Sun Yat- sen Management Review
Vol.4, No.2, November 1996

A Link Between Neural Networks and Mathematical Programming

Ray R. Tsaih
Department of Management Information Systems
National Chengchi University

Abstract

In this paper, we illustrate how an artificial Neural Networks approaches linear programming problems, without any assumption to be placed on the solution. Also a link between ANN and the theory of mathematical programming is illustrated.

1. Introduction

Artificial Neural Networks (ANN) have been studied for more than 40 years, but their applications to optimization problems did not cause much attention until Hopfield presented his work in [1][2]. Since that, some efforts have been made to extend Hopfield's model to several relevant problems, or to improve the performance of Hopfield's model. Those include the work of Hopfield and Tank [3], Tank and Hopfield [4], Kennedy and Chua [5], Cichocki et al. [6], Maa et al. [7], and Lillo et al. [8].

Hopfield and Tank presented a dedicated ANN for solving the traveling salesman problem. Tank and Hopfield proposed the first ANN for solving linear programming (LP) problems. Even though the ANN was rapid, efficient, and did produce some useful results, the obtained solution is unreliable [9][7]. Kennedy and Chua modified Tank and Hopfield model so that it actually converges. Their proof was however incomplete from the optimization theory point of view. Maa et al. analyzed the models of [5], and then established a model based on Kennedy and Chua network. Lillo et al. showed that the implementation evolved from the Kennedy and Chua circuit may not converge to the correct answer, and proposed a network using nonlinearity inherent in the circuit implementation to ensure that the trajectory was forced to the feasible region. The emphasis of Cichocki et al. is on the linear system of equations.

In basic, all the work mentioned above have built up analog electrical circuits that are close-loop networks, and into which the objective function and constraints are mapped. The idea behind the ANN is as follows [7]:

(The constraint) violations are fed back to adjust the states of the neurons of the networks. In this way the overall energy of the network is always decreasing until it achieved a minimum. When the energy attains its minimum the states of the neurons of the network are taken to be a minimizer of the original problem.

In summary, they have claimed the networks as completely stable designs with the concept of the energy function, where the energy functions are formed by adding the original cost functions with penalty terms regarding to the violation of constraints.

Their link to the theory of the mathematical programming however was left mostly unexplored. In basic, techniques of the mathematical programming normally involve an iterative process. At each iteration, a feasible direction with tending to decrease the objective function is determined from the last feasible solution. Then a search task along the feasible direction takes place to find a feasible point that reduces the value of the objective function; or a one-dimensional optimization task along the feasible direction takes place to find a feasible point that at least locally maximally reduces the value of the objective function. The process stops either when there is no feasible direction or when the decrement of the objective function is less than a predetermined terminating criterion.

Here we present a discrete-time version of ANN system (called the NN_LP system) to illustrate how an ANN approaches the LP problem, without any assumption to be placed on the solution. Also a link between the ANN and the theory of the mathematical programming is illustrated.

2. The NN_LP system

The following is a general form for the LP problem:

$$\text{Maximize } \sum c_j x_j$$

subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \qquad \text{all } i = 1, \ldots, m, \qquad \text{(I)}$$

$$x_j \ge 0 \qquad \text{all } j = 1, \ldots, n$$

The given data (the a_{ij}, b_i, and c_j) constitutes the parameters (input constants) of the system. The $\sum_{j=1} a_{ij} x_j \le b_i$ constraints are called functional constraints. The $x_j \ge 0$ restrictions are called non-negativity constraints.

In the previous works of ANN, there is no distinct way of handling with the functional constraints and the nonnegativity constraints; they are considered as the same category of constraints. In contrast, the simplex algorithm of the mathematical programming first transforms the functional inequality constraints into equality constraints by introducing the appropriate slack variables, then the optimal solution is persuaded while maintaining the satisfying of the feasibility requirement.

