact solution.

Prevalent and conventional methods are gradient methods which involves constructing an appropriate computational energy function for the optimization problem and designing a neural network model which performs some form of gradient descent on that function. The gradient methods have an advantage in neural network models that may be defined directly using the derivative of the energy function. But its shortcoming is that the convergence is not guaranteed, especially in the case of unbounded solution sets. Therefore, a rigorous analysis of a resulting network must be considered [19], [3]. In [17], [18], [20] and [21], some globally convergent neural networks with non gradient methods are presented. Xia and Wong in [22] presented a globally convergent neural network that can solve problems with bounded and unbounded solution sets. However, there is no deterministic procedure to be used directly to construct neural networks for solving SILP problems. We thus feel that it is important to investigate deterministic methods such that designed neural networks are convergent and can solve SILP problems.

To solve the SILP problems, we propose in this paper, a general methodology for designing convergent neural network as an alternative to the existing solution method. It is based on local reduction method; where the numerical result shows the efficacy of the proposed method.

2. Modeling of Neural Network

According to [22], to formulate an optimization problem in term of neural network, there exist two types of methods. One approach which commonly used in developing an optimization neural network is to convert the constrained optimization problem into an associated unconstrained optimization problem, and then, designing a neural network that solves the resulted unconstrained problem with gradient methods. The other approach is to construct a set of differential equation such that their equilibrium points correspond to the desired solution, and then, finding an appropriate Lyapunov function such that all trajectories of the system converge to some equilibrium points [17], [18], [20] and [21]. By combining the above two types of methods, Xia and Wong in [22] propose a more general method. That is, a neural network is constructed by using the below procedure:

Step 1: Suppose the continuous function $\Phi: \Omega \subset \mathbb{R}^l \to \mathbb{R}$, be the objective function and l be the unknown vector dimension in unconstrained minimization problem, such that its minima corresponds to the exact or approximate solution to the main problem; here, Ω is the set of admissible region's points.

Step 2: Construct a continuous vector valued function $F: \Omega \subset \mathbb{R}^l \to \mathbb{R}^l$ such that satisfies the following two conditions:

1) If u^* is a minimizer of Φ, then $F(u^*) = 0$;
2) $(u-u^*)^{'} F(u) \leq -\alpha\big(\Phi(u) - \Phi(u^*)\big), \forall u \in \Omega$,

where $F(u)$ satisfies the local Lipschitz conditions [15], $\alpha > 0$ and fixed.

Step 3: Let the neural network model for solving the corresponding programming problem be represented by the following dynamic system:

$$\frac{du}{dt} = -\Lambda F(u), \quad u \in \Omega, \tag{1}$$

where $\Lambda = diag(\beta_1, \beta_2, \dots, \beta_l)$, and $\beta_i > 0$ which is to scale the convergence rate of (1).

Step 4: Based on the systems (1), design the neural network architecture for solving the main problem.

In the aforementioned method, the first step is to establish a computational energy function such that the lowest energy state corresponds to the desired solution. Toward this, the basic approach is to transform the constrained problem (P) to an unconstrained problem, for which we need to find some nonlinear inequalities and equations through the saddle point theorem, the Kuhn-Tucker conditions, projection theorem or some equivalent results concerning optimal solution to (P). The second step is to establish the relation between equilibrium points of the system (1) and the lowest energy state, to ensure that the network trajectory globally converges to stable equilibrium. Next coming theorem states this convergence; its proof can be found in [22].

Theorem 2.1: *Any neural network derived from the proposed method is Lyapunov stable and globally convergent to an exact or approximate solution to programming problem.*

Now we have the preliminary necessary information for designing effective neural network models to solve the Semi infinite linear programming problem (P).

3. Neural Network Model Based on the Local Reduction Method

Assume that all coefficients of $\sigma = \{a_t' x \geq b_t, t \in T\}$ belong to $C^1(T)$, with $T = \{t \in \mathbb{R}^m | V_j(t) \leq 0,\ j \in J\}$, where J is a finite set and $\{V_j :\ j \in J\} \subset C^1(\mathbb{R}^m)$.

