Optimal Production Decision Making by using Artificial Neural Networks and Fuzzy Linear Programming

Khalan J. Rostam[1] Suzan S. Haydar[2]
[1, 2] Sulamani University, Collage of Administration and Economics, Department of Statistics and Informatics
[1] E-Mail: khalan.rostam@univsul.edu.iq
[2] E-Mail: sozan.haider@univsul.edu.iq

Abstract

This research aims to build a mathematical model for some of the products of the Company (Beiji Oil Refinery/ Iraq) and solve the model using the fuzzy linear programming methods in addition to that, how to employ these linear programming models in artificial neural networks by depending on the results of the optimal solution that were reached in the five methods of fuzzy linear programming since artificial neural networks are information processing systems that have the capabilities to imitate the human neural system by developing a model structure to map complex non-linear relationships and processes that are inherent among several influencing variables. The application side of the company implemented (Beiji Oil Refinery/ Iraq) in 2021, such that a neural network was trained using a data set of solved linear programming problems. The objective function used in this training had ten (10) variables and twenty-four (24) constraints equations. This trained neural network was used to optimize oil production profit for 10 different kinds of oil; gas oil, fuel oil, diesel oil, naphtha, light puffs, heavy jet, heavy kerosene, liquid gas, gasoline, and white oil, such that the neural network structure consisted of 274 inputs and 11 outputs with a neural structure of 194 hidden neuron layers. The training algorithm used was Levenberg-Marquardt backpropagation. The neural network results when compared with five methods of fuzziness and comparison between the methods of removing fuzzy in a linear programming model and finding the best method to get maximum profit. The maximum projected profit was up to 98% in a bounded and decomposition method increase from 993423791 IQD to 1943043833 IQD in a day. This paper will increase the current rate of crude oil products in Beiji Oil Refinery and increase the profit of production while maintaining the same quantity of raw materials for daily crude oil products. The paper reached several conclusions, the most prominent of which is that there is a difference in determining the optimal quantities of production and the reflection of this matter on revenues the total achieved when using each of the methods of removing fuzzy, as the results showed that the best way in terms of achieving the highest revenues was when using bounded and decomposition method. Therefore; the importance of this research lies in the topic that I dealt with, which is to make the optimal decision for production using the fuzzy linear programming method and artificial neural networks.

KEYWORDS: Artificial Neural Network, Fuzzy Linear Programming, Fuzziness**,** Levenberg-Marquardt backpropagation.

1. Introduction

Recently, interest in the topics of optimization has increased because of their vital role in most scientific, engineering and administrative problems. In general, optimization is the search for the best result under certain conditions. The main objective of all these decisions is to either reduce the effort or increase the desired and desired benefit. The optimization problem consists of a real-valued objective function that is subject to a set of other real constraints. Due to the different nature of the objective function associated with optimization problems, there is no single method for solving all types of optimization problems in an efficient manner. Therefore, many optimization techniques have been created for different types of optimization problems. Optimization problems have intrinsically interesting properties, such as whether solutions exist or do not exist, or whether those solutions are unique or have more potential with respect to the situation. Linear programming (LP) is one of the most commonly used topics in optimization. It deals with the optimization of a linear function while satisfying a set of linear equality and/or inequality constraints or restrictions. In practice, a LP model involves a lot of parameters whose values are assigned by experts / decision makers. However, both experts and decision makers do not precisely know the value of those parameters in most of the cases. Therefore, fuzzy linear programming (FLP) problem was introduced and studied **[9]**.

The neuron (Greek: Nerve Cell) is the fundamental of the nerve system, neuron is a simple processing unit that receives and combines signals from many other neurons through filamentary input paths **[12]**. The name of artificial neural networks (ANN) indicates computational networks that attempt to simulate, in a great manner, the networks of neurons of the biological (human or animal) central nervous system. This simulation is a great (neuron-by-neuron, cell-by-cell) simulation. It borrows from the neurophysiologic knowledge of biological neurons and of networks such as biological neurons. As an outcome, it differs from conventional (digital or analog) computing machines that serve to replace, develop, or speed up human brain computation without regard to the organization of the computing elements and of their networking **[2]**. The foundation of the artificial neural network's paradigm was laid in the 1950s. Since then, ANNs have earned significant attention because of the development of more powerful hardware

and neural algorithms **[3]**. Artificial neural networks (ANNs) are information processing systems that have the capabilities to imitates the human neural system by developing a model structure to map complex non-linear relationships and processes that are inherent among several influencing variables. In a simpler term, it is a form of a nonlinear regression model that performs an input-output mapping using a set of weights. A feed-forward neural network consists of an input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. This ANN approach is found to be fast and efficient to model complex relationships among variables even in noisy environments and has been employed to solve several real-world problems **[8]**.

2. Definition and Concept of an Artificial Neural Networks (ANN)

An artificial Neural Network is an information processing system that is inspired by the models of biological neural networks. It is an adaptive system that changes its structure or internal information that flows through the network during the training time. In terms of definition, Artificial Neural Network is a computer simulation of a "brain-like" system of interconnected processing units. Neurons are information-processing cells and nothing more than a switch with information input and output. The switch will be activated if there are enough stimuli of other neurons hitting the information input. Then, at the information output, a pulse is sent to, for example, other neurons **[1]**. The biological neurons consist of four important basic parts:

1. **Dendrite**: Receives signals from other neurons
2. **Soma**: Processes the information
3. **Axon**: Transmits the output of this neuron
4. **Synapse**: Point of connection to other neurons

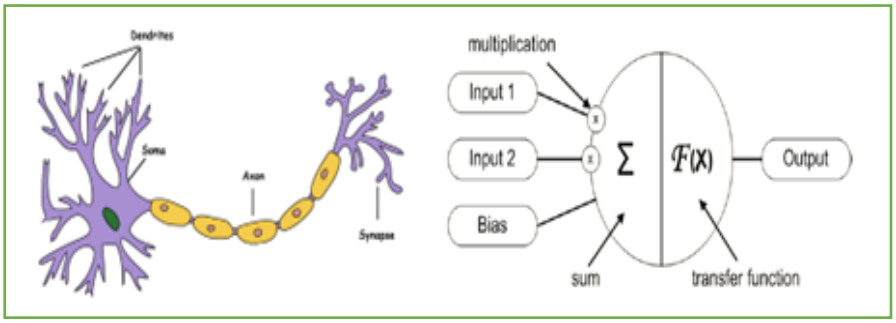

Fig. 1: Biological and artificial neuron design.

When combining two or more artificial neurons we are getting an artificial neural network. If single artificial neuron has almost no usefulness in solving real-life problems the artificial neural networks, have it. In fact, artificial neural networks are capable of solving complex real-life problems by processing information in their basic building blocks (artificial neurons) in a non-linear, distributed, parallel and local way [13].

Artificial neural network architecture consists of:

Input layer: This layer is responsible for receiving information (data), signals, features, or measurements from the external environment. These inputs are usually normalized within the limit values produced by activation functions.

Hidden layer: These layers are composed of neurons which are responsible for extracting patterns associated with the process or system being analyzed. These layers perform most of the internal processing from a network.

Output layer: This layer is also composed of neurons, and thus is responsible for producing and presenting the final network outputs, which result from the processing performed by the neurons in the previous layers [11].

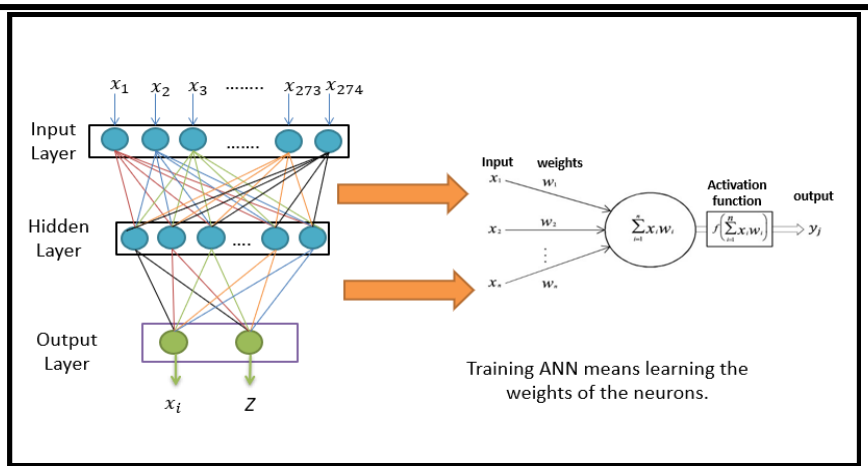

Fig. 2: Represents the architecture of ANNs

3. Learning or Training Artificial Neural Networks

The learning of neural networks may be called training the property that is of primary significance for a neural network, is the ability of the network to learn from the environment, and to improve its performance through learning. Usually, they can be employed by any given type of artificial neural network architecture. Each learning paradigm has many training algorithms. Learning is divided into three types **[13]**:

1. **Supervised learning:** Supervised learning is a machine learning technique that sets the parameters of an ANN from training data. The task of the learning ANN is to set the value of its parameters for any valid input value after having seen the output value. The training data consist of pairs of input and desired output values that are traditionally represented in data vectors.
2. **Unsupervised learning:** Unsupervised learning is a machine learning technique that sets parameters of an ANN based on given data and a cost function which is to be minimized. Cost function can be any function and it is determined by the task formulation. Unsupervised learning is mostly used in applications that fall within the domain of estimation problems such as statistical modelling, compression, filtering, blind source separation and clustering.
3. **Reinforcement learning:** Reinforcement learning is a machine learning technique that sets parameters of an ANN, where data is usually not given, but generated by interactions with the environment. Reinforcement learning is concerned with how an artificial neural network ought to take actions in an environment to maximize some notion of long-term reward.