As same as in the simplex algorithm, the NN_LP system converts all the functional constraints into equivalent equality constraints by introducing slack variables, $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$, that the LP problem (I) is now become as[1]

$$\text{Maximize} \sum c_j x_j$$

subject to:

$$x_{n+i} + \sum a_{ij} x_j = b_i \quad \text{all } i = 1, \ldots, m, \qquad \text{(II)}$$

$$x_j \geq 0 \qquad \text{all } j = 1, \ldots, n+m$$

The NN_LP system, as shown in Figure 1, consists of two similar subsystems which can be run on two independent processor concurrently. The one on the left box, called the P(rimal) subsystem, is constructed directly from the problem (II). The other on the right box, called the D(ual) subsystem is constructed from the dual problem of (I). The network structures of the P subsystem and the D subsystem are shown in Figure 2 and Figure 3. Below, we take the P subsystem to illustrate the idea of their designs.

The bottom layer is designed for the satisfaction of the equality constraints in problem (II). Whenever the activation value of the j^{th} node is x_j for every $j = 1, \ldots, n$, the activation value of the $(n+i)^{th}$ node x_{n+i} has to equal $b_i - \sum_{j=1} a_{ij} x_j$ for every $i = 1, \ldots, m$. So the weights from all of the first n nodes to each of the last m nodes are assigned simply to be the coefficients of the original variables respectively, as shown in Figure 2.

The upper layer and the feedback mechanism are designed for the satisfaction of the nonnegativity constraints and the goal of maximization in problem (II). The concepts of the penalty method and the energy function are adopted here. The more detail is explained in the following section.

3. The Penalty Term, The Energy Function, And The P Subsystem

The activation value of the k^{th} node at the upper layer is h_k for every $k = 1, \ldots, n+m$, and $h_k \equiv \exp(-r\, x_k)$. $\exp(-r\, x_k)$ is a monotonically decreasing function of $r\, x_k$. As the parameter r is being increased, the requirement of a small value of $\exp(-r\, x_k)$ tends to rendering x_k be at least positive, and the tendency becomes stronger as the value of r is increased further. Therefore, $\sum_{j=1} \exp(-r\, x_j) + \sum_{i=1} \exp(-r\, x_{n+i})$ is

1 If there are functional equality constraints, they would be transformed into equivalent equality constraints as $x_k + \sum_{j \neq k} a_{ij}' x_j = b_i'$.

adopted here as the penalty term regarding to the violation of nonnegativity constraints, and r as the penalty parameter.

Let $\mathbf{x} \equiv (x_1, x_2, \ldots, x_n)^t$, $\mathbf{X} \equiv (x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})^t$, $E(r, \mathbf{x}) \equiv - \sum_{j=1}^{n} c_j x_j + \gamma \left[\sum_{j=1}^{n} \exp(-r\, x_j) + \sum_{i=1}^{m} \exp(-r\, b_i + \sum_{j=1}^{n} r\, a_{ij} x_j)\right]$, and $\tilde{E}(r, \mathbf{X}) \equiv - \sum_{j=1}^{n} c_j x_j + \gamma \left[\sum_{j=1}^{n} \exp(-r\, x_j) + \sum_{i=1}^{m} \exp(-r\, x_{n+i})\right]$. $E(r, \mathbf{x})$ is equivalent to $\tilde{E}(r, \mathbf{X})$ since $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j$, for all i = 1, ..., m. γ is a weighted factor that determines the relative contribution of the penalty terms to designed function $\tilde{E}(r, \mathbf{X})$. Although $E(r, \mathbf{x})$ is defined on the whole $\{\mathbf{x}\}$ space, $\tilde{E}(r, \mathbf{X})$ is a defined on a convex set $\Omega \equiv \{\mathbf{X}: x_{n+i} = b_i - \sum_{j=1} a_{ij} x_j$, for all i = 1, ..., m$\}$, and Ω is independent of r.

The function $\tilde{E}(r, \mathbf{X})$ is designed with the idea of the energy function. In the concept of the energy function, the Generalized Liapunov Result stated below is usually used to demonstrate the convergence property of the system. For a proof, the reader should refer to [10].

Let $\mathbf{x} = \mathbf{x}(t)$ be a solution of the following autonomous dynamic system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

where function **f** is continuously differentiable in the domain of interest. A set G is an invariant set for the dynamic system if whenever a point **x** on a system trajectory is in G, the trajectory remains in G.

