Neural Network Implementation for Integer Linear Programming Problem

G.M. Nasira
Professor & Vice Principal
Sasurie College of Engineering
Viyayamangalam, Tirupur (Dt),
Tamil Nadu, India

S. Ashok kumar
Sr. Lecturer, Dept of CA
Sasurie College of Engineering
Vijayamangalam, Tirupur (Dt),
Tamil Nadu, India

T.S.S. Balaji
Executive Director
Sasurie Institutions
Vijayamangalam, Tirupur (Dt),
Tamil Nadu, India

ABSTRACT
Integer Linear Programming Problem (ILPP) is a special case of Linear Programming Problem (LPP) in which a few or all the decision variables are required to be non-negative integers. For solving ILPP, normally Gomory cutting plane or Branch and Bound technique is used. In this paper, for implementation of the problem in neural network we have taken a new and simple hybrid (primal-dual) algorithm which finds the optimal solution for a class of integer linear programming problems. Normally for solving ILPP Gomory method or Branch and Bound technique is used. This new algorithm does not use the simplex method unlike Gomory cutting plane method and Branch and Bound techniques. The algorithm was considered for implementation with Artificial Neural Network (ANN) and the result shows a great improvement in prediction of results. Literature shows that ILPP occurs most frequently in transportation problems, assignment problems, traveling salesman problems, sequencing problems, routing decisions etc. Thus the implementation of the neural network on the new algorithm will provide comprehensive results when applied with any of the said problems.

Keywords
Artificial Neural Network (ANN), Gomory Cutting Plane, Integer Linear Programming Problem (ILPP), Linear Programming Problem (LPP), Branch and Bound technique, Primal-Dual.

1. INTRODUCTION
Linear programming Problem (LPP) is an optimization method applicable for the solution of problems in which the objective function and the constraints are linear functions of the decision variables. The constraints in a linear programming problem may be in the form of equalities or inequalities [2]. Integer Linear Programming problem (ILPP) is an extension of linear programming that some or all the variables in the optimization problem are restricted to take only integer values. The most popular methods used for solving all-integer LPP is cutting plane method designed by Gomory and Branch and Bound method. The integer programming literature contains many algorithms for solving all integer programming problems. After various literature surveys we have obtained a new technique for solving the integer linear programming problem which is considered in this paper is the neural network implementation. The algorithm taken is a hybrid (i.e., primal-dual) cutting – plane method for solving all Integer Linear Programming Problems [3]. ILPP occurs most frequently in various problems like transportation problems, assignment problems, traveling salesman problems and sequencing problems [5,7,8,9,11].

2. INTEGER LINEAR PROGRAMMING PROBLEM
Consider the following type of integer linear programming problem.

Maximize $Z = CX$ such that $AX \leq B$ where

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \text{-------} & \\ a_{n1} & a_{n2} \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \text{---} \\ b_n \end{pmatrix}$$

$$C = (c_1, c_2) \text{ and } X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Where a_{i1}, a_{i2}, b_i, x_1, x_2, c_1, c_2 >= 0, i=1,2,…n and x_1, x_2 are integers.

The algorithm presented by Karthi Keyan and Sampath Kumar [2] is found to have many advantages like, that it does not need the repeated use of simplex algorithm, no need of additional constraints are involved as in Gomory cutting plane method, no branching of the given ILPP into sub problems is required and also it overcomes the problem of degeneracy. Due to such advantages the algorithm was taken for neural network implementation which could provide faster solution, handle huge volume of data and time saving than conventional systems.

Solution to the above ILPP can be obtained by using the following steps

Step 1: Form the coefficient matrix table.

Table 2.1 Matrix Table

x_1	x_2	RHS

a_{11}	a_{12}	b_1
a_{21}	a_{22}	b_2
..	..	..
a_{n1}	a_{n2}	b_n

Step 2: Form the ratio table.

i.e., Divide *bi* by the corresponding non zero values of a_{i1} *and* a_{i2}

Step 3: Find the minimum of { b_1 / a_{11} , b_2 / a_{21}, bn / a_{n1}} and the minimum of { b_1 / a_{12} , b_2 / a_{22}, b_n / a_{n2}} and $\hat{x}_1$ and $\hat{x}_2$ denote its integer values.