4. Feed-forward Artificial Neural Networks

Artificial neural network with feed-forward topology is called Feed-Forward Neural Network (FFNN) and as such has only one condition: information must flow from input to output in only one direction with no back-loops. There are no limitations on number of layers, type of transfer function used in individual artificial neuron or number of connections between individual artificial neurons **[13]**.

5. Backpropagation Algorithm

In order to train the established FFNN, the Back Propagation (BP) algorithm can be utilized. Considering a multilayer FFNN, such as the one with a three-layer. the BP algorithm is a method for training the weights in a multilayer feed-forward neural network. As such, it requires a network structure to be defined of one or more layers where one layer is fully connected to the next layer. A standard network structure is one input layer, one hidden layer, and one output layer. BP can be used for both classification and regression problems. It is a gradient descent technique that is used to minimize the output error based on the updated network weights **[14]**.

6. Levenberg-Marquardt Backpropagation

The Levenberg-Marquardt algorithm, also known as the damped least squares (DLS) method, is used to solve non- linear least squares problems. While backpropagation is a steepest descent algorithm, the Levenberg-Marquardt algorithm is derived from Newton's method that was designed for minimizing functions that are sums of squares of nonlinear functions [1,2]. **[7]** Newton's method for optimizing a performance index $F(\mathbf{x})$ is

$$x_{k+1} = x_k - A_k^{-1} g_k \quad (1)$$

$$A_k = \nabla^2 F(x)|_{x=x_k} \quad (2)$$

$$g_k = \nabla F(x)|_{x=x_k} \quad (3)$$

Where $\nabla^2 F(x)$ is the Hessian matrix and $\nabla F(x)$ is the gradient. Assume that $F(\mathbf{x})$ is a sum of squares function:

$$F(x) = \sum_{i=1}^{N} v_i^2(x) = v^T(x)v(x) \quad (4)$$

then the gradient and Hessian matrix are:

$$\nabla F(x) = 2J^T(x)v(x) \quad (5)$$

$$\nabla^2 F(x) = 2J^T(x)J(x) + 2S(x) \quad (6)$$

where J(x) is the Jacobian matrix

$$J(x) = \begin{bmatrix} \frac{\partial v_1(x)}{\partial x_1} & \frac{\partial v_1(x)}{\partial x_2} & \cdots & \frac{\partial v_1(x)}{\partial x_n} \\ \frac{\partial v_2(x)}{\partial x_1} & \frac{\partial v_2(x)}{\partial x_2} & \cdots & \frac{\partial v_2(x)}{\partial x_n} \\ \frac{\partial v_N(x)}{\partial x_1} & \frac{\partial v_N(x)}{\partial x_2} & \cdots & \frac{\partial v_N(x)}{\partial x_n} \end{bmatrix} \quad (7)$$

And $S(x) = \sum_{i=1}^{N} v_i(x)\nabla^2 v_i(x)$ (8)

If S(x) is assumed to be small then the Hessian matrix can be approximated as:

$$\nabla^2 F(x) \cong 2J^T(x)J(x) \quad (9)$$

Substituting Eq. (5) and Eq. (9) into Eq. (1), we achieve the Gauss-Newton method as:

$$\Delta x_k = -[J^T(x_k)J(x_k)]^{-1} J^T(x)v(x_k) \quad (10)$$

One problem with the Gauss-Newton method is that the matrix may not be invertible. This can be overcome by using the following modification to the approximate Hessian matrix:

$$G = H + \mu I \quad (11)$$

This leads to the Levenberg-Marquardt algorithm.

$$\Delta x_k = -[J^T(x_k)J(x_k) + \mu_k I]^{-1} J^T(x_k)v(x_k) \quad (12)$$

Using this gradient direction, and recompute the approximated performance index. If a smaller value is yield, then the procedure is continued with the μ_k divided by some factor $g > 1$. If the value of the performance index is not reduced, then $\mu_k$is multiplied by ϑ for the next iteration step. The key step in this algorithm is the computation of the Jacobian matrix. The elements of the error vector and the parameter vector in the Jacobian matrix (8) can be expressed as:

$$v^T = [v_1 \quad v_2 \quad K \quad v_n] = \left[e_{11} \quad e_{21} \; K \; e_{S^m 1} \quad e_{S^m 2} \; K \; e_{S^m Q}\right] \quad (13)$$

$$x^T = [x_1 \quad x_2 \quad K \quad x_N] = \left[w_{11}^1 \quad w_{12}^1 \; K \; w_{S^1 R}^1 \quad b_1^1 \; K \; b_{S^1}^1 \; w_{11}^2 \; K \; b_{S^M}^M\right] \quad (14)$$

where the subscript *N* satisfies:

$$N = Q \times S^M \quad (15)$$

and the subscript *n* in the Jacobian matrix satisfies:

$$n = S^1(R+1) + S^2(S^1+1) + L + S^M(S^{M-1}+1) \quad (16)$$

Making these substitutions into Eq. (7), then the Jacobian matrix for multilayer network training can be expressed as:

$$J(x) = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_{11}^1} & \frac{\partial e_{11}}{\partial w_{12}^1} & L\frac{\partial e_{11}}{\partial w_{S^1 R}^1} & \frac{\partial e_{11}}{\partial b_1^1} & L \\ \frac{\partial e_{21}}{\partial w_{11}^1} & \frac{\partial e_{21}}{\partial w_{12}^1} & L\frac{\partial e_{21}}{\partial w_{S^1 R}^1} & \frac{\partial e_{21}}{\partial b_1^1} & L \\ M & M & M & M & L \\ \frac{\partial e_{S^M 1}}{\partial w_{11}^1} & \frac{\partial e_{S^M 1}}{\partial w_{12}^1} & L\frac{\partial e_{S^M 1}}{\partial w_{S^1 R}^1} & \frac{\partial e_{S^M 1}}{\partial b_1^1} & L \\ \frac{\partial e_{12}}{\partial w_{11}^1} & \frac{\partial e_{12}}{\partial w_{12}^1} & L\frac{\partial e_{12}}{\partial w_{S^1 R}^1} & \frac{\partial e_{12}}{\partial b_1^1} & L \\ M & M & M & M & \end{bmatrix} \quad (17)$$

In standard backpropagation algorithm, the terms in the Jacobian matrix is calculated as:

$$\frac{\partial \hat{F}(x)}{\partial x_l} = \frac{\partial e_q^T e_q}{\partial x_1} \quad (18)$$

For the elements of the Jacobian matrix, the terms can be calculated by:

$$[J]_{h,l} = \frac{\partial v_h}{\partial x_l} = \frac{\partial e_{kq}}{\partial w_{ij}} \quad (19)$$

Thus, in this modified Levenberg-Marquardt algorithm, we compute the derivatives of the errors, instead of the derivatives of the squared errors as adopted in standard backpropagation. Using the concept of sensitivities in the standard backpropagation process, here we define a new Marquardt sensitivity as:

$$S/0_{ih}^{o/m} = \frac{\partial v_h}{\partial n_{iq}^m} = \frac{\partial e_{kq}}{\partial n_{iq}^m} \qquad (20)$$

Where $h = (q-1)S^M + k$.