Generalized Liapunov Result -- Invariant Set Theorem *Suppose that*

(a) $V(\mathbf{x}) \in C^2$ *is a scalar function,*
(b) The set $\Omega s = \{x: V(\mathbf{x}) \le s\}$ *is bounded,*
(c) $\dot{V}(\mathbf{x}) \le 0$ *within* Ωs,
(d) S is the set of points within Ωs, *where* $\dot{V}(\mathbf{x}) = 0$, *and G is the largest invariant set within S.*
Then every trajectory in Ωs *tends to G as time increases*

The scalar function, V, in the above theorem is often called energy function or potential function. In mathematical literature, it is also called a Liapunov function. The key issue of verifying the stability of a dynamic system is sometimes to find an associated energy function, since the existence of an energy function for a dynamic system is often as indication of its good behavior. A way of identifying an energy function of the dynamic system is let $V(\mathbf{x}) = -\int \mathbf{f}(\mathbf{x})\, d\mathbf{x}$ (in other words, $\dot{\mathbf{x}} = - \nabla_x V(\mathbf{x})$), and then check if $V(\mathbf{x})$ has the above properties (a), (b), (c), and (d).

As shown in Figure 2, via the feedback mechanism, $x_j^{new} \equiv x_j + \eta\ [c_j + \gamma\ (r\ h_j - r \sum_{i=1}^{m} a_{ij}\, h_{n+i})\ ]$, for all $j = 1, ..., n$. Let $\Delta x_j = x_j^{new} - x_j$. Thus, $\Delta x_j = \eta\ [c_j + \gamma\ r\,(h_j - \sum_{i=1}^{m} a_{ij}\, h_{n+i})] = - \eta\ \frac{\partial E(r,x)}{\partial x_j}$. Thus, the adjustment of $\mathbf{x}$ follows the negative gradient of $E(r, \mathbf{x})$, and the value of $E(r, \mathbf{x})$ decreases as $\mathbf{x}$ is being adjusted. Therefore, the value of $\tilde{E}(r, \mathbf{X})$ decreases as $\mathbf{x}$ is being adjusted.

Fixed the value r, the Hessian matrix of $E(r, \mathbf{x})$ is positive semidefinite throughout the $\{\mathbf{x}\}$ space. Therefore, given a value r,

(1) $E(r, \mathbf{x})$ is a convex function defined on the $\{\mathbf{x}\}$ space;[2]
(2) The set where $E(r, \mathbf{x})$ achieves its minimum is convex;[3]
(3) Any relative minimum of $E(r, \mathbf{x})$ is a global minimum.[3]

With the property of linear mapping from $\mathbf{x}$ to $\mathbf{X}$, the following statements are also true: (4) if there is a feasible region Ψ of the problem (II), there is a convex set $\tilde{\Psi}$ in the $\mathbf{X}$ space corresponding to that feasible region; otherwise, there is no such $\tilde{\Psi}$; and, when the value r is given, (5) $\tilde{E}(r, \mathbf{X})$ is a convex function defined over Ω, (6) the minimum set Γ_r, where $\tilde{E}(r, \mathbf{X})$ achieves its minimum in Ω, is convex, and (7) any relative minimum of $\tilde{E}(r, \mathbf{X})$ is a global minimum.

Thus, given a value of r, $\tilde{E}(r, \mathbf{X})$ is an energy function of the P subsystem since it has the properties of the energy function.

4. The Link Between NN_LP and Mathematical Programming

To illustrate the link between the NN_LP system and the theory of the mathematical programming, the discrete-time version of the operating procedures of the P subsystem and the D subsystem (shown in Table 1 and Table 2) are used. In the point of view of the mathematical programming, the NN_LP system could involve an iterative process. At each iteration, a feasible direction with tending to decrease the $E(r, \mathbf{x})$ function is determined from the last feasible solution. Then a search task along the feasible direction takes place to find a feasible point that reduces the value of the $E(r, \mathbf{x})$ function. The process stops either when there is no feasible direction or a pre-determined stopping criterion is satisfied. Those idea are adopted in the NN_LP system.