Also suppose that the gradients of the active constraints defining T are linearly independent at every $t \in T$, and, finally, T is bounded and the Slater constraint qualification holds [4]; the function V_j's can be considered as elements of a countable base of $C^1(\mathbb{R}^m)$, such as polynomials.

According to the theorem (7-1 of [4]), x^* is the optimal point of (P) if and only if x is a feasible point and c is a non-negative linear combination of n vectors of $\{a_t, t \in T(x)\}$ (Carathéodory lemma for cone), say $a(t_k), t_k \in T(x),\ k = 1,2,\dots,n,$ where $T(x) = \{t: g(t,x) = a_t' x - b_t = 0\}$ In other word, A necessary and sufficient condition for x to be an optimal solution of (P), is the existence of $(t_1, t_2, \dots, t_n) \in T^n$ and $y \in \mathbb{R}_+^n$ such that:

$$\sum_{k=1}^{n} y_k a(t_k) = c, \tag{2}$$

$$g(t_k, x) = a_t' x - b_t = 0, \qquad k = 1,2,\dots,n, \tag{3}$$

and

$$g(t,x) \geq 0, \forall t \in T. \tag{4}$$

Obviously, taking into account (3), (4) can be replaced by the following: t_k is a global minimizer of $g(\cdot, x), k = 1,2,\dots,n.$ this implies, by the classical KKT Theorem, the existence of unique non-negative Scalars $\{\theta_j^k, j \in J\}$ such that

$$\nabla_t g(t_k, x) + \sum_{j \in J} \theta_j^k \nabla V_j(t_k) = 0_m, \qquad k = 1,2,\dots,n, \tag{5}$$

and the complementarity conditions

$$\theta_j^k V_j(t_k) = 0, k = 1,2,\dots,n, j \in J. \tag{6}$$

Therefore the existence of a solution; $t_1, t_2, \dots, t_n;\ y_1, y_2, \dots, y_n;\ \theta_j^k,\ j \in J, k = 1,2,\dots,n,$ for the nonlinear system {(2); (3); (5); (6)} is a necessary condition for x to be an optimal solution of (P). These simultaneous equations can be solved by using the Newton or quasi-Newton methods. Therefore the convergence of the local reduction method is as the same as the Newton or quasi-Newton methods [13]. Indeed, for using these methods to solve the SILP problem, it is necessary to choose the initial point near the optimal solution [4]. Also, we remind that feasibility is the stopping condition of this method such as the other methods of solving SILP problems.

The necessary and sufficient condition for x^* to be the optimal solution of (P), is that there exist the variables $k_1, k_2, \dots, k_n, x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, t_1, t_2, \dots, t_n$ such that the following conditions are satisfied:

$$\sum_{k=1}^{n} y_i a(t_i) = c;$$
$$a'(t_i) x^* = b(t_i), \qquad i = 1,2,\dots,n;$$
$$\nabla_t g(t_i, x^*) + k_i \nabla V(t_i) = 0, \qquad i = 1,2,\dots,n;$$
$$k_i V(t_i) = 0, \qquad i = 1,2,\dots,n;$$
$$y_i \geq 0\ \ k_i \geq 0, \qquad i = 1,2,\dots,n.$$

Goberna in [4] has proved that the solutions of the above simultaneous equations $y_i,\ i = 1,2,\dots,n$ are indeed the values of $\mu^*(t_i)$, where

μ^* is the dual optimal solution corresponding to (P). Thus, the objective function of dual problem (D) can be converted from integral to a summation as follow:

$$\int_T b_t d\mu = \sum_{i=1}^{n} b(t_i) y_i.$$

So, we define a new minimization problem where its objective function is the difference between the values of primal and dual objective functions:

$$Min: (c' x^* - \sum_{i=1}^{n} b(t_i) y_i)^2$$
$$S.to: \sum_{k=1}^{n} y_i a(t_i) = c,$$
$$a'(t_i) x^* = b(t_i), \qquad i = 1,2,\dots,n;$$

