Control and Optimization in Applied Mathematics (COAM)
Vol. 1, No. 1, Spring-Summer 2016(55-67), ©2016 Payame Noor University, Iran

Solving Linear Semi-Infinite Programming Problems using Recurrent Neural Networks

A. Malek[1], G. Ahmadi[2]* and S. M. M. Alizamini[3]

[1] Department of Applied Mathematics, Faculty of Mathematical Sciences, Tarbiat Modares University, P.O. Box. 14115-134, Tehran, Iran
[2,3] Department of Mathematics, Payame Noor University, P.O. Box. 19395-3697, Tehran, Iran.

Received: January 5, 2014; Accepted: June 14, 2014.

Abstract. Linear semi-infinite programming problem is an important class of optimization problems which deals with infinite constraints. In this paper, to solve this problem, we combine a discretization method and a neural network method. By a simple discretization of the infinite constraints,we convert the linear semi-infinite programming problem into linear programming problem. Then, we use a recurrent neural network model, with a simple structure based on a dynamical system to solve this problem. The portfolio selection problem and some other numerical examples are solved to evaluate the effectiveness of the presented model.

Keywords. Linear semi-infinite programming, Recurrent neural network, Dynamical system, Discretization, Linear programming.
MSC. 90C34.

* Corresponding author
mala@modares.ac.ir, g.ahmadi@pnu.ac.ir, seyyedmehdi-mirhosseini@yahoo.com
http://mathco.journals.pnu.ac.ir

Archive of SID

www.SID.ir

1 Introduction

A *linear semi-infinite program* (LSIP) is an optimization problem with finitely variables $x = (x_1,\ x_2, \cdots,\ x_n) \in R^n$ on a feasible set described by infinitely many constraints:

$$\begin{aligned} P: \quad & \max_x z \ = c^T x \\ s.t \quad & Ax \le b, \\ & K(t)\, x \le u(t)\,, \\ & x \ge 0, \qquad t \in T, \end{aligned}$$

where T is an arbitrary infinite set, the vector $c \in R^n$, $b \in R^p$ and the matrix $A \in R^{p\times n}$ are arbitrary. $K(t)$ is a $q \times n$ matrix in variable t and $u(t)$ is a $q \times 1$ vector in variable t. By F we denote the *feasible set* of P, whereas $v := sup\{c^T x | x \in F\}$ is the *optimal value*, and $S := \{x \in F\ |c^T\ x = v\}$ is the *optimal set* or the set of *minimizers* of the problem. We say that P is *feasible* or *consistent* if $F = \emptyset$, and set $v = +\infty$ when $F = \emptyset$.

There are many applications of SIP in different fields such as Chebyshev and reverse approximation [6], robotics [18], mathematical physics [10], engineering designs [6], optimal control [18], transportation problems [10], fuzzy sets [10], cooperative games [10], robust optimization [16], statistics, economics [14], etc.

Various numerical procedures have been presented over decades for solving semi-infinite programming problems. Ferris and Philpott ([5]) proposed an interior point algorithm for SIP. Anderson and Levis ([3]) proposed an extension of the simplex algorithm for LSIP. Hettich and Kortanek ([9]) proposed a discretization method to solve SIP. Also Reemtsen and Gorner ([13]) extends the numerical methods in Hettich and Kortanek ([9]). Lars Abbe ([1]), built a method based on a stable approach for nonlinear SIP like penalty methods. Reemtsen and Gorner's survey ([13]) reviews the most recent advances and uses of SQP, including a trust region variation. One of the most recently published interior point methods appears in Stein and Still ([14]). Since the computing time required for solving SIP and other optimization problems depends on the dimension and the structure of the problem, the conventional numerical methods are usually less effective in real-time applications such as robotics. One promising approach to handle online applications is to employ recurrent neural networks based on circuit implementation. The essence of neural optimization lies in its dynamic nature for optimization and the availability of electronic implementation. Unlike other parallel algorithms, neural networks can be implemented physically by designated hardware.

Tank and Hopfield ([15]) first proposed a neural network for linear programming that was mapped onto a closed-loop circuit. Their work has inspired many researchers to develop other neural networks for solving different classes of optimization problems such as linear, nonlinear, quadratic programming, convex and pseudo-convex optimization, complementarity and variational inequality problems ([2, 4, 11, 19, 20] and the references therein). As mentioned above, in many real-world optimization problems, one has to deal with linear semi-infinite programming problems. The objective of this paper is

Archive of SID

www.SID.ir

to use recurrent neural networks for solving LSIP. Malek and Yari ([11]) proposed an effective recurrent neural network model to solve linear programming problems. Here, using discretization method, we reduce the LSIP into a linear programming problem. Then, using a recurrent neural network model based on the model proposed by Malek and Yari, we solve the reduced problem. simulation results show that this method is globally convergent and the speed of convergence is high. This paper is organized as follows: In Section 2, the discretization method to solve SIP is discussed. In Section 3, the linear semi-infinite programming and the discretized form of it will be described. In Section 4, a recurrent neural network model, based on Malek and Yari model ([11]) is proposed. In Section 5, some numerical examples of the proposed method are discussed to evaluate the effectiveness of the proposed neural network.