In addition, there are three kinds of possible results in solving an LP problem: an optimal one, an infeasible one, or an unbounded one. The operation of the P subsystem deals with the cases of the problem (I) having an optimal result or an infeasible result. But it is ineffective to deal with the case of having an unbounded result via the P subsystem. As we know, when an LP problem has an unbounded result, its corresponding dual problem has an infeasible result; and when an LP problem has an infeasible result, its corresponding dual problem has an unbounded or infeasible result (i.e., no optimal result). Those idea are also adopted in the NN_LP system.

2 For the proof we refer the reader to *Proposition* 5 in [11].

3 For the proof we refer the reader to Theorem 1 in [12].

To illustrate in more details, let first take the P subsystem to demonstrate the link. As shown in Table 1, the adjusting **x** consecutively in the direction of negative gradient of E(r, **x**) is set up at *Step* 5. *Step* 4 and *Step* 5 detect that if the current **x** hits the neighborhood of a stationary point. *Step* 2 detects if there is a feasible solution. The predetermined stopping criteria are set up in *Step* 8 (which detects if $\tilde{\Psi}$ is empty) and *Step* 9 (which finds out a near-optimal solution if there is one).

In fact, either of the following situations may happen in the operation of the P subsystem: encountering with no stationary point or encountering with a stationary point.

If there is no stationary point, $\Delta\mathbf{x}$ will never be a zero vector. Then, **X** will be drifted to the (positive) infinity since the value of $\tilde{E}$(r, **X**) decreases as **x** is being adjusted and the **X** on the lowest bound of $\tilde{E}$(r, **X**) value is at the (positive) infinity. This situation happens at the case of the problem (II) having an unbounded result. If the problem (II) has a unbounded result, the corresponding Ψ is unbounded at the infinity, and therefore the minimum set Γ_r of $\tilde{E}$(r, **X**) is at the infinity. Whenever this situation happens, the process will take a much long time.

The D subsystem is constructed to check if the dual of the problem (I) has an infeasible result. The arrow line depicted in Figure 1 means that whenever the D subsystem gets an infeasible result (*Step* 8 in Table 2), it sends a message to inform the P subsystem that the problem (I) must has no optimal result, and this is the only situation in that the D subsystem will interrupt the P subsystem. From the simulation results, in the case of the problem (II) having an unbounded result, the D subsystem responds much more quickly than the P subsystem. Therefore, the D subsystem is included to be an auxiliary mechanism of the P subsystem.

If a stationary point **x** is encountered, as shown in Table 1, the value r will be increased whenever the criteria in *Step* 8 and *Step* 9 are not satisfied. $\tilde{E}$(r, **X**) alters and the minimum set G_r drifts as the r value is increased, since they are functions of r. Furthermore, the drifting of the encountered stationary point is bounded. In the operation of the P subsystem, the situation of encountering with a stationary point happens only when the corresponding $\tilde{\Psi}$ is empty or bounded. Under the requirement of a small value of exp(-r y), the tendency of y being at least positive becomes stronger as the value of r is increased further. Therefore, if there is a feasible region Ψ, the minimum set Γ_r tends to $\tilde{\Psi}$ when r tends to being larger.

According to the penalty function theorem, the true minimizer can only be obtained when the penalty parameter is infinite. But, whether there is no feasible solution or a near-optimal solution for the problem (II) can be claimed at a finite, large value of r. Therefore, the goal of the P subsystem is to detect if $\tilde{\Psi}$ is empty and to find out a near-optimal solution if there is one.

We have set up the NN_LP system to testify its performance at 1200 random LP problem. At each LP problem, the number of the (original) variables lies between 1 and 12, the number of functional constraints is set randomly as long as it is bigger than the number of the variables, and each coefficient has a value between -10 and 10. The results have been checked with SAS-OR software, and they are 100% perfect.

5. Conclusion And Future Work

In this paper, we have illustrated how the ANN approaches the LP problems with the concept of the energy function, where the energy functions are formed by adding the original cost functions with penalty terms regarding to the violation of constraints.

Then a link between the ANN and the theory of the mathematical programming has been illustrated.

The future work lie in a couple of directions. Some of them are to speed up the process and to derive a reasonable value of γ for some application problem. Another one is then to evaluate and compare the NN_LP system with other ANN's that solve the same optimization problem.