Using the Marquardt sensitivity with backpropagation recurrence relationship, the elements of the Jacobian can be further calculated by:

$$[J]_{h,l} = \frac{\partial e_{kq}}{\partial w_{ij}^m} = \frac{\partial e_{kq}}{\partial n_{iq}^m}\frac{\partial n_{iq}^m}{\partial w_{ij}^m} = S/0_{ih}^{o/m}\, a_{jq}^{m-1} \qquad (21)$$

If x_l is a bias,

$$[J]_{h,l} = \frac{\partial e_{kq}}{\partial b_i^m} = \frac{\partial e_{kq}}{\partial n_{iq}^m}\frac{\partial n_{iq}^m}{\partial b_i^m} = S/0_{ih}^{o/m} \qquad (22)$$

The Marquardt sensitivities can be computed using the same recurrence relations as the one used in the standard BP method, with one modification at the final layer. The Marquardt sensitivities at the last layer can be given by:

$$S/0_{ih}^{0/m} = \frac{\partial e_{kq}}{\partial n_{iq}^M} = \frac{\partial (t_{kq} - a_{kq}^M)}{\partial n_{iq}^m} = -\frac{\partial a_{kq}^M}{\partial n_{iq}^m} = \begin{cases} -f\&M\left(\partial n_{1q}^m\right) & i = 1 \\ 0 & i \neq 1 \end{cases} \qquad (23)$$

After applying the P_q to the network and computing the corresponding output a_m^q , the LMBP algorithm can be initialized by

$$S/0_q^{0/M} = -F\&M\left(n_q^m\right) \qquad (24)$$

Each column of the matrix should be backpropagated through the network so as to generate one row of the Jacobian matrix. The columns can also be backpropagated together using:

$$S/0_q^{0/m} = F\&M\left(n_q^m\right)(W^{m+1})S_q^{0/m+1} \qquad (25)$$

The entire Marquardt sensitivity matrices for the overall layers are then obtained by the following augmentation

$$S_0^{0/m} = [S_1^{0/m} | S_2^{0/m} | K | S_Q^{0/m+1}] \qquad (26)$$

7. Application

After describing the optimal solution methods for production by using the five methods in fuzzy linear programming, we will now explain in more detail how to reach the optimal decision for the quantities produced and to be available in the company to prevent a deficit, by focusing on (demand quantities, production quantities) as inputs and the case, the revenues generated as outputs. Based on the fuzzy linear programming method through the model in the following ways:

1- Robust ranking method**.**
2- Central of gravity method.
3- Pascal method.
4- Graded mean and integration method.
5- Bounded and decomposition method.

The data was arranged and collected according to the following:

The following tables represent the selling prices, daily production quantities, and daily order quantities.

Table 1: Selling prices of products and production costs in Iraqi dinars

Product	Selling price	Production cost		
gas oil	275000	٩٩٠٣٤	٩٥٧٠٤	85031
fuel oil	100000	١١٣٥٠٠	١٠٩٨٣٤	61871
diesel oil	65000	93402	60436	30508
naphtha	:	:	:	:
light puffs	:	:	:	:
heavy jet	:	:	:	:
heavy kerosene	:	:	:	:
liquid gas	:	:	:	:
gasoline	300000	١٥٧٦١٠	١٥٤٨٥١	103637
white oil	125000	106933	114905	69695

Table 2: Daily production quantities

Product	Available quantity	Planned Quantity	Actual quantity
gas oil	1141	958	817
fuel oil	3417	3255	3033
diesel oil	620	485	310
Naphtha	:	:	:
light puffs	:	:	:
heavy jet	:	:	:
heavy kerosene	:	:	:
liquid gas	:	:	:
gasoline	804	737	685
white oil	869	497	390

Table 3: Daily order quantities

Product	Highest order	Actual Order	Lowest Order
gas oil	964	703	673
fuel oil	3206	2945	2850
diesel oil	610	425	300
naphtha	:	:	:
light puffs	:	:	:
heavy jet	:	:	:
heavy kerosene	:	:	:
liquid gas	:	:	:
gasoline	622	571	563
white oil	691	276	225

Table 4: The needs of production inputs

Product	**Cooling Water /m3**	**Water vapors/m3**	**Electric power/watt**	**Fuel used/m3**
Gas oil	2.928	3.401	4.815	2.955
	3.212	4.238	5.478	3.921
	4.496	6.075	6.555	4.593
fuel oil	2.928	3.401	4.815	2.955
	3.212	4.238	5.478	3.921
	4.496	6.075	6.555	4.593
diesel oil	2.928	3.401	4.815	2.955
	3.212	4.238	5.478	3.921
	4.496	6.075	6.555	4.593
	:	:	:	:
:	:	:	:	:
:	:	:	:	:
gasoline	19.966	2.372	0.174	4.7
	23.4	3.219	1.953	5.55
	27.021	4.156	3.272	6.43
white oil	2.928	4.501	4.815	2.955
	3.212	5.238	5.478	3.921
	4.496	6.075	6.555	4.593
The available quantity of	180202	81200	3600722	100500
	300400	11440	6000722	200500

production supplies	350000	150980	7201444	410200

7.1 Mathematical model for the fuzzy linear programming problem

A fuzzy linear programming problem model was formulated depending on the data taken and the type of problem to be solved. Building the model first requires defining the decision variables that represent the quantity of products produced by the Beiji refinery, which are 10 products:

X_1: gas oil , X_2: $fuel\ oil$,X_3: $diesel\ oil$, X_4: $naphta$, X_5: $light\ puffs$

X_6: $heavy\ jet$, X_7: $heavy\ kerosen$, X_8: $liquid\ gas$, X_9: $gasoline$

X_{10}: $white\ oil$

$\tilde{X}_i = (x_i, y_i, t_i) \quad i = 1,2, \dots\dots\dots,10$

$P_j = Price \quad j = 1,2, \dots\dots\dots,10$

And that model consists of:

- ✓ The objective function is a maximization function (selling prices - fuzzy production costs)

$Max\ (Z_1, Z_2, Z_3) = (275000) * (x_1, y_1, t_1) + (100000) * (x_2, y_2, t_2) + (65000) * (x_3, y_3, t_3) + \dots + (300000) * (x_9, y_9, t_9) + (125000) * (x_{10}, y_{10}, t_{10}) - (99034,95704,85031) * (x_1, y_1, t_1) - (113500,109834,61871) * (x_2, y_2, t_2) - (93402,60436,30508) * (x_3, y_3, t_3) - \dots - (157610,154851,103637) * (x_9, y_9, t_9) - (106933,114905,69695) * (x_{10}, y_{10}, t_{10})$

Subject to:

Production constraints

$(x_1, y_1, t_1) \leq (1141,958,817)$

$(x_2, y_2, t_2) \leq (3417,3255,3033)$

$\vdots$

$(x_9, y_9, t_9) \leq (804,737,685)$

$(x_{10}, y_{10}, t_{10}) \leq (869,497,390)$

Product Demand constraints

$(x_1, y_1, t_1) \geq (964,703,673)$

$(x_2, y_2, t_2) \geq (3206,2945,2850)$

$\vdots$

$(x_9, y_9, t_9) \geq (622,571,563)$

$(x_{10}, y_{10}, t_{10}) \geq (691,276,255)$

Production requirement constraints

- ✓ Cooling water constraints

$(2.928,3.212,4.496) * (x_1, y_1, t_1) + (2.928,3.212,4.496) * (x_2, y_2, t_2) + (2.928,3.212,4.496) * (x_3, y_3, t_3) + \dots + (19.966,3.42,7.021) * (x_9, y_9, t_9) + (2.928,3.212,4.496) * (x_{10}, y_{10}, t_{10}) \leq (180202, 300400, 350000)$

- ✓ Water vapor constraints

$(3.401,4.238,6.075) * (x_1, y_1, t_1) + (3.401,4.238,6.075) * (x_2, y_2, t_2) + (3.401,4.238,6.075) * (x_3, y_3, t_3) + \dots + (2.372,3.219,4.156) * (x_9, y_9, t_9) + (4.501,5.238,6.075) * (x_{10}, y_{10}, t_{10}) \leq (81200,11440,150980)$

- ✓ Electric power constraints

$(4.815,5.478,6.555) * (x_1, y_1, t_1) + (4.815,5.478,6.555) * (x_2, y_2, t_2) + (4.815,5.478,6.555) * (x_3, y_3, t_3) + \dots + (0.174,1.953,3.272) * (x_9, y_9, t_9) + (4.815,5.478,6.555) * (x_{10}, y_{10}, t_{10}) \leq (3600722,6000722,7201444)$

- ✓ Fuel used constraints

$(2.955,3.921,4.593) * (x_1, y_1, t_1) + (2.955,3.921,4.593) * (x_2, y_2, t_2) + (2.955,3.921,4.593) * (x_3, y_3, t_3) + \dots + (4.7,5.55,6.43) * (x_9, y_9, t_9) + (2.955,3.921,4.593) * (x_{10}, y_{10}, t_{10}) \leq (100500,200500,410200)$

Non-negative constraint

$(x_i, y_i, t_i) \geq 0$

After removing the fuzzy from the model by using the five methods, which are (Robust ranking method, Central of gravity method, Pascal method, Graded mean and integration method, Bounded and decomposition method) and solving the model in each method, a comparison was made between the results of the methods used, as we found that there is a difference in the optimal quantities to be produced in all the methods used in the research in addition to there is a difference in the value of the net profit, as Table (5) shows the differences between the quantities of products that must be produced in each method used.