Step 4: If both $\hat{x}_1$ and $\hat{x}_2$ is zero then choose the next minimum of x_1 and x_2

This method involves the following three possible cases.

Case I

a. Minimum of x_1 and x_2 are both distinct positive integers.
b. Minimum of x_1 and x_2 are both non integer values with fraction value less than 0.5 or one is non integer value with fraction values less than 0.5 and another one is an integer.
c. Minimum of x_1 and x_2 are both same positive integers.
d. Minimum of any one of x_1 and x_2 or both is zero.

Case II

If both the minimum of x_1 and x_2 are non-integers with fractional value >= 0.5 then it falls into Case II where different steps are involved to find the solution

Case III

If one x_i is integer and another in non-integer with fractional value >=0.5 then it falls into Case III where different steps is involved to find the solution.

In this paper we consider only the Case I type problems for the implementation with neural network. The Case I type problems have the following sub cases.

Case I - (a)

i. Name the minimum of x_1 and x_2 as $\hat{x}_1$ and $\hat{x}_2$
ii. Compute $c_1 \hat{x}_1$ and $c_2 \hat{x}_2$
iii. Compute $\left|c_1\hat{x}_1 - c_2\hat{x}_2\right|$ and $\left|c_1 - c_2\right|$
iv. If $\left|c_1\hat{x}_1 - c_2\hat{x}_2\right| > \left|c_1 - c_2\right|$ then go to step (v) otherwise go to step (vi)
v. Subtract $\left|c_1 - c_2\right|$ from $\hat{x}_i$ which has the highest value and let it be x_i . Go to step (vii)
vi. Subtract $\left|c_1 - c_2\right|$ from $\hat{x}_i$ which has the lowest value and let $\hat{x}_i = x_i$
vii. Obtain $\hat{x}_i$ which corresponds to max($c_1 \hat{x}_1$, $c_2 \hat{x}_1$). If $c_1 \hat{x}_1 = c_2 \hat{x}_2$) then, choose the x_i from the maximum of latest $\hat{x}_i$, go to step (viii)
viii. Substitute the value of x_i in the given constraints and get another value of the variable by considering its minimum integer value.
ix. Substitute the values of x_1 and x_2 in the objective function z to get the optimal value of z.

Case I - (b)

Consider the integer value among the minimum of x_i by omitting its fractional value and follow the steps as in Case I (a)

Case I - (c)

If the minimum of x_1 and x_2 are same integers then find directly the max($c_1 \hat{x}_1$, $c_2 \hat{x}_2$) and follow the steps from (vii) to (ix) of Case I (a).

Case I - (d)

(i) If the minimum of any one of $\hat{x}_i$ or both are zeros, then choose the next minimum as $\hat{x}_i$ and follow steps as in Case I (a)

The data set is formed for Case I type problems with all the 4 sub cases for two variables and was implemented with neural network.

3. ARTIFICIAL NEURAL NETWORKS

Neural networks take a different approach in solving a problem than that of conventional methods. Conventional methods use algorithmic approach, where the method follows a set of instructions in order to solve the problem. Unless we know the specific steps in prior that the method needs to follow, only then the computer can solve the problem. That restricts the problem solving capability of conventional methods to solving problems. But a method would be so much more useful if they could do things that we don't exactly tell them rather train them how to do [1].

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements called neurons which works in parallel to solve a specific problem. A detailed literature survey was made in the area of neural network which has motivated us to apply this technique to solve this problem [7].

4. BACKPROPAGATION ALGORITHM

Backpropagation algorithm is chosen for solving the ILPP Many versions of Backpropagation training algorithms are available in which Resilient Backpropagation training is used which provides

faster results [7]. The following is the algorithm for a three-layer neural network with one hidden layer.