References

[1] Hopfield, J. (1982). Neural Networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci.*, Vol. 79, pp. 2544-2558, April 1982.

[2] Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons, *Proc. Natl. Acad. Sci.*, Vol. 81, pp. 3088-3092, May 1984.

[3] Hopfield, J. & Tank, D. (1985). Neural computation of decisions in optimization problems, *Biological Cybernetics*, Springer-Verlag, Vol. 52, pp. 141-152, 1985.

[4] Tank, D. & Hopfield, J. (1986). Simple "Neural" Optimization Networks: An a/d Converter, Signal Decision Circuit, And A Linear Programming Circuit, *IEEE Transactions on Circuits and Systems*, CAS-33(5): 533-541, May.

[5] Kennedy, M. & Chua, L. (1988). Neural Networks For Nonlinear Programming, *IEEE Trans. Circuits and Systems*, 35(5): 554-562, May.

[6] Cichocki, A. & Unbehauen, R. (1992). Neural Networks for Solving Systems of Linear Equations and Related Problems, *IEEE Transactions on Circuits and Systems, - I: Fundamental Theory and Applications*, 39(2): 124-138, February.

[7] Maa, C. & Shanblatt, M. (1992). Linear and Quadratic Programming Neural Network Analysis, *IEEE Transactions on Neural Networks*, 3(4): 580-594, July.

[8] Lillo, w., Loh, M., Hui, S., & Zak, S. (1993). On Solving constrained optimization problems with neural networks: a penalty method approach, *IEEE Transactions on Neural Networks*, vol. 4, no. 6, Nov..

[9] Kennedy, M. & Chua, L. (1987). Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming, *IEEE Trans. Circuits and Systems*, 34(2), Feb..

[10] Vidyasagar, M. (1979). *Nonlinear System Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, page 157.

[11] Luenberger, D. (1984). *Linear and Nonlinear Programming*, second edition, Addison-Wesley Publishing, Reading, Mass, page 180.

[12] Luenberger, D. (1984). *Linear and Nonlinear Programming*, second edition, Addison-Wesley Publishing, Reading, Mass, page 181.

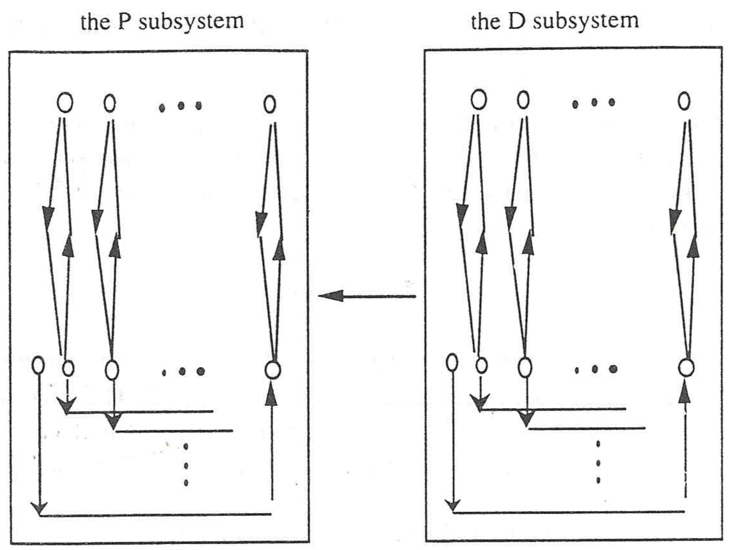

Figure 1 The NN_LP system

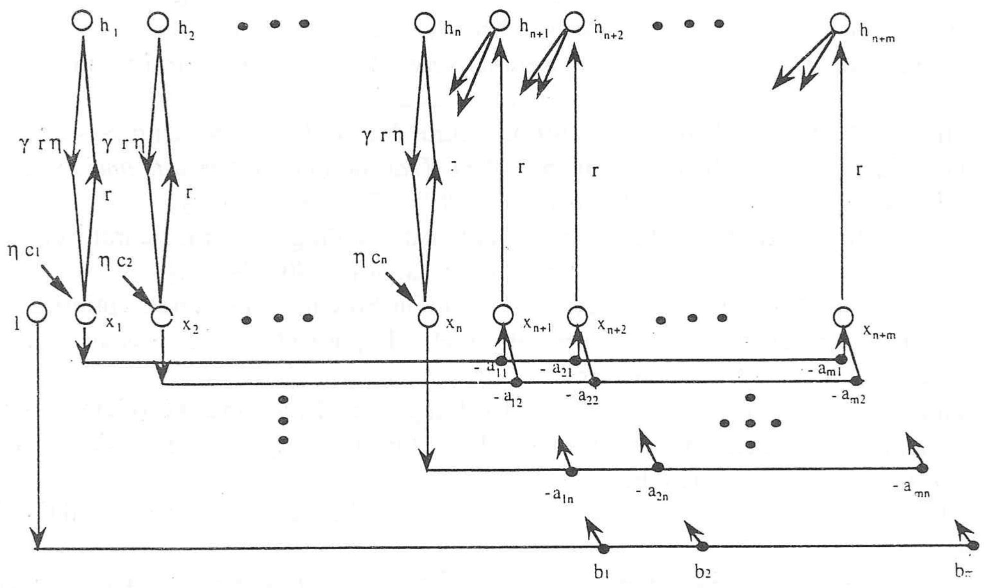

Figure 2 The P subsystem of the NN_LP system for the LP problem (I). Notice that, although not shown in Figure, each node h_{n+i}, i = 1, ..., m, is connected with every node x_j, j = 1, ..., n, and the weight on the connection is $-\gamma r \eta\, a_{ij}$.

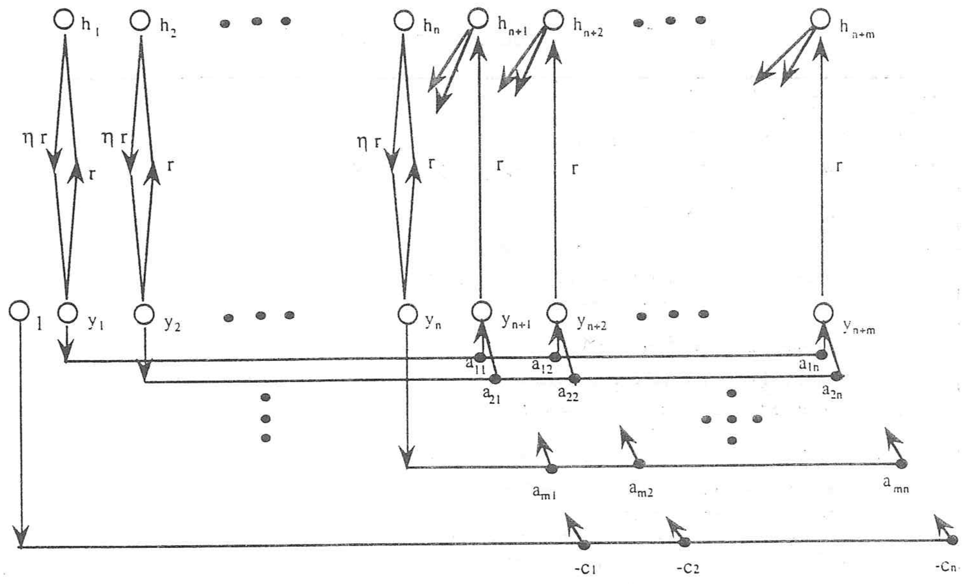

Figure 3 The D subsystem of the NN_LP system for the LP problem (I). Notice that, although not shown in Figure, each node h_{m+i}, $i = 1, \ldots, n$, is connected with every node y_j, $j = 1, \ldots, m$, and the weight on the connection is $-r\eta\, a_{ji}$.

Table 1 The procedure of the P subsystem.

```
Step 0.1 : Set a reasonable, positive $\gamma$ value.
Step 0.2 : Set a tiny, positive $\eta$ value, and $r = 1$.
Step 0.3 : Set NON_NEG = 0.
Step 0.4 : Set $E^* = $ MAXI.
Step 0.5 : Set $x_j^* = -$MAXI, for all $j = 1, 2, \ldots, n$.
Step 0.6 : Set $x_j = 0.0$, for all $j = 1, 2, \ldots, n$.
Step 1 :   Calculate the activation values, $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$, respectively:
           $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j$, all $i = 1, \ldots, m$.
Step 2 :   If $x_j \geq 0$ all $j = 1, \ldots, n+m$, then NON_NEG = 1.
Step 3 :   Calculate the activation value, $h_1, h_2, \ldots, h_{n+m}$, respectively:
           $h_k = \exp(-r x_k)$, for all $k = 1, \ldots, n+m$.
Step 4 :   Calculate $E(r, x)$ and $\nabla E(r, x)$.
           If $\|\nabla E(r, x)\| < 10^{-4}$, then go to Step 7.
Step 5 :   Calculate the new activation values of $x_1, x_2, \ldots, x_n$, respectively:
           $x_j^{new} = x_j + \eta\,[c_j + \gamma\,(r\,h_j - r \sum_{i=1}^{m} a_{ij} h_{n+i})]$, for all $j = 1, \ldots, n$
           where $\eta$ is a suitable value such that $E(r, x^{new}) < E(r, x)$.
           If $\eta < 10^{-30}$, then go to Step 7.
Step 6 :   $x_j^{new} \rightarrow x_j$ for all $j = 1, \ldots, n$. Then go to Step 1.
Step 7 :   $E(r, x) \rightarrow E^{*new}$, and $x_j \rightarrow x_j^{*new}$ for all $j = 1, \ldots, n$.
Step 8 :   If ((NON_NEG = 0) and ($E^{*new} > E^*$)), then STOP and claim it is an infeasible LP
Step 9 :   If ((NON_NEG = 1) and (($x_j^{*new} - x_j^*) < 10^{-4}$, for all $j = 1, \ldots, n$)), then STOP and claim it
           has an optimal result, and print out the result.
Step 10 :  $2.0\,r \rightarrow r$, $E^{*new} \rightarrow E^*$, $x_j^{*new} \rightarrow x_j^*$ for all $j = 1, \ldots, n$, then go to Step 1
```

Table 2 The procedure of the D subsystem.

Step 0.1 :	Set a tiny, positive η value, and $r = 1$.
Step 0.2 :	Set $E^* = \text{MAXI}$.
Step 0.3 :	Set $y_j = 0.0$, for all $j = 1, 2, \ldots, m$.
Step 1 :	Calculate the activation values, $y_{m+1}, y_{m+2}, \ldots, y_{m+n}$, respectively: $y_{m+i} = -c_i + \sum_{j=1}^{m} a_{ji} y_j$ all $i = 1, \ldots, n$.
Step 2 :	If $y_j \geq 0$ all $j = 1, \ldots, n+m$, then **STOP**.
Step 3 :	Calculate the activation value, $h_1, h_2, \ldots, h_{n+m}$, respectively: $h_k = \exp(-r\, y_k)$ all $k = 1, \ldots, n+m$.
Step 4 :	Calculate $E(r, y) = \sum_{k=1}^{m+n} h_k$ and $\nabla E(r, y)$. if $\lVert \nabla E(r, y) \rVert < 10^{-4}$, then go to *Step* 7.
Step 5 .	Calculate the new activation values of $y_1, y_2, \ldots, y_m$, respectively: $y_j^{new} = y_j + \eta\, [r\, h_j + r \sum_{i=1}^{n} a_{ji}\, h_{m+i})]$ all $j = 1, \ldots, m$ where η is a suitable value such that $E(r, y^{new}) < E(r, y)$. if $\eta < 10^{-30}$, then go to *Step* 7.
Step 6 :	$y_j^{new} \rightarrow y_j$ for all $j = 1, \ldots, m$. Then go to *Step* 1.
Step 7 :	If $y_j \geq 0$ all $j = 1, \ldots, n+m$, then **STOP**; otherwise $E(r, y) \rightarrow E^{*new}$.
Step 8 :	If $E^{*new} > E^*$, then **STOP** and claim **it is an infeasible LP**, and send a **no optimal solution** message to the P subsystem; otherwise then $2.0\, r \rightarrow r$, $E^{*new} \rightarrow E^*$the corresponding Ψ is empty; if the constrained problem (II) has an optimal result, the corresponding Ψ is bounded, then go to *Step* 1