$$\nabla_t g(t_i, x^*) + k_i \nabla V(t_i) = 0, i = 1,2, \dots, n;$$
$$k_i V(t_i) = 0, \quad i = 1,2, \dots, n;$$
$$y_i \geq 0, k_i \geq 0, \quad i = 1,2, \dots, n;$$
$$u = (x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n, t_1, t_2, \dots, t_n, k_1, k_2, \dots, k_n)'$$

we redefine the above problem as:

$$Min: F(u)$$
$$S.to: G(u) = 0;$$
$$H(u) = 0;$$
$$L(u) = 0;$$
$$W(u) = 0;$$
$$Y \geq 0, K \geq 0;$$

Where

$$F(u) = (f_1(u), f_2(u), \dots, f_n(u))', \quad f_i(u) = (c' x^* - \sum_{i=1}^n b(t_i) y_i)^2,$$
$$G(u) = (g_1(u), g_2(u), \dots, g_n(u))', \quad g_i(u) = c_i - \sum_{j=1}^n a_i(t_j) y_i,$$
$$H(u) = (h_1(u), h_2(u), \dots, h_n(u))', \quad h_i(u) = a'(t_i) x^* - b(t_i),$$
$$L(u) = (l_1(u), l_2(u), \dots, l_n(u))', \quad l_i(u) = \nabla_t g(t_i, x^*) + k_i \nabla V(t_i),$$
$$W(u) = (w_1(u), w_2(u), \dots, w_n(u))', \quad w_i(u) = k_i V(t_i),$$
$$Y = (y_1, y_2, \dots, y_n), \quad K = (k_1, k_2, \dots, k_n).$$

By using the penalty method, constrained optimization problem is converted to an unconstrained one as follow:

$$E(u) = F(u) + \frac{s}{2}\left\{\sum_{i=1}^n (g_i(u))^2 + \sum_{i=1}^n (h_i(u))^2 + \sum_{i=1}^n (l_i(u))^2 + \sum_{i=1}^n (w_i(u))^2 + \frac{1}{2}K'(K - |k|) + \frac{1}{2}Y'(Y - |Y|)\right\}$$

where s is the penalty parameter. We remind that the two last terms of the right hand side are applied to guarantee that $Y \geq 0$ and $K \geq 0$. Now, the gradient method can be employed to convert the solution of above minimization problem to the solution of the following ordinary differential equations system:

$$\frac{du}{dt} = -\Lambda\{\Delta f(u) + s[\Delta g(u) g(u) + \Delta h(u) h(u) + \Delta l(u) l(u) + \Delta w(u) w(u) + 2K^- + 2Y^-]\} \quad (7)$$

where u$\in \mathbb{R}^{4n}$, $Y \geq 0$, $K \geq 0$ and Λ was introduced in (1); here K^- and Y^- are the negative components of K and Y respectively.

The convergence and stability of the above model that is known as Chau and Kennedy neural network model, is proved by the following theorem in [22].

Theorem 3.1: *Assume that the functions W, L, H, G and F are twice continuously differentiable, then the proposed neural network is Lyapunov stable and at least locally convergent to the optimal solution of (P). Moreover, if the function E is convex on* $\mathbb{R}^{4n}$*, the convergence will be global.*

Example 3.1. Consider the following problem from Goberna [4]:

$$Min: \ 2x_1 + x_2$$
$$S.to: \ tx_1 + (1-t)x_2 \geq t - t^2 \ \ \forall t \in [0,1];$$

The dual problem can be shown as:

$$Sup: \ \int_0^1 t - t^2 \, d\mu$$
$$S.to: \ \int_0^1 t \, d\mu = 2;$$
$$\int_0^1 1 - t \, d\mu = 1;$$
$$\mu \in \mathcal{M}^+(T), \ \mu \geq 0.$$

We have solved this problem by using the proposed method. Simultaneous equations have been solved based on the Euler method and by using MATLAB 7.6.