Initialize the weights in the network (often randomly)
i) Do
For each example data (e) in the training set
O = neural-net-output (network, e);
forward pass, T = output for e
Calculate error (T - O) at the output units
Compute delta_wi for all weights
from hidden layer to output layer ;
backward pass Compute delta_wi for all weights
from input layer to hidden layer;backward pass continued
Update the weights in the network
Until all dataset classified correctly or stopping criterion satisfied
ii) Return the network.

4.1 Training Process

We have used Backpropagation for training the network and simulating it with new inputs. There are generally four steps in the training process:

a. Assemble the training data set.
b. Create the network object.
c. Train the network with sample data.
d. Simulate the network with new inputs.

Properly trained networks tend to give reasonable answers when presented with inputs that they have never seen. The sample data for ILPP are assembled, trained and simulated with the network structured.

5. RESULTS OF WORK DONE

Using programming techniques we have solved 150 ILLP analytically. Out of these 150 data sets we have taken 140 set for training the neural network and 10 set for simulating the network with new inputs to predict the output. The data set consists of 8 input variables viz., a_{11}, a_{12}, b_1, a_{21}, a_{22}, b_2, c_1, c_2, and 3 output variables viz., x_1, x_2 and Z.

Table 5.1 Training Data

Inputs									Outputs		
S.No	a11	a12	b1	a21	a22	b2	c1	c2	x1	x2	Z
1	4	4	40	5	10	50	5	6	9	0	45
2	5	10	50	6	4	40	5	4	5	2	33
3	4	6	40	5	10	49	5	7	7	1	42
4	9	7	50	4	4	29	5	6	6	0	30
5	2	3	45	1	3	42	2	4	20	1	44
.							.				
.							.				
136	2	3	35	2	3	43	4	5	16	1	69
137	2	3	45	1	4	42	2	4	20	1	44
138	2	3	34	2	3	42	4	5	16	0	64
139	1	2	28	3	5	40	5	6	12	0	60
140	2	3	30	2	3	43	3	5	13	1	44

Table 5.1 shows few training data which was used to train the network.

Table 5.2 Testing Data Set

Inputs								
S.No	a11	a12	b1	a21	a22	b2	c1	c2
1	5	10	61	4	4	40	6	8
2	1	2	7	5	2	16	7	6
3	6	10	50	6	4	41	5	4
4	2	3	47	1	3	42	2	4
5	4	6	40	5	10	52	7	7
6	1	2	28	3	5	41	5	7
7	2	3	33	2	3	43	3	5
8	6	6	36	1	3	13	10	5
9	1	1	9	4	5	20	1	2
10	7	7	35	1	3	21	10	5

Table 5.2 shows the testing data set with input values. Table 5.3 shows the analytical outputs for the inputs given in Table 5.2.

Table 5.3 Output (Analytical)

Outputs			
S.No	x1	x2	Z
1	8	2	64
2	3	0	21
3	5	2	33
4	21	1	46
5	10	0	70
6	11	1	62
7	14	1	47
8	4	2	50
9	4	0	4
10	5	0	50

The network structure used to train the network is shown in the Figure 5.1. This network structure consists of one Input layer consisting of 8 neurons, and one Hidden Layer consisting of 80 neurons and one Output Layer having 3 neurons. The training of the network was carried out with the neural network toolbox using Matlab 7.0.4.

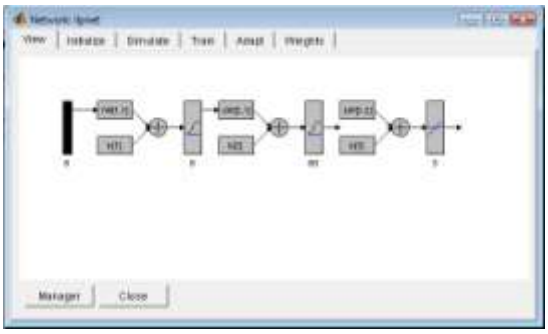

Figure 5.1 Network Structure

During the training process, the goal was set up to 0.01and it has taken 8314 epochs to train the network and the error performance was less than 0.0099, which shows the convergence. Figure 5.2 show the convergence achieved for the ILPP problems with 8314 epochs.