Table 5: A comparison of the methods used to remove fuzzy from the FLP problem

Name of Product	Actual quantity	Robust Ranking Method	Central of Gravity Method	Pascal Method	Graded Mean Integration Reorientation Method	Bounded and Decomposition Method
gas oil	817	979.24	972.03	968.42	964.81	1141
fuel oil	3033	3225.33	3235.32	3240.32	2972.67	3417
diesel oil	445	465.00	471.67	475.00	478.33	620
naphtha	:	:	:	:	:	:
light puffs	:	:	:	:	:	:
heavy jet	:	:	:	:	:	:
heavy kerosene	:	:	:	:	:	:
liquid gas	:	:	:	:	:	:
gasoline	685	744.83	742.38	741.16	739.93	804
white oil	397	629.48	585.31	563.23	541.14	869
Net profit/ IQD	498615140	645244602.75	572409206.24	536988667.41	502724408.99	993423791

In table (5), profits in the production of five products and which are (Gas oil, Naphtha, Heavy jet, Gasoil, and white oil) because the cost of production is lower than the price of the sell, while the production of five products and which are (Fuel oil, Diesel oil, Light puffs, Heavy kerosene, and Liquid gas) causes losses because the cost of production is higher than the price of selling so the production has tended to meet the costs of demand only to reduce losses in all ways above. The net profit of the five methods was higher than the net profit achieved in the Northern Refineries (Al-Beiji Refinery) of (498615140) dinars, with the highest net profit level achieved at the bounded and decomposition method of the results achieved and a value of (993423791) dinars.

7.2 Building a Model Using Artificial Neural Networks

To reach the best quantities (production, demand) for each of the company's ten products. Creating 30 models for data of five methods of fuzzy linear programming problems by using Monte Carlo simulation, will be based on programming (Excel 2019) and solving it using neural networks based on the (MatlabR2019b) program, but before starting, the model data must be configured to suit the requirements of artificial neural networks as shows in the following table.

sample	Inputs												Outputs					
	Coefficient of constraints							Coefficient of objective function										
	a11	a12	.....	a19	a110	b 1	...	c 1	c 2	...	c 9	c 10	X1	X2	...	X9	X10	Z
1	1	0	.....	0	0	979.24	...	182967.44	12314.44	...	169376.71	36686.46	979.24	3225.33	...	744.83	629.48	645244602.75
2	1	0	.....	0	0	986.12	...	197745.06	12522.52	...	185284.58	33239.43	901.90	2922.57	...	717.61	540.03	670677861.79
3	1	0	.....	0	0	1040.18	...	179318.45	13196.40	...	195656.00	35061.57	755.60	3604.28	...	697.68	495.49	585124850.24
4	1	0	.....	0	0	911.87	...	170496.06	14769.14	...	205824.22	33790.70	878.42	3520.24	...	646.45	512.03	558878331.58
5	1	0	.....	0	0	935.91	...	153874.36	16112.25	...	215718.93	36402.97	851.46	4052.59	...	756.46	486.43	601563253.43
6	1	0	.....	0	0	1063.13	...	148739.89	14836.80	...	213406.99	32116.04	759.53	4390.35	...	716.12	536.30	650377539.33
7	1	0	.....	0	0	935.27	...	162920.90	15693.30	...	232808.67	30346.89	592.29	4044.05	...	716.11	465.54	630243740.60
8	1	0	.....	0	0	876.85	...	197447.31	17767.31	...	243789.07	28523.34	512.67	4590.61	...	729.02	513.44	555062915.76
9	1	0	.....	0	0	900.01	...	210135.91	16613.16	...	205915.15	19294.88	417.81	4209.56	...	779.66	532.02	476810142.04
10	1	0	.....	0	0	850.84	...	215034.22	18751.61	...	194683.64	18098.10	395.67	4406.60	...	815.38	454.32	551349544.20
11	1	0	.....	0	0	768.10	...	217860.65	17326.90	...	179067.50	19886.80	392.27	4202.49	...	655.02	421.21	497333792.65
12	1	0	.....	0	0	766.73	...	180672.53	18350.50	...	181366.06	20361.06	383.62	4029.76	...	622.44	326.10	483118864.62
13	1	0	.....	0	0	885.04	...	165450.67	20268.89	...	203329.15	19682.48	360.73	3738.91	...	733.01	359.70	426486533.81
.	.	.	.....	.	.	.	...	.	.	...	.	.	.	.	...	.	.	.
.	.	.	.....	.	.	.	...	.	.	...	.	.	.	.	...	.	.	.
24	1	0	.....	0	0	727.46	...	144026.01	18629.28	...	185212.21	23275.53	410.39	2534.06	...	999.88	556.85	565785528.19
25	1	0	.....	0	0	659.08	...	162604.34	19590.74	...	223304.21	21463.39	399.81	2296.55	...	938.02	504.95	568217437.07
26	1	0	.....	0	0	765.40	...	185523.95	19919.24	...	224712.75	20756.20	445.51	2302.88	...	953.77	460.11	475138180.08
27	1	0	.....	0	0	776.37	...	190667.90	22225.63	...	210193.95	19251.95	440.98	2501.57	...	1050.47	447.74	481705205.91
28	1	0	.....	0	0	803.42	...	186425.62	22866.45	...	235686.91	17864.93	424.89	2895.98	...	1211.19	416.34	496683523.41
29	1	0	.....	0	0	657.74	...	201004.58	23590.16	...	263893.80	17978.64	368.39	2890.02	...	1152.94	503.54	538524065.17
30	1	0	.....	0	0	537.76	...	206253.14	23085.02	...	225080.04	17620.38	381.08	2991.56	...	1323.94	525.63	543539621.87

Table 6: Represents the inputs and outputs

a. Robust ranking method

Robust ranking method is the method to fuzziness for removing fuzzy in fuzzy linear programming **[4]**. After the training process is completed for the data obtained from the robust ranking method, we have the neural networks as in Fig.3 shows the neural networks in the final form after the completion of the training process of the Robust ranking method.

Table 7: Represents the outputs of neural networks for the robust ranking method

Sample		1	2	3	4	5	
Output	X1	526.66	655.25	-1082.45	-89.71	626.82	3
	X2	3849.18	5382.59	8351.77	5008.80	1374.38	5
	X3	1627.27	2056.35	2964.03	2656.76	2941.91	2
	X4	-702.30	-1693.66	-2035.45	-2488.12	-1212.30	-2
	X5	326.30	-78.59	-857.75	-204.18	-313.60	-
	X6	-4296.81	-2807.36	-1749.61	-1688.17	-2647.59	-2
	X7	-237.83	345.86	155.35	-537.43	151.14	-
	X8	111.19	58.46	52.56	137.50	-9.04	-
	X9	-15.74	-934.37	-1280.57	-1550.76	-2239.64	-
	X10	567.99	761.31	988.58	1613.78	1478.90	1
	Z	460375474.31	432008302.53	1373048427.11	1044726661.03	603570659.05	2500
Sample		7	8	9	10	11	
O =	X1	164.00	467.13	1453.13	1590.80	1624.11	2

	X2	1766.48	742.43	-1389.91	-3771.83	68.85	-2
	X3	3176.97	2063.81	2736.83	2691.36	1684.84	1
	X4	-2395.11	-1037.59	-1160.09	-1021.23	-2232.34	-1
	X5	453.23	1493.95	2764.46	2798.15	2946.35	2
	X6	-2078.55	-1066.67	-2715.09	-1977.41	-1035.10	-5
	X7	333.32	-320.96	921.25	1019.54	967.04	2
	X8	-194.68	-105.83	219.44	267.94	-45.77	-
	X9	-89.24	1538.98	570.69	413.60	1335.63	8
	X10	972.81	1043.85	1099.02	1183.28	1499.52	1
	Z	1175043333.30	582795482.90	507693985.55	1039970073.74	538551054.39	1024
Sample		13	14	15	16	17	
Output	X1	214.68	487.19	-5.72	-528.18	-1142.09	-2
	X2	-4739.12	1923.19	6480.44	6651.49	6516.73	10
	X3	1612.60	2218.07	1953.72	1438.46	2804.16	2
	X4	-532.00	-739.42	-2150.27	-1375.07	-1805.12	-3
	X5	2186.21	3163.52	3547.99	3328.11	2402.63	1
	X6	343.38	-1614.16	-2966.16	-2290.23	-829.51	-8
	X7	1051.77	212.03	572.54	818.56	474.40	-1
	X8	-128.44	-65.26	-275.95	-115.91	-150.73	-
	X9	1126.09	1059.46	-781.77	1993.53	706.26	1
	X10	777.78	535.71	656.49	753.51	765.18	9
	Z	898397436.93	1361986373.38	973628572.93	513931866.61	1168219398.38	7587
Sample		19	20	21	22	23	
	X1	-1518.09	-1635.67	371.27	-1359.19	577.04	7
	X2	6752.09	6792.75	1172.94	5047.45	1429.72	-6
	X3	2257.52	2767.65	3252.17	3398.19	2427.39	2
Output	X4	-3378.22	-1333.94	-1981.15	-2363.40	-2371.87	-1
	X5	2387.48	1900.20	2010.51	2902.88	2599.90	1
	X6	-1390.59	-486.84	211.70	-1675.42	-1775.23	-8
	X7	-377.95	-1902.16	-114.65	299.02	71.18	-5
	X8	-61.90	-6.57	102.25	183.44	12.99	-
	X9	-50.47	3268.00	630.93	85.32	113.74	5
	X10	1381.62	1222.04	1003.46	1181.35	1378.25	1
	Z	-33564971.60	45657152.15	629468615.08	785031856.72	1082901269.00	1669
Sample		25	26	27	28	29	
Output	X1	1096.35	8.57	1762.51	1703.15	1758.13	1
	X2	-4572.88	-7037.61	-2674.91	-6498.42	-1144.68	-1
	X3	1488.47	1419.79	718.42	1174.90	688.26	-3
	X4	-2008.50	-2310.64	-1572.98	-658.24	-273.30	7