The step length and number of iteration have been considered as 0.0001 and 1000 respectively. The obtained solution was $x^* = (0.1003, 0.4651)$ and $\lambda^* = 2.9270$ in $t = 0.6291$ where the accurate solution of the problem is $x^* = (0.\bar{1}, 0.\bar{4})$ and $\lambda = 3$ in $t = 0.\bar{6}$.

4. Conclusion

Based on the KKT optimality conditions for the primal problem, a model of neural network is achieved to solve SILP problems. In this model, the problem is converted to a nonlinear optimization problem, and then it is solved by the Chau and Kennedy neural network model. According to this model, the obtained solution is approximately convergent to the optimal solution. In comparison to the known solution methods of SILP problems like cutting plane algorithms, primal and dual exchange methods and etc, this approach has much simpler structure and friendly users. Therefore, it causes to have less algorithm complexity and less time consuming. Moreover, the numerical result shows the effectiveness of the proposed method.

References.

[1] A. Bouzerdoum and T.R. Pattison, Neural networks for quadratic optimization with bound constraints, IEEE Trans. Neural Networks, 4 (1993), 293-304.

[2] A.V. Fiacco and K. O. Kortanek, Semi-Infinite programming and Application: An Introdction. Symposium, Austin, Texas, September 8-10, 1981.

[3] M.P. Glazos, S. Hui and S.H. Zak, Sliding models in solving convex programming problems, SIAM J. Contr. Optimization, 36 (1998), 680-690.

[4] M.A. Goberna and M.A. Lopez, Linear Semi-infinite Optimization, Alicant University, (1998).

[5] J.J. Hopfield, and D.W. Tunk, Neural computation of decisions in optimization problems, Biol. Cybern., 52 (1985), 141-152.

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

[7] T. Leo'n, S. Sanmatias and E. Vercher, On the numerical treatment of linearly constrained semi-infinite optimization problems, European Journal of Operations Research, 121 (2000), 78-91.

[8] P. Mateede, Application of khobotov's algorithm to variational inequalities and network equilibrium problems, Inform. Syst. Oper. Res., 29 (1991), 258-270.

[9] J.J. Mote and G. Toroaldo, On the solution of large quadratic programming problems with bound constraints, SIAM J. Optimization, 1 (1991), 93-113.

[10] J.S. Pang, A posteriori error bounds for the linearly-constrained variational inequality problem, Math. Operation Res., 12 (1987), 474-484.

[11] R. Reemtsen, and J. Ruckmann, Nonconvex Optimization and Its Application: Semi-Infinite Programming, KLUWER ACADEMIC PUBLISHERS London.

[12] A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J.I. Huertas, and E. Sanches-Sinencio, Nonlinear switched-capacitor neural networks for optimization problems, IEEE Trans. Circuits Syst., 37 (1990), 384-397.

[13] W. Sun, and Y. Yuan, Optimization Theory And Methods: Nonlinear Programming, Springer (2006).

[14] 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., CAS-33 (1986), 533-541.
[15] W. Walter, Ordinary Differential Equations , Graduate text in mathematics, Springer, (1998).
[16] M. Wang, Y. E. Kuo, A perturbation method for solving linear semi-infinite programming problems, Computers and Mathematics with Applications, 37 (1999), 181-198.
[17] X. Wu, J. Xia, J. Li, and W.K. Chen, A high performance neural network for solving linear and quadratic programming problems, IEEE Trans. Neural Networks, 7 (1996), 643-651.
[18] Y. Xia, and Wang, J., Neural network for solving linear programming problems with bounded variables, IEEE Trans. Neural Networks, 6 (1995), 515-519.
[19] Y. Xia, A new Neural network for solving linear programming problems and its applications, IEEE Trans. Neural Networks, 7 (1996), 525-529.
[20] Y. Xia, Anew Neural network for solving linear and quadratic programming problems, IEEE Trans. Neural Networks, 7 (1996), 1544-1547.
[21] Y. Xia, Neural network for solving extended linear programming problems, IEEE Trans. Neural Networks, 8 (1997), 519-525.
[22] Y. Xia, A general methodology for designing globally convergent optimization neural networks, IEEE Trans. Neural Networks, 9 (1998), 1331-1343.