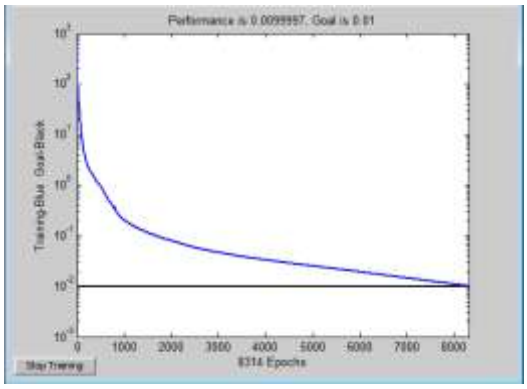

Figure 5.2 ANN Convergence

6. PERFORMANCE ANALYSIS

The trained network was then simulated using the 10 testing data set. The results were compared with the analytical results and were found to be different in fractional values which almost near to the analytical results. The results obtained by simulating the network are shown in the following Table 6.1.

Table 6.1 ANN Outputs

Outputs			
S.No	x1	x2	Z
1	8.4898	1.9827	64.7600
2	3.1546	-0.0383	21.0650
3	5.0025	1.9367	32.9813
4	21.2135	0.7516	46.4952
5	10.0608	-0.1395	70.4971
6	11.4274	0.7714	61.7478
7	13.9686	1.0889	47.0648
8	3.9252	1.6386	49.5028
9	4.0650	0.3912	4.4306
10	5.1899	0.2419	50.1869

The problem taken is considered as the integer programming problem hence the values will fall into integer values only. Thus the values obtained through ANN are rounded and shown in Table 6.2 which achieves nearly to the exact solution of analytical approach.

Table 6.2 ANN Outputs – Rounded Values

Outputs			
S.No	x1	x2	Z
1	8	2	65
2	3	0	21
3	5	2	33
4	21	1	46
5	10	0	70
6	11	1	62
7	14	1	47
8	4	2	50
9	4	0	4
10	5	0	50

Thus the network prediction of the solution to the ILPP was tremendous and with very little error percentage of range from (0-1.6)%. The comparison of the results of Analytical and ANN values of x1, x2 and Z values are shown in Figures 6.1, 6.2 and 6.3 respectively.

Figure 6.1 shows the comparison of the results obtained through Analytical and those of the results obtained through ANN method for the Z value. Hence the error percentage is calculated using the formula

Error Percentage (EP) = ((Analytical Value – ANN value) / Analytical value)* 100

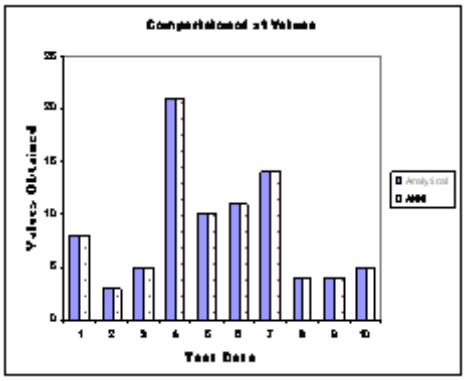

Figure 6.1 Comparison of x1 values

Figure 6.1 shows the output comparison obtained from the analytical method with the ANN for x1 values which does not have any error percentage.