	X5	1472.72	1439.92	1029.10	832.17	1529.54	-
	X6	-1664.02	-1805.00	-1874.31	-1586.39	-216.82	3
	X7	810.70	1064.20	-215.33	1306.01	2388.12	2
	X8	-126.36	82.04	-75.72	-127.10	-204.40	-
	X9	182.23	-813.36	1835.58	-1167.06	121.59	-
	X10	1073.50	1168.22	1082.65	631.41	365.83	4
	Z	970661671.05	1225012968.10	1490575710.24	2035965970.23	1307605846.89	1840

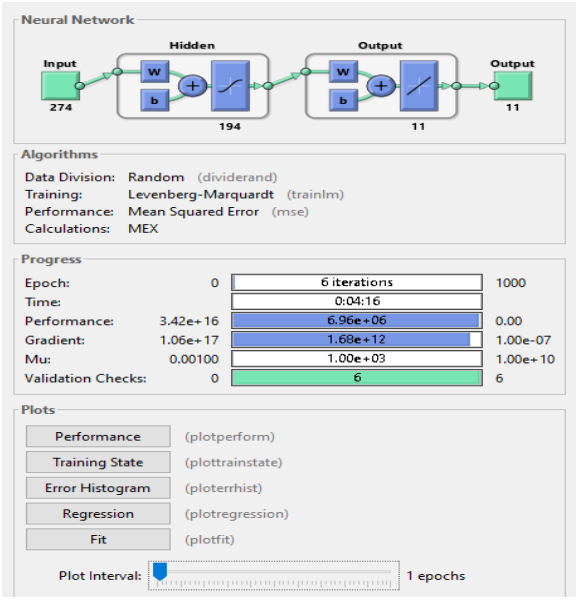

Fig.3: NN in the final image after completing the training process method of robust ranking method

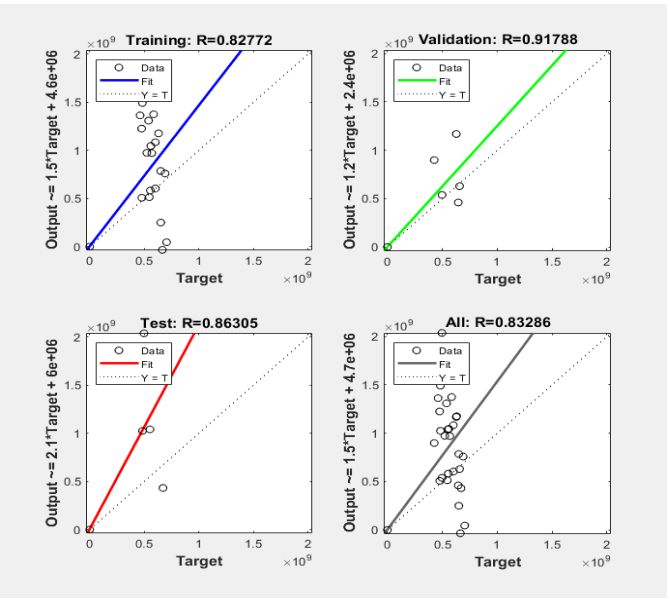

Fig.4: NNs training regression of robust ranking method

In Figu g ı determinations value, which values between (0,1) to determine the acceptability of the simulation results in fuzzy linear programming. The simulation results of the robust ranking method were successful but did not reach the optimum level in decision-making, as the value of the coefficient of determinations ranged between (0.82-0.86), so to get better results we will repeat the training process for several attempts to reach more accurate results.

b. Central of gravity method

Central of gravity method is the method to fuzziness for removing fuzzy in fuzzy linear programming **[10]**. After the training process is completed for the data obtained from the central of gravity method, we have the neural networks as in Fig.5 shows the neural networks in the final form after the completion of the training process of the central of gravity method.

Table 8: Represents the outputs of neural networks for the central of gravity method

Sample		1	2	3	4	5	6
Output	X1	993.45	1311.23	1074.02	909.46	944.12	902
	X2	2861.07	-1290.10	2788.55	3052.50	3187.49	3524
	X3	555.88	401.84	273.09	168.30	476.43	563
	X4	1079.95	781.68	1025.10	1120.91	1427.20	1134
	X5	992.05	889.43	603.69	604.81	618.62	352
	X6	530.10	1747.63	1462.12	1615.53	1870.21	1906
	X7	551.73	970.53	767.87	670.12	1050.56	841
	X8	257.00	68.47	571.17	352.80	530.39	319
	X9	556.09	424.59	803.94	349.67	725.22	566
	X10	586.34	-167.30	494.98	494.39	397.11	670

	Z	719714680.41	5602094.28	543124175.79	491853852.16	443060360.76	647292
Sample		7	8	9	10	11	12
Output	X1	1214.57	1449.40	943.32	1224.59	1361.67	1318
	X2	3780.37	5260.83	4291.77	4352.58	3667.46	4116
	X3	218.43	107.70	363.21	163.83	380.66	203
	X4	1422.47	958.63	1562.02	1332.29	1302.34	1157
	X5	557.33	111.45	744.66	606.76	218.68	494
	X6	1871.22	1021.87	1440.07	1818.29	1572.15	1441
	X7	1064.51	1493.72	1207.14	1286.42	1040.82	1200
	X8	366.10	133.30	227.21	261.36	436.71	522
	X9	377.16	688.20	452.61	663.90	530.76	465
	X10	800.12	512.20	551.44	494.43	390.76	757
	Z	492584057.02	506473067.89	544889238.90	467943818.38	569235072.62	570207
Sample		13	14	15	16	17	18
Output	X1	1100.21	1010.31	1035.72	1678.16	1128.14	1623
	X2	4288.42	4155.56	646.06	871.47	3114.32	2554
	X3	255.25	286.02	720.31	454.51	509.39	71.
	X4	2163.01	1429.21	2118.50	1493.55	1350.37	1274
	X5	365.36	405.90	1103.51	873.32	351.95	129
	X6	2716.46	1383.60	3556.38	2306.28	1643.50	1520
	X7	715.93	926.36	-960.40	-1275.43	954.34	1412
	X8	-58.57	313.88	383.31	175.37	364.76	211
	X9	600.53	503.38	870.25	900.56	753.27	715
	X10	233.23	656.08	590.01	496.57	955.28	894
	Z	2170185.68	665905385.44	341911291.58	655177408.64	899110807.72	746628
Sample		19	20	21	22	23	24
	X1	1078.19	1557.16	1000.87	-185.10	1050.93	655
	X2	2270.18	668.86	1408.04	5319.21	1756.63	2725
	X3	103.55	765.71	489.28	631.44	67.89	-107
Output	X4	1488.41	1429.76	1720.77	1417.55	1205.15	1311
	X5	547.66	673.66	468.78	172.02	345.12	114
	X6	1787.71	2786.70	1899.40	1994.00	2473.34	2154
	X7	1448.31	2542.74	1795.06	3534.94	1992.67	890
	X8	338.01	429.44	68.59	137.10	437.12	389
	X9	918.43	773.59	741.46	592.10	573.99	589
	X10	837.91	1259.24	1033.12	1205.70	1051.94	1175
	Z	792786882.35	433754759.88	723766094.21	896516146.90	753442448.93	907810
Sample		25	26	27	28	29	30
Output	X1	1094.44	1089.03	1052.90	1612.62	1137.63	1042
	X2	1656.44	1604.90	2023.58	-679.49	1974.15	7172
	X3	123.71	410.90	539.37	728.75	332.57	259
	X4	1344.70	1319.46	1479.49	1020.72	1466.60	1208
	X5	274.11	358.91	328.88	650.53	430.53	-221
	X6	2189.34	2179.33	2345.38	2690.53	2399.79	2469
	X7	1932.16	2099.57	1990.51	1742.77	1936.07	4000
	X8	562.30	318.66	403.69	384.85	131.15	-264
	X9	737.69	524.84	451.75	828.92	602.39	1295
	X10	971.11	815.15	675.08	994.46	887.71	828
	Z	738105308.01	797603735.83	645796319.39	701349244.64	550708066.75	320626

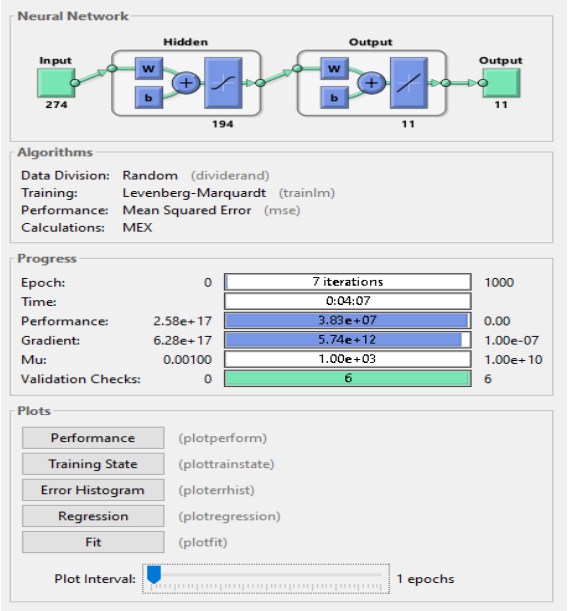

Fig.5: NN in the final image after completing the training process method of gravity method

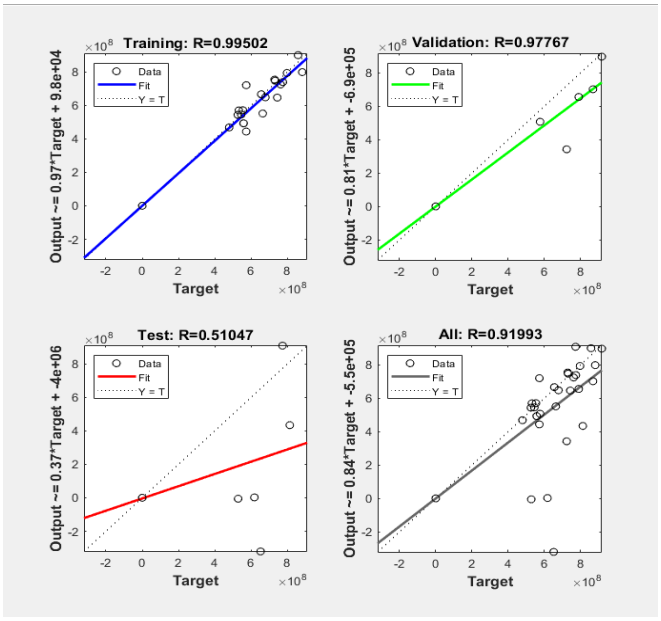

Fig.6: NN training regression of gravity method

In Figure (5) the simulation results of the central of gravity method, which depends on the coefficient of determinations value, which values between (0,1) to determine the acceptability of the simulation results in fuzzy linear programming. The simulation results of the central of gravity method were successful but did not reach the optimum level in decision-making, as the value of the coefficient of determinations ranged between (0.51-0.99), so to get better results we will repeat the training process for several attempts to reach more accurate results.

c. Pascal method

Pascal method is the method to fuzziness for removing fuzzy in fuzzy linear programming **[6]**. After the training process is completed for the data obtained from the pascal method, we have the neural networks as in Fig7 shows the neural networks in the final form after the completion of the training process of the pascal method.

Table 9: Represents the outputs of neural networks for the pascal method

Sample		1	2	3	4	5	6
Output	X1	978.71	-720.36	1263.51	1073.24	1063.45	1504
	X2	3366.14	3282.78	3399.75	3791.32	3416.00	3008
	X3	709.04	659.26	543.10	446.53	192.85	518
	X4	973.76	2254.27	1505.93	1181.73	1479.60	1479
	X5	514.29	1109.57	789.02	446.21	762.90	722
	X6	1309.72	1518.91	1505.76	1086.85	1370.57	1153
	X7	935.24	1003.49	665.23	738.92	732.58	887
	X8	345.40	387.55	149.75	205.35	378.40	-222
	X9	748.60	1504.45	687.36	860.19	482.64	1018
	X10	1053.37	833.98	447.14	122.00	720.06	532
	Z	636058169.23	431967131.63	709930165.30	610175742.83	546759974.04	608510
Sample		7	8	9	10	11	12
Output	X1	1139.81	1151.44	1621.48	1444.48	1107.78	1093
	X2	3360.22	3959.48	3445.13	2830.99	2622.12	3066
	X3	598.73	104.77	358.21	546.16	529.30	531
	X4	1359.03	1052.77	1501.11	1500.85	1670.61	1834
	X5	422.64	576.59	233.92	866.07	435.29	837
	X6	1661.21	1854.70	1337.31	753.84	1269.27	1186

	X7	1035.56	1042.73	1003.28	1077.15	806.85	860
	X8	364.13	22.82	126.55	146.10	281.26	537
	X9	475.18	484.20	746.74	813.92	835.15	1190
	X10	491.76	559.71	229.98	146.03	326.29	560
	Z	576425861.70	1072315393.01	618241025.22	670331705.76	634068209.78	616167
Sample		13	14	15	16	17	18
Output	X1	2932.43	1339.59	1137.15	178.71	1312.28	1423
	X2	3019.03	2541.37	2860.76	4118.52	2765.70	2960
	X3	294.73	643.21	585.36	630.78	313.34	461
	X4	1216.73	1709.15	1071.45	524.82	1600.54	1869
	X5	524.71	482.24	750.35	611.60	585.81	595
	X6	972.43	1164.88	1884.56	858.04	1133.01	1355
	X7	1582.34	857.60	897.25	1063.69	893.52	973
	X8	244.92	144.08	187.48	25.61	277.47	86.
	X9	799.87	896.78	672.86	852.11	651.21	627
	X10	655.34	620.82	907.25	192.89	347.92	617
	Z	601911998.66	510837245.33	351997616.42	552937539.99	475437320.59	445660
Sample		19	20	21	22	23	24
	X1	1695.36	2679.25	1603.67	695.24	1350.56	1620
	X2	2944.32	2776.86	3003.44	3896.45	3475.21	2729
	X3	532.33	158.93	354.06	483.01	433.23	447
Output	X4	1875.71	1382.27	1625.15	1601.71	1472.61	1589
	X5	276.87	558.49	817.26	183.65	769.44	1608
	X6	1367.88	1012.68	1126.30	1369.54	900.72	704
	X7	1135.12	1247.23	1225.87	1021.31	1034.06	1017
	X8	114.45	469.82	345.47	313.42	383.09	502
	X9	900.64	688.76	974.90	933.45	469.05	993
	X10	294.61	-73.76	271.33	245.41	493.26	858
	Z	457976708.47	636265753.36	550685228.25	124031335.78	503098928.10	180084
Sample		25	26	27	28	29	30
Output	X1	1463.75	1161.98	1067.97	3988.72	1088.51	-155
	X2	2737.39	2787.45	3113.86	2312.60	3025.49	6767
	X3	245.67	560.78	630.35	471.16	359.92	211
	X4	1498.27	1920.44	1468.19	1385.12	1207.45	614
	X5	598.21	477.51	572.71	440.31	604.88	192
	X6	838.29	1131.78	1703.00	364.91	1225.39	885
	X7	747.55	707.72	868.91	840.69	575.98	955
	X8	492.62	249.56	423.20	485.19	457.31	347
	X9	717.48	659.59	1183.51	1251.64	987.44	364
	X10	616.55	411.86	532.14	319.92	655.10	871

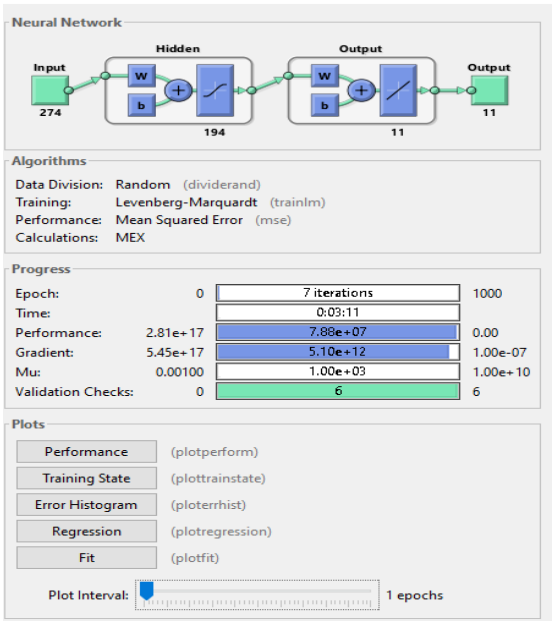

Fig.7: NNs in the final image after completing the training process method of pascal method

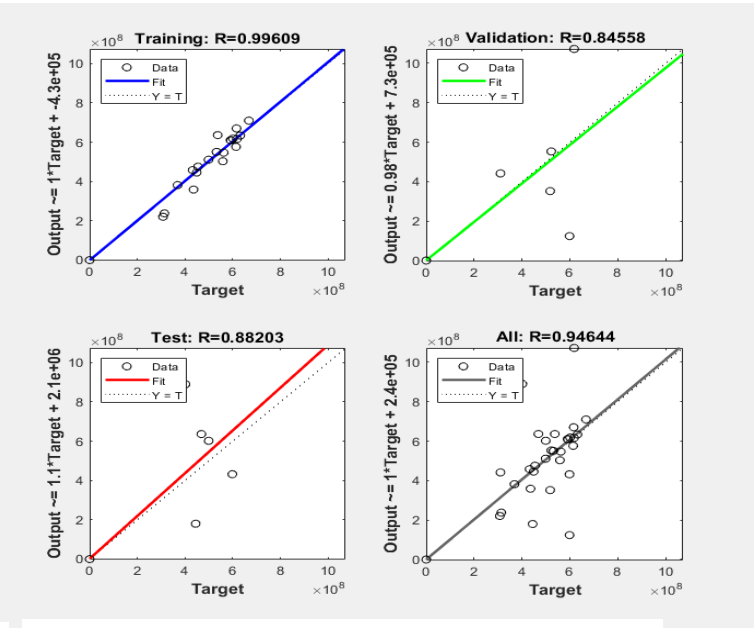

Fig.8: NNs training regression of pascal method

In Figure (8) the simulation results of the pascal method, which depends on the coefficient of determinations value, which values between (0,1) to determine the acceptability of the simulation results in fuzzy linear programming. The simulation results of the pascal method were successful but did not reach the optimum level in decision-making, as the value of the coefficient of determinations ranged between (0.88-0.99), so to get better results we will repeat the training process for several attempts to reach more accurate results.

d. Graded mean and integration method

Graded mean and integration method is the method to fuzziness for removing fuzzy in fuzzy linear programming **[6]**. After the training process is completed for the data obtained from the graded mean and integration method, we have the neural networks as in Fig.9 shows the neural networks in the final form after the completion of the training process of the graded mean and integration method.

Table (10): Represents the outputs of neural networks for the graded mean and integration method

Sample		1	2	3	4	5	6
Output	X1	1332.85	1354.18	958.32	726.44	779.90	789
	X2	2175.52	2354.65	2600.22	2992.02	2481.99	2733
	X3	705.90	585.11	441.26	521.62	440.88	311
	X4	-894.97	524.05	849.30	982.96	1017.86	1132
	X5	716.33	891.79	622.94	524.72	522.46	779
	X6	755.31	635.87	1303.48	1344.72	1409.32	1258
	X7	662.31	389.82	457.61	696.13	538.65	462
	X8	214.10	397.83	281.84	265.08	232.08	138
	X9	665.10	789.01	219.27	341.74	252.72	269
	X10	1589.39	1271.25	426.04	396.80	456.96	426
	Z	193831189.64	279049269.72	429594638.50	412464031.07	443579194.03	468463
Sample		7	8	9	10	11	12
Output	X1	922.38	798.17	835.92	766.94	888.72	809
	X2	2361.12	1322.87	1570.43	1349.40	4543.11	3774
	X3	494.42	548.01	466.15	420.90	543.88	196
	X4	967.41	1068.97	1010.96	1895.57	2096.35	1654
	X5	611.77	658.24	459.63	487.37	672.65	424
	X6	986.68	169.77	1200.90	1926.84	2169.47	1340

	X7	449.24	488.84	474.54	534.63	529.29	389
	X8	48.04	321.72	403.23	228.10	247.79	440
	X9	266.36	786.24	334.79	719.79	1442.59	672
	X10	507.36	949.10	602.37	876.97	573.69	301
	Z	387707204.44	101461572.03	292931765.04	418118228.42	263029687.71	195835
Sample		13	14	15	16	17	18
Output	X1	762.66	877.36	905.94	778.03	875.95	831
	X2	1475.32	1479.79	690.96	1514.86	1649.00	1629
	X3	427.95	448.79	124.96	556.09	394.35	564
	X4	1012.19	846.53	1096.56	1032.56	896.87	856
	X5	432.32	427.93	823.33	599.37	365.03	595
	X6	1495.26	1575.62	2368.06	1574.23	1386.99	1208
	X7	579.80	527.06	794.40	596.38	559.80	535
	X8	223.95	167.98	282.28	355.77	49.95	142
	X9	288.82	356.44	-523.90	169.12	332.29	156
	X10	699.89	732.68	1221.58	967.86	844.67	864
	Z	283412383.95	316072194.83	356843686.48	316219473.75	313321173.05	313363
Sample		19	20	21	22	23	24
	X1	700.67	747.93	813.00	785.50	765.36	864
	X2	1525.32	1672.77	-538.84	1897.24	1689.14	1611
	X3	578.04	573.20	1246.91	611.94	401.89	683
Output	X4	827.72	809.60	1265.79	615.74	621.61	423
	X5	478.39	520.20	568.67	577.86	351.64	378
	X6	1017.23	1103.70	1631.24	868.47	839.21	683
	X7	386.48	433.30	835.50	446.64	488.01	558
	X8	142.51	167.88	198.91	122.94	94.78	186
	X9	244.91	245.62	467.66	231.18	171.08	171
	X10	827.61	843.26	930.33	772.60	1040.08	1023
	Z	279721478.37	272200229.22	534293840.13	329700680.36	372776160.94	362964
Sample		25	26	27	28	29	30
Output	X1	816.87	590.14	954.69	1423.16	975.98	866
	X2	1467.12	-360.86	1554.11	2940.97	1452.07	1441
	X3	675.18	630.87	665.23	647.23	796.26	872
	X4	528.77	547.52	628.78	1084.52	705.87	656
	X5	406.95	701.80	359.56	263.31	549.78	503
	X6	615.83	-10.58	620.08	586.87	809.51	828
	X7	451.53	738.16	635.09	418.40	612.22	523
	X8	84.36	75.22	52.52	409.28	383.38	34.
	X9	199.13	-61.02	135.63	64.02	130.34	238
	X10	960.46	1396.17	1217.71	435.68	1033.62	1055
	Z	336517592.38	771726064.02	363462911.49	111051326.39	353831761.87	445311

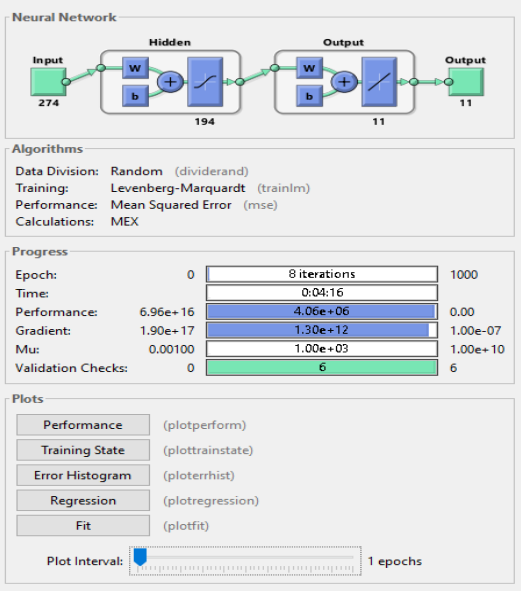

Fig.9: NNs in the final image after completing the training process method of graded mean and integration method

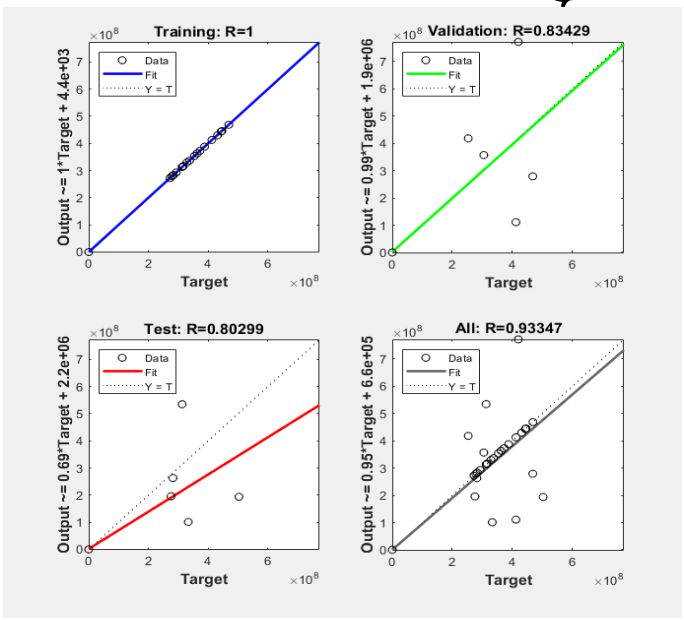

Fig.10: NNs training regression of graded mean and integration method

In Figure (10) the simulation results of the graded mean integration representation method, which depends on the coefficient of determinations value, which values between (0,1) to determine the acceptability of the simulation results in fuzzy linear programming. The simulation results of the graded mean integration representation method were successful but did not reach the optimum level in decision-making, as the value of the coefficient of determinations ranged between (0.80-1), so to get better results we will repeat the training process for several attempts to reach more accurate result.

e. Bounded and decomposition method

Bounded and decomposition method is the method to fuzziness for removing fuzzy in fuzzy linear programming **[5]**. After the training process is completed for the data obtained from the bounded and decomposition method, we have the neural networks as in Fig.11 shows the neural networks in the final form after the completion of the training process of the bounded and decomposition method.

Table (11): Represents the outputs of neural networks for the bounded and decomposition method

Sample		1	2	3	4	5	6
Output	X1	990.59	1103.00	1437.08	1367.32	-17.77	554
	X2	4833.16	-3381.03	2847.79	4116.45	4453.43	4778
	X3	1512.00	1299.47	1409.11	994.46	876.20	921
	X4	1543.27	1491.34	1945.29	1612.21	1552.10	2011
	X5	273.44	457.83	375.36	597.90	722.03	694
	X6	2049.95	2361.79	1755.11	2451.78	2871.29	2848
	X7	1108.36	834.42	1039.79	822.36	976.08	1002
	X8	526.18	496.35	660.14	896.40	758.79	733
	X9	777.08	689.08	530.43	377.02	744.25	712
	X10	881.33	1483.01	1115.13	1505.01	1355.92	1387
	Z	1112047179.71	743352648.46	978409479.61	843921534.12	801211783.12	892486
Sample		7	8	9	10	11	12
Output	X1	808.43	2014.56	969.83	416.05	424.75	1299
	X2	4458.52	-2747.11	4251.73	4750.32	5144.28	5487
	X3	681.42	536.36	880.67	-96.37	-123.79	219
	X4	1747.59	2224.99	1821.87	2093.33	1858.83	2288
	X5	322.11	166.81	679.72	878.23	418.01	91.

	X6	1765.83	3453.21	2679.90	1718.69	2255.17	2156
	X7	1047.58	1321.77	1516.73	1259.65	1319.83	1076
	X8	607.75	332.14	457.69	513.42	296.25	491
	X9	871.59	1015.68	769.29	1383.68	1393.33	1371
	X10	643.61	624.71	1188.08	1140.63	1214.06	385
	Z	881687868.02	1943043833.33	914906410.67	969394534.15	955040327.49	1046795
Sample		13	14	15	16	17	18
Output	X1	1622.84	1115.49	885.94	883.31	393.72	1481
	X2	6194.24	6291.57	20587.68	5124.50	6455.50	5708
	X3	556.89	1184.89	971.18	1742.77	1495.71	1213
	X4	2338.99	1466.10	208.68	1937.89	2596.24	2281
	X5	228.21	584.07	57.89	887.78	423.02	619
	X6	2229.41	1791.23	1407.79	463.16	2204.29	1994
	X7	948.02	995.38	890.82	1170.10	1332.13	1468
	X8	618.49	905.84	799.81	509.17	323.58	420
	X9	1403.63	909.27	34.81	794.43	1076.59	1192
	X10	330.60	326.80	-614.74	-163.30	92.26	38.
	Z	1291331349.28	1252947743.41	1108546389.54	1483452892.89	1556699729.19	137951
Sample		19	20	21	22	23	24
	X1	821.18	159.91	313.84	-211.38	880.62	841
	X2	4944.74	5572.12	5279.54	15107.27	7036.07	6127
	X3	996.26	832.27	1419.15	1617.64	909.26	813
Output	X4	1798.79	2339.48	2839.99	1755.77	2439.96	4339
	X5	765.73	563.38	706.32	1016.78	1038.13	1208
	X6	1301.07	2058.41	1393.03	126.78	1778.52	756
	X7	993.80	1358.03	1338.45	1416.65	1118.36	1367
	X8	687.47	524.87	466.15	323.70	454.56	367
	X9	432.47	660.08	-122.12	502.22	2098.24	1978
	X10	546.13	445.29	394.12	22.81	17.09	-213
	Z	1190713635.91	1309831560.83	1227348258.07	1673252634.56	1316066888.96	1227773
Sample		25	26	27	28	29	30
Output	X1	1835.32	985.84	12.55	587.69	386.79	406
	X2	9467.91	7338.19	6489.56	-3050.08	8094.59	6513
	X3	723.70	214.48	33.63	692.98	708.03	472
	X4	2363.80	2231.35	2901.31	3545.12	2881.97	3880
	X5	989.23	954.78	951.36	706.82	200.25	157
	X6	1754.25	2750.79	1527.45	1090.05	1437.73	762
	X7	1489.85	1856.04	1749.37	1717.75	1339.60	1275
	X8	386.26	285.15	124.02	5.06	54.60	-54.
	X9	1667.24	1447.14	1218.26	1284.33	1055.32	-425
	X10	530.09	244.33	714.42	905.96	676.52	135
	Z	1346993785.09	1108785792.53	1033244126.66	1429243451.77	941183291.09	703130

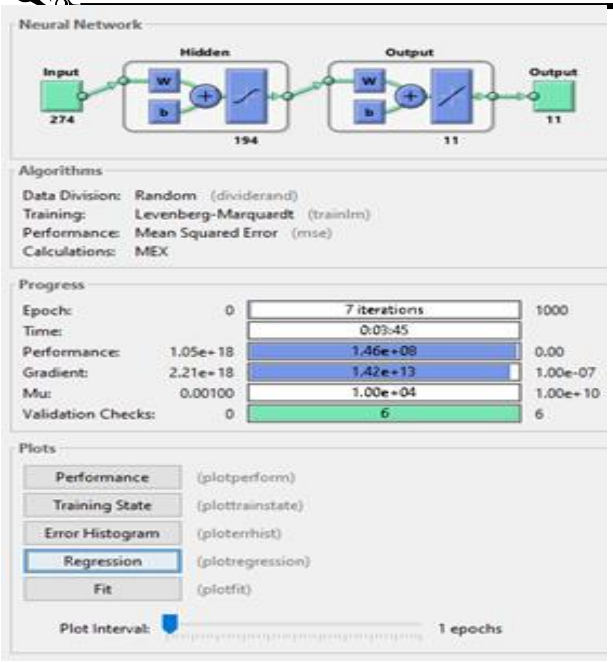

Fig.11: NNs in the final image after completing the training process method of bounded and decomposition method

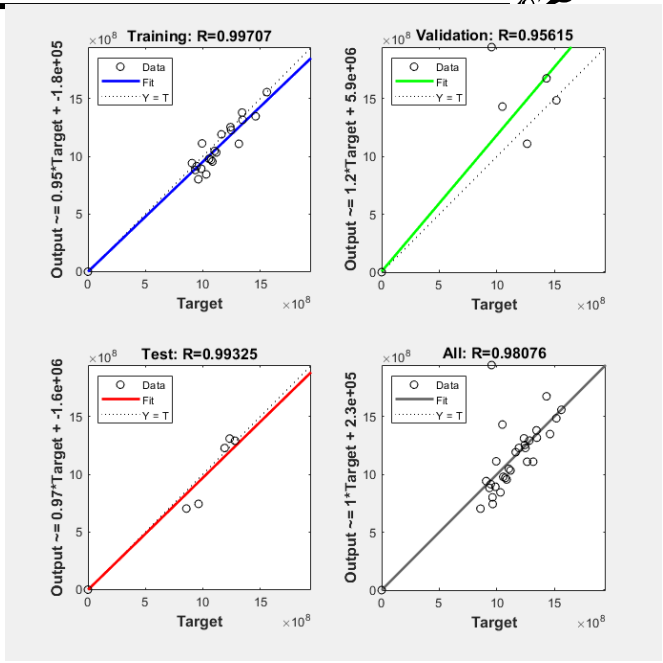

Fig.12: NNs training regression of bounded and decomposition method

In Figure (12) the simulation results of the bounded and decomposition method, which depends on the coefficient of determinations value, which values between (0,1) to determine the acceptability of the simulation results in fuzzy linear programming. The simulation results of the bounded and decomposition method were successful but did not reach the optimum level in decision-making, as the value of the coefficient of determinations ranged between (0.993-0.997), so to get better results we will repeat the training process for several attempts to reach more accurate results.

Table 12: The comparison between five different methods by back propagation NN training

	Training	Validation	Test	All	Superlative
Bounded and decomposition	0.99707	0.95615	0.99325	0.98076	1
Pascal	0.99607	0.84558	0.88208	0.94644	2
Graded mean and integration	1	0.83429	0.80299	0.93347	3
Central of gravity	0.99502	0.97767	0.51047	0.91993	4
Robust ranking	0.82772	0.91788	0.86305	0.83286	5

In table 12, shows the comparisons between five methods of fuzzy linear programming such as (Robust ranking, Central of gravity, Pascal, Graded mean, and integration, and bounded and decomposition) by using artificial neural networks. As a result, we realize that bounded and decomposition is the best method because the coefficient of determinations ranged between (0.993-0.997) and all test coefficient of determinations is (0.98).

8. Conclusion

Providing solutions to linear programming problems through the neural network approach is an interesting area of research. The results showed the efficiency of the fuzzy linear programming model, which was formulated to deal with problems of an unstable and fluctuating nature to