2 Preliminaries

This section deals with some definitions and notions about ordinary discretization method and dual of LSIP. consider the problem of approximating the feasible set

$$F = \{x|Ax \le b,\ K(t)x \le u(t),\ t \in T\}$$

of our linear semi-infinite problem P by imposingonly finitely many constraints. The simplest way is through ordinary discretization. Choose $\overline{T} \subset T$, where $|\overline{T}| < \infty$, and replace F by

$$F(\overline{T}) = \{x|Ax \le b,\ K(t)x \le u\,(t)\,, t \in \overline{T}\}$$

and consider the approximation problem

$$\begin{aligned} P(\overline{T}):\quad & \max_x z \ = c^T x \\ s.t \quad & Ax \le b, \\ & K\,(t)\,x \le u\,(t)\,, \\ & t \in\ \overline{T},\quad x \ge 0, \end{aligned}$$

typically $\overline{T}$ is termed a grid.

Definition 1 (See [8]). Let $\overline{T} = \{t_1, t_2, \ldots, t_r\}$ be a grid of T. Denote by $k(\overline{T})$ its convex hull. Next determine r continuous functions $\varphi_1,\ \varphi_2,\ \ldots\ ,\ \varphi_r$ such that

(i) each $t \in (\overline{T})$ has representation $t = \sum_{j=1}^{r} \varphi_j(t) t_j$,

(ii) $\sum_{j=1}^{r} \varphi_j\,(t) = 1$,

(iii) $\varphi_j\,(t) \ge 0,\quad j = 1,\ 2,\ \ldots\ ,\ r$,

(iv) for each $t \in (\overline{T})$ at most $k+1$, $\varphi_j\,(t)$ are strictly positive.

Archive of SID

www.SID.ir

Let now, a function φ be defined on T. Define $L\varphi$ through

$$(L\varphi)(t) \ = \sum_{j=1}^{r} \varphi_j\,(t),$$

then L will be called the positive linear interpolator induced by T and $\varphi_1, \varphi_2, \dots\ , \varphi_r$.

Theorem 1 (See [8]). *Let T be a compact set. Then, there is a finite subset $\overline{T} \subset T, \overline{T} = \{t_1, t_2, \dots, t_r\}$ such that P is computationally equivalent to the task: Minimize the linear form $c^T x$ over all vectors $x \in R^n$subject to the constraints*

$$Ax \le b, \ \ LK(t)x \le Lu(t), \ t \in T,$$

where L is defined as in definition 1.

As a consequence of Theorem 1, an LSIP could be replaced by a linear programming problem.

Associated with P, different dual problems can be defined. For instance the so called Haar dual can be defined in the following manner:

$$\begin{aligned} D: \quad & \min_y w \ = b^T y + \sum_{t \in T} u(t)\lambda(t) \\ s.t \quad & A^T y \ + K(t)^T \lambda(t) \ge c, \\ & y \ge 0, \\ & \lambda(t) \ge 0, \ \ t \in T, \end{aligned}$$

where we allow only for a finite number of the dual variables, $\lambda(t), \ t \in T$ to take positive values. By v_D we denote the optimal value of D. Choose $\overline{T} = \{t_1, t_2, \dots, t_r\} \subset T$ and replace the problem P by $P(\overline{T})$:

$$\begin{aligned} P\left(\overline{T}\right): \quad & \max_x z \ = c^T x \\ s.t \quad & Ax \le b, \\ & K\left(t_i\right) x \le u\left(t\right), \\ & i = 1, 2, \dots, r\ , \ x \ge 0, \end{aligned}$$

Now, we have a finite linear programming problem with $p + rq$ constraints. The dual problem corresponding to P is given by

$$\begin{aligned} D: \quad & \min_x w \ = b^T y + \sum_{t \in T} u(t)\lambda(t) \\ s.t \quad & A^T y \ + K(t_i)^T \lambda(t_i) \ge c, \\ & y \ge 0, \\ & \lambda\left(t_i\right) \ge 0, \ \ i = 1, 2, \dots, r. \end{aligned}$$

Archive of SID

2.1 A recurrent neural network model

Let us to suppose that in P

$$\mathbf{X}{=}\mathbf{x},\ \ \mathbf{C}{=}\mathbf{c},\quad \mathbf{A}{=}\begin{pmatrix} A \\ K(t_1) \\ K(t_2) \\ \vdots \\ K(t_r) \end{pmatrix},$$

$$\mathbf{B}{=}\begin{pmatrix} A \\ u(t_1) \\ u(t_2) \\ \vdots \\ u(t_r) \end{pmatrix},\quad \mathbf{Y}{=}\begin{pmatrix} A \\ \lambda(t_1) \\ \lambda(t_2) \\ \vdots \\ \lambda(t_r) \end{pmatrix}. \tag{1}$$

So, instead of P and D, we have the following problems:

$$\begin{aligned} P:\ & \max_{\mathbf{X}} z\ = \mathbf{C}^T\mathbf{X} \\ s.t\ & \mathbf{A}\mathbf{X} \leq \mathbf{B}, \\ & \mathbf{X} \geq 0, \end{aligned}$$

The dual problem D of P is

$$\begin{aligned} D:\ & \min_{\mathbf{Y}} w\ = \mathbf{B}^{\mathbf{T}}\mathbf{Y} \\ s.t\ & \mathbf{A}^T\mathbf{Y}\ \geq \mathbf{C}, \\ & \mathbf{Y} \geq 0, \end{aligned}$$

recurrent neural network model proposed by Malek and Yari (See [11]), based on a non-linear dynamical system for solving linear programming problem. Here, we describe this dynamical system, using an economic problem.

A company produces n different type of products P_1, P_2, ..., P_n.The value of producing P_i is about $\mathbf{C}_i$. These products are made from m different type of resources R_1, R_2, ..., R_m of limited supply. The amount of the resource R_j is about $\mathbf{B}_j$. Let $\mathbf{X_i}$ show the amount of product P_i where $1 \leq i \leq n$. The LP problem is to maximize the value of products $\mathbf{C}^T\mathbf{X}$, under the resources $\mathbf{AX} \leq \mathbf{B}$, where $\mathbf{B}{=}(\mathbf{B}_1\ ,\ \mathbf{B}_2,\ ...,\ \mathbf{B}_n)$, $\mathbf{X}{=}(\mathbf{X}_1,\ \mathbf{X}_2,\ ...,\ \mathbf{X}_n)$, $\mathbf{C}{=}(\mathbf{C}_1,\ \mathbf{C}_2,\ ...,\ \mathbf{C}_m)$, $\mathbf{A} = (a_{ij})$, a_{ij} is the amount of resource R_j needed to produce the product P_i and is the amount of resources needed to make one unit of each product. Let $\mathbf{Y}_j$ show the price of resource R_j where $1 \leq j \leq m$. The strategy of the producer is The rate of increasing in the amount of each product is proportional with profitability of that product ($d\mathbf{X}/dt$= value minus cost= $\mathbf{C}{-}\mathbf{A}^T\mathbf{Y}$). Similarly, the resource owner's strategy is:

Archive of SID

The price of a resource is based on the supply and demand law (demand minus supply= $\mathbf{AX}-\mathbf{B}$). Rate of changes in price of resource each resource is proportional to demand for it ($d\mathbf{Y}/dt = \mathbf{AX}-\mathbf{B}$).

To solve the problem P, let us to describe the dynamics of this problem by the following dynamical system:

$$\begin{cases} \dot{\mathbf{X}}\left(s\right)=\mathbf{C}-\mathbf{A}^{T}\mathbf{Y}\left(s\right) \\ \\ \dot{\mathbf{Y}}\left(s\right)=\mathbf{AX}\left(s\right)-\mathbf{B} \end{cases} \tag{2}$$

where $\mathbf{X}\left(s\right)\geq 0$, $\mathbf{Y}\left(s\right)\geq 0$, $s\geq 0$ and $\left(\mathbf{X}\left(s\right),\ \mathbf{Y}\left(s\right)\right)^{T}$ is a state vector. Consider a recurrent neural network with two layers, that in which, each primal neuron plays the role of producer of a given type of products and each dual neuron plays the role of a resource owner selling a resource. Based on dynamical system (2), Malek and Yari in [11] proposed the following recurrent neural network to solve P:

$$\begin{cases} \frac{d\mathbf{X}}{ds}=\mathbf{C}-\mathbf{A}^{T}\left(\mathbf{Y}+\eta_{1}\frac{d\mathbf{Y}}{ds}\right) \\ \\ \frac{d\mathbf{Y}}{ds}=\mathbf{A}\left(\mathbf{X}+\eta_{2}\frac{d\mathbf{X}}{ds}\right)-\mathbf{B} \end{cases} \tag{3}$$

where $\mathbf{X}=\mathbf{X}\left(s\right)\geq 0$ and $\mathbf{Y}=\mathbf{Y}\left(s\right)\geq 0$ and η_1 and η_2 are some learning rates. The term $\eta_1\frac{d\mathbf{Y}}{ds}$ signifies the fact that the producer not only use the current resource : prices in their cost calculation, but also take into account the trend of this prices. Like the producers, the resource owners base their decisions not only on the current demands $\mathbf{AX}$, but also on the trend of this demands $\mathbf{A}\frac{d\mathbf{X}}{ds}$.

Theorem 2. *If the recurrent neural network whose dynamics is described by the differential equations (2) in (3) converges to a stable state, then the convergence will be the optimal solutions for FLP problem and is DFLP problem.*

Proof. Let $\mathbf{X}_i$ be the i-th element of $\mathbf{X}$, and $\mathbf{Y_i}$ be the i-th element of $\mathbf{Y}$. Instead of the first equation of (3), we can write the following two equations:

$$\begin{cases} \left[\frac{d\mathbf{X}}{ds}\right]_i=-\left[\mathbf{A}^{T}\left(\mathbf{Y}+\eta_{1}\frac{d\mathbf{Y}}{ds}\right)\right]_i+\left[\mathbf{C}\right]_{\mathbf{i}}, & \mathbf{X}_i>0 \\ \\ \left[\frac{d\mathbf{X}}{ds}\right]_i=\max\left\{\left[\mathbf{C}-\mathbf{A}^{T}\left(\mathbf{Y}+\eta_{1}\frac{d\mathbf{Y}}{ds}\right)\right]_i,0\right\}, & \mathbf{X}_i=0. \end{cases} \tag{4}$$

Since $\mathbf{X}_i\geq 0$, the second equation of (4) is required. Instead of the second equation of (3), we can write the following two equations:

$$\begin{cases} \left[\frac{d\mathbf{Y}}{ds}\right]_j=-\left[\mathbf{B}\right]_{\mathbf{j}}+\left[\mathbf{A}\left(\mathbf{X}+\eta_{2}\frac{d\mathbf{X}}{ds}\right)\right]_j, & \mathbf{Y}_j>0 \\ \\ \left[\frac{d\mathbf{Y}}{ds}\right]_j=\max\left\{\left[\mathbf{A}\left(\mathbf{X}+\eta_{2}\frac{d\mathbf{X}}{ds}\right)-\mathbf{B}\right]_j,0\right\}, & \mathbf{Y_j}=0. \end{cases} \tag{5}$$

Since $\mathbf{Y_j}\geq 0$, the second equation of (5) is required. Suppose that $\mathbf{X}^*$ and $\mathbf{Y}^*$ are stable states for $\mathbf{X}$ and $\mathbf{Y}$, i.e., $\lim_{t\to\in fty}\mathbf{X}\ =\mathbf{X}^*$ and $\lim_{s\to\in fty}\mathbf{Y}\ =\mathbf{Y}^*$. Because of convergence of these states,

Archive of SID

www.SID.ir

$$\frac{d\mathbf{X}^*}{ds} = \frac{d\mathbf{Y}^*}{ds} = 0.$$

Therefore, we have

$$\begin{cases} \left[\mathbf{A}^T\mathbf{Y}^*\right]_i = [\mathbf{C}]_\mathbf{i},\ \mathbf{X}^*{}_i > 0 \\ \max\left\{\left[\mathbf{C} - \mathbf{A}^T\mathbf{Y}^*\right]_i, 0\right\} = 0\ ,\ \mathbf{X}^*{}_i = 0 \end{cases} \tag{6}$$

and

$$\begin{cases} [\mathbf{A}\mathbf{X}^*]_j = [\mathbf{B}]_\mathbf{j},\ \mathbf{Y}^*{}_j > 0 \\ \max\left\{[\mathbf{A}\mathbf{X}^* - \mathbf{B}]_j, 0\right\} = 0,\quad \mathbf{Y}^*{}_\mathbf{j} = 0 \end{cases} \tag{7}$$

or

$$\begin{cases} [\mathbf{C}]_\mathbf{i} = \left[\mathbf{A}^T\mathbf{Y}^*\right]_i,\ \ \mathbf{X}^*{}_i > 0 \\ [\mathbf{C}]_\mathbf{i} \le \left[\mathbf{A}^T\mathbf{Y}^*\right]_i,\ \ \mathbf{X}^*{}_i = 0 \end{cases} \tag{8}$$

and

$$\begin{cases} [\mathbf{A}\mathbf{X}^*]_j = [\mathbf{B}]_\mathbf{j},\ \ \mathbf{Y}^*{}_j > 0 \\ [\mathbf{A}\mathbf{X}^*]_j \le [\mathbf{B}]_\mathbf{j},\qquad \mathbf{Y}^*{}_\mathbf{j} = 0. \end{cases} \tag{9}$$

Thus,

$$\mathbf{C} \le \mathbf{A}^T\mathbf{Y}^*, \mathbf{A}\mathbf{X}^* \le \mathbf{B}.$$

Therefore, $\mathbf{X}^*$ and $\mathbf{Y}^*$ are the feasible solution for the problems P and D. Furthermore, from (8) we have

$$\mathbf{C}^T\mathbf{X}^* = \mathbf{Y}^{*\mathbf{T}}\mathbf{A}\mathbf{X}^*, \tag{10}$$

and from (9) we have

$$\mathbf{B}^T\mathbf{Y}^* = \mathbf{X}^{*\mathbf{T}}\mathbf{A}^T\mathbf{Y}^*. \tag{11}$$

From the equations (10) and (11),

$$\mathbf{B}^T\mathbf{Y}^* = \mathbf{X}^{*\mathbf{T}}\mathbf{A}^T\mathbf{Y}^* = \mathbf{Y}^{*\mathbf{T}}\mathbf{A}\mathbf{X}^* = \mathbf{C}^T\mathbf{X}^*.$$

According to complementary slackness theorem, we can conclude that $\mathbf{X}^*$ and $\mathbf{Y}^*$are the optimal solutions for the problems P and D.

□

Archive of SID

www.SID.ir

3 Numerical Examples

In this section some illustrative examples are solved to demonstrate the effectiveness of the proposed recurrent neural network model. The software MATLAB is used to make this simulations.

Example 1. Consider the following linear semi-infinite programming problem:

$$\begin{aligned} P\ :\quad \max z\ &= -2x_1 - x_2 \\ s.t:\ -tx_1 + (t-1)\,x_2 &\le t^2 - t, \\ t \in [0,1]\,,\quad & x \ge 0. \end{aligned}$$

This problem has an optimal solution $z = 0.666$. The numerical results obtained for different discretization and 500 iterations are given in Table 1. In this table, m is the number of constraints which we put instead of an infinite constraint in our discretization.

Figure 1 displays the behavior of the proposed model, when it is applied for this example. according to Figure 1, it is obvious that after some iterations z and w tends the optimal value 0.666. By attention to Table 1, It is clear that by refinance of discretization, we receive a better approximation.

Table 1: Optimal value of the LSIP in Example 1 for 500 iterations.

	$m = 11$	$m = 101$	$m = 1001$	$m = 10001$
z	0.6512	0.6629	0.6659	0.6665
w	0.6564	0.6642	0.6668	0.6665
$\|z-w\|$	0.0130	0.0011	0.0009	0.0000

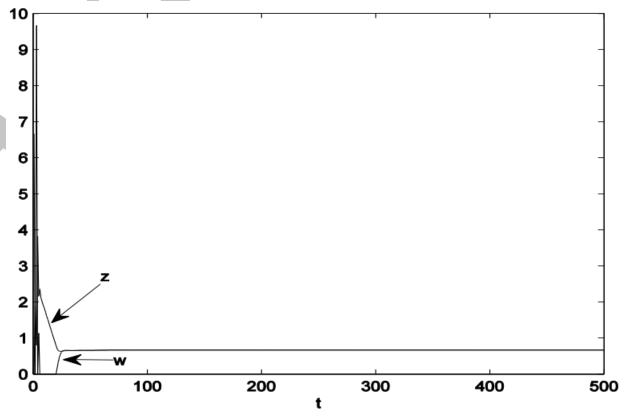

Figure 1: Optimal value of the LSIP in Example 1

Example 2. Consider the following linear semi-infinite programming problem:

Archive of SID

$$\begin{aligned} P\ :\quad & \max z\ = x_1 + \frac{1}{2}x_2 + \left(\frac{1}{2} + \frac{\varepsilon}{3}\right) x_3 \\ s.t:\quad & x_1 + tx_2 + \left(t + \varepsilon t^2\right) x_3 \le e^t, \\ & t \in [0,1]\,,\quad x_i \ge 0, \quad i = 1,2,3\ . \end{aligned}$$

This problem has an optimal solution $v\ = (1/2)(1\ +\ e)$, when $\varepsilon = 0$, and has an optimal solution $v = (1/4)(3e^3 + e)$, when $\varepsilon \neq 0$. The numerical results obtained for different discretizations, 10000 iterations and $\varepsilon = 1$ are given in Table 2.

Table 2: Optimal value of the LSIP in Example 1 for 500 iterations.

	z	w	x_1	x_2	x_3
$m = 6$	1.7260	1.7256	1.0182	0.0000	0.8494
$m = 11$	1.7264	1.7265	1.0181	0.0000	0.8500
$m = 16$	1.7263	1.7264	1.0177	0.0000	0.8504

In this table, m is the number of constraints which we put instead of an infinite constraint in our discretization. Figure 2 displays the behavior of the proposed model, when it is applied for this example. according to Figure 2, we see that after a great number of iterations, z and w tends to the optimal value

$$v\ = (\frac{1}{4})\left(3\exp\left(\frac{1}{3} + e\right)\right) 1.7263.$$

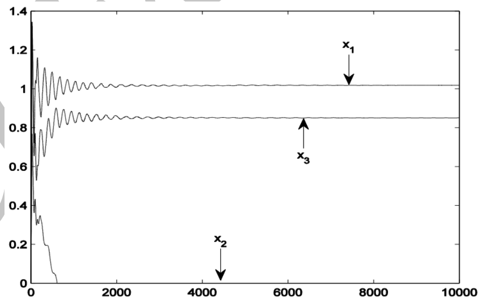

Figure 2: Optimal solution of the LSIP in Example 2 for $\varepsilon = 1,\ m = 16$ and 10000 iterations.

By attention to Table 2, it is clear that by refinance of discretization or more iterations, we receive a better approximation. Figure 3 illustrates optimal value ofthe LSIP in Example 2 for $\varepsilon = 1,\ m = 100$ and 1000 iterations. Figure 4 illustrates optimal solution of the LSIP in Example 2 for $\varepsilon = 0$, m =100 and 1000 iterations.

Example 3 (Stein & Still, 2003)**.** Let 1 euro be invested in a portfolio comprised of N shares. At the end of a given period the return per 1 Euro invested in share i is $t_i > 0$. Our goal is to determine the amount x_i to be invested in share i, $i = 1,\ 2,\ \ldots\ ,\ N$, so as to maximize the end-of-period portfolio value $z = t^T x$.

Archive of SID

www.SID.ir

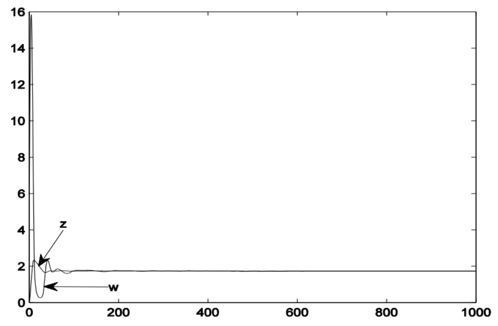

Figure 3: Optimal value of the LSIP in Example 2 for $\varepsilon = 1,\ m = 100$ and 1000 iterations.

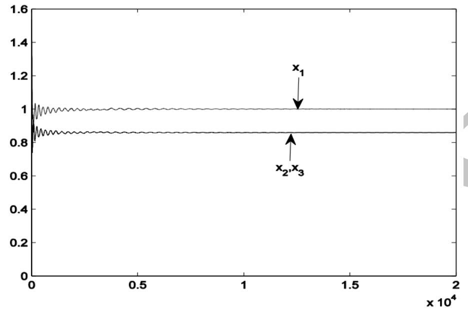

Figure 4: Optimal solution of the LSIP in Example 2 for $\varepsilon = 0,\ m = 100$ and 20000 iterations.

Archive of SID

If the vector t was certain, the solution of this optimization problem would be evident. In this case we could invest all the money into the share with maximal value t_i. A more realistic assumption is that t varies in some nonempty compact set $T \subset R^N$. Upon moving the objective function to the constraints set, we obtain the following linear semi infinite optimization problem with $n = N + 1$ and $m = N$:

$$\begin{aligned} P\ :&\quad \max z\ = x_{N+1}\\ s.t:\ &\sum_{i=1}^{N} x_i = 1,\\ &x_{N+1} - t^T x \le 0\\ &t \in T,\quad x \ge 0, \end{aligned}$$

Because of our assumption that the problem must be in standard form, instead of equality constraint $\sum_{i=1}^{N} x_i = 1$, we put two inequalities $\sum_{i=1}^{N} x_i \le 1$ and $-\sum_{i=1}^{N} x_i \le -1$. Suppose that T has the form

$$T = \left\{ t \in R^N | \sum_{i=1}^{N} \frac{(t_i - \bar{t}_i)^2}{\sigma_i^2} \le \theta^2 \right\},$$

www.SID.ir

where $\bar{t}_i$ is some "nominal" value of t_i, σ_i is a scaling parameter, $i = 1,\ 2,\ \dots,\ N$ and θ measures the risk a version of the decision maker. With the particular choices

$$\begin{aligned} t_i &= \ 1.15\ +\ i.\frac{0.05}{N}N,\ i = 1, 2, \dots, N,\\ \sigma_i &= \frac{0.05}{N}\sqrt{2N\ \ (N\ +\ 1)\,i},\\ i\ &=\ 1,\ 2,\ \dots\ ,\ N,\ \ \theta = 1.5, \end{aligned}$$

one can show that the optimal policy is to invest equally in all shares, i.e., $x_i = 1/N$ with optimal value 1.15. If we apply the method for $N = 4, m = 11$ and 40000 iterations, we attain the optimal value $w = 1.15000$, $z = 1.15000$ and $x_i = 0.250000$, $i = 1,\ 2,\ \dots\ ,\ 10$, which is illustrated in Figure 5. If we apply the method for $N = 10$, $m = 11$ and 40000 iterations we attain the optimal value $w = 1.1500054$, $z = 1.149997$ and $x_i =\ 0.099999$, which is illustrated in Figure 6. Table 3 shows the behavior of the proposed model, when it is applied for this problem with $N = 4, 10, 20, 50, 100$ and 150.

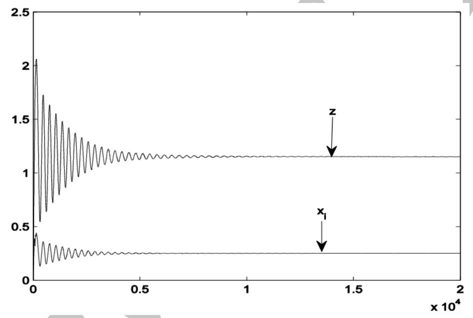

Figure 5: Optimal solution of the portfolio problem in Example 3 for $N = 4$, $m = 11$ and 20000 iterations.

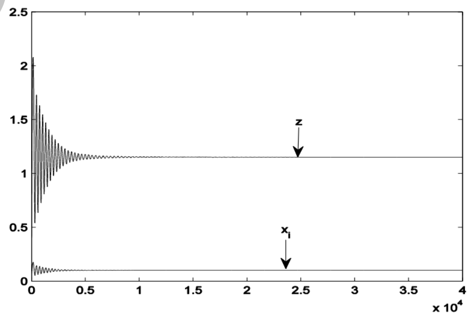

Figure 6: Optimal solution of the portfolio problem in Example 3 for $N = 10$, $m = 11$ and 40000 iterations.

Archive of SID

www.SID.ir

Table 3: Optimal solution for the portfolio problem in Example 3

N	**Opt. value**	**Approx. opt. value**	x_i	m	**Iterations**	**CPU time**
4	1.15	1.1507	0.2501	11	20000	3.1668
10	1.15	1.1501	0.1000	11	20000	8.2369
20	1.15	1.1504	0.0500	21	16000	4.3524
50	1.15	1.1519	0.02000	11	16000	4.4148
100	1.15	1.1501	0.0100	6	20000	8.3929
150	1.15	1.1500	0.0067	4	20000	8.1413

References

[1] Abbe L. (2001). " Two logarithmic barrier methods for convex semi-infinite programming ", In M. A. Goberna and M. A. Lopez, (editors) Semi-infinite Programming Recent Advances, Nonconvex Optimization and Its Applications 57, 169-195.

[2] Alipour M., Rostamy D., Malek, A. (2011). " A recurrent neural network for nonlinear convex optimization with application to a class of variational inequalities problems ", Australian Journal of Basic and Applied Sciences, 5, 5, 894-909.

[3] Anderson E. J., Levis A. S. (1989). " An extension of the simplex algorithm for semi-infinite linear programming ", Mathematical Programming 44, 247-269.

[4] Chen Y. H., Fang S. C. (1998). " Solving convex programming with equality constraints by neural networks ", Computers & Mathematics with Applications, 36, 41-68.

[5] Ferris M. C., Philpott A., B. (1989). " An interior point algorithm for semi-infinite linear programming ", Mathematical Programming 43, 257-276.

[6] Floudas C. A., Stein O. (2007). " The adaptive convexification algorithm: A feasible point method for semi-infinite programming ", SIAM Journal on Optimization, 18, 4, 1187-1208.

[7] Goberna M. A. (2007). " Semi-infinite programming ", European Journal of Operational Research, 180, 491-518.

[8] Gustafson S. (1979). " On numerical analysis in semi-infinite programming ", In Semi-Infinite Programming (Lecture Notes in Control and Information Sciences), 15, 51-65.

[9] Hettich R., Kortanek K. O. (1993). " Semi-infinite programming: Theory, methods and applications ", SIAM Review, 35, 3, 380-429.

[10] Lopez M., Still G. (2007). " Semi-infinite programming ", European Journal of Operational Research, 180, 491-518.

[11] Malek A., Yari A.(2005). " Primal dual solution for the linear programming problems using neural networks ", Applied Mathematics and Computation, 167, 198-211.

Archive of SID

www.SID.ir

[12] Malek A., Yashtini M. (2010). " Image fusion algorithms for color and gray level images based on LCLS method and novel artificial neural network ", Neuro computing, 73, 937-943.

[13] Reemsten R., Gorner S. (1998). " Numerical methods for semi-infinite programming: A survey ",In R. Reemsten and J. Rueckmann (editors), Semi-infinite Programming, Non-convex Optimization and Its Applications, 25, 101, 195-275.

[14] Stein O., Still G. (2003). " Solving semi-infinite optimization problems with interior point techniques ", SIAM Journal on Control and Optimization, 42, 3, 769-788.

[15] Tank D. W., Hopfield J. J. (1986). " Simple neural optimization network: An A/D convertor, signal decision circuit, and a linear programming circuit ", IEEE Transactions on Circuits and Systems, 35, 533-541.

[16] Vazquez F. G., Ruckmann J. J., Stein O., Still G. (2008). " Generalized semi-infinite programming: A tutorial ", Journal of Computational and Applied Mathematics 217, 394-419.

[17] Vaz F. I. A., Fernandes E. M. G. P., Gomes M. P. S. F. (2003). " Robot trajectory planning with semi-infinite programming ", European Journal of Operational Research, 153, 607–617.

[18] Vaz F. I. A., Ferreira E. C. (2009). " Air pollution control with semi-infinite programming ", Applied Mathematical Modelling 33, 1957-1969.

[19] Xia Y., Leung H., Wang J. (2002) . " A projection neural network and its application to constrained optimization problems ", IEEE Transactions on Circuits and Systems—I: Fundamental theory and applications, 49, 4, 447-458.

[20] Xia Y., Wang J. (2004). " A recurrent neural network for nonlinear convex optimization subject to nonlinear inequality constraints", IEEE Transactions on Circuits and Systems—I: Regular Papers, 51, 7, 1385-1394.

Archive of SID

www.SID.ir

حل مسائل برنامه‌ریزی نیمه نامتناهی با استفاده از شبکه‌های عصبی

ملک ع.
دانشیار ریاضی کاربردی
ایران، تهران، دانشگاه تربیت مدرس، دانشکده علوم ریاضی، گروه ریاضی کاربردی، صندوق پستی ۱۳۴-۱۴۱۱۵
mala@modares.ac.ir
احمدی ق. و.
دانشجوی دکتری ریاضی کاربردی - نویسنده مسئول
ایران، تهران، گروه ریاضی، دانشگاه پیام نور، صندوق پستی ۳۶۹۷-۱۹۳۹۵
g.ahmadi@pnu.ac.ir
عالیزمینی س. م. م.
دانشجوی دکتری ریاضی کاربردی
ایران، تهران، گروه ریاضی، دانشگاه پیام نور، صندوق پستی ۳۶۹۷-۱۹۳۹۵
seyyedmehdi-mirhosseini@yahoo.com

Archive of SID

چکیده
برنامه‌ریزی خطی نیمه نامتناهی، دسته‌ی مهمی از مسائل بهینه‌سازی است که بی‌نهایت قید را شامل می‌شود. در این مقاله، برای حل این دسته مسائل، یک روش گسسته‌سازی با یک روش شبکه عصبی ترکیب شده است. با یک گسسته‌سازی ساده، مسئله برنامه‌ریزی خطی نیمه نامتناهی به یک مسئله برنامه‌ریزی خطی تبدیل شده است. سپس از یک مدل شبکه عصبی بازگشتی با یک ساختار ساده بر اساس یک سیستم دینامیکی، برای حل مسئله مورد استفاده قرار گرفته است. مسئله انتخاب پورت فولیو و چند مثال عددی دیگر برای نشان دادن کارآمدی مدل ارائه شده، مطرح گردیده است.

کلمات کلیدی
برنامه‌ریزی خطی نیمه نامتناهی، شبکه عصبی بازگشتی، سیستم دینامیکی، گسسته‌سازی، برنامه‌ریزی خطی.

www.SID.ir