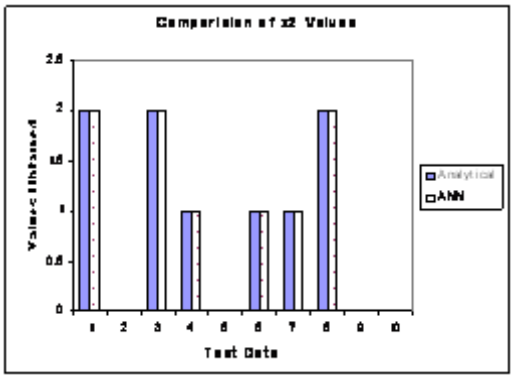

Figure 6.2 Comparison of x2 values

Figure 6.2 shows the output comparison obtained from the analytical method with the ANN for x2 values which also does not have any error percentage

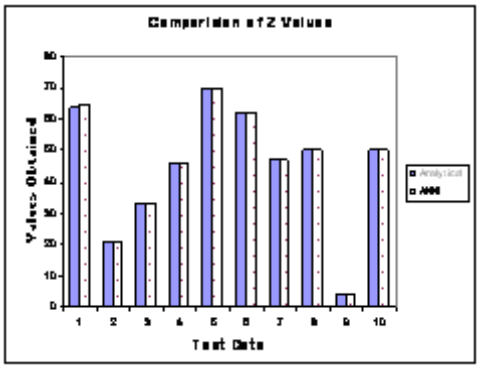

Figure 6.3 Comparison of Z values

Figure 6.3 shows the output comparison obtained from the analytical method with the ANN for Z values with the error percentage range between (0-1.6)%.

Thus the above comparison results shows that the ANN values are highly comparable with the Analytical results and the error percentages was found to be well within the accepted range of (0 – 1.6)%.

7. CONCLUSION

The analysis of the results obtained through ANN shows that it produces almost matching results as of the analytical method in solving out the integer Linear Programming Problem without the use of Gomory method or branch and bound method. Thus the implementation of the proposed new solving method in solving ILPP with neural network will produce very fast and efficient results. This method and implementation thus can serve as an alternate to the Gomary cutting-plane method or the Branch and bound method in solving integer linear programming problems.

8. ACKNOWLEDGMENTS

Our thanks to ACM for allowing us to modify the templates they had developed. The authors wish to express their gratitude to International Journal on Futuristic Computer Applications (IJFCA) for publishing the work in their proceedings.

9. REFERENCES

[1] Kartalopoulous, Stamatios V. “Under- standing neural networks and fuzzy logic”, Prentice hall 2003.

[2] K. Karthikeyan, V.S. Sampath kumar. “A Study on a Class of Linear Integer Programming Problems”, ICOREM 2009. pp. 1196-1210.

[3] Gupta.A. K., Sharma. J.K. “Integer Quadratic Programming. Journal of the Institution of Engineers”, Part Pr: Production Engineering Division Vol. 70, (2). pp. 43 – 47.

[4] M. Ida, K. Kobuchi, and R. ChangYun Lee. “Knowledge-Based Intelligent Electronic Systems”, 1998 Proceedings KES Second International Conference on Vol. 2. Issue, 21-23.

[5] S.Cavalieri. “Solving linear integer programming problems by a novel neural model”. Institute of Informatic and Tele- communications, Italy

[6] Meera Gandhi, S.K. Srivatsa. “Improving IDS Performance Using Neural Network (NN) For Anomalous Behavior Detection”, Karpagam JCS. Vol. 3. Issue 2. Jan-Feb 2009.

[7] Foo Yoon-Pin Simon Takefuji. T. “Integer linear programming neural networks for job-shop scheduling”, IEEE International Conference on Neural Networks1988. pp. 341-348 Vol. 2.

[8] Haixiang Shi. “A mixed branch-and-bound and neural network approach for the broadcast scheduling problem”, 2003**.** ISBN:1-58603-394-8 pp. 42-49.

[9] “Branch and bound algorithm for a facility location problem with concave site dependent costs”, International Journal of Production Economics, Volume 112, Issue 1, March 2008, Pages 245-254.

[10] H. J. Zimmermann, Angelo Monfroglio. “Linear programs for constraint satisfaction problems”. European Journal of Operational Research, Vol. 97, Issue 1, pp. 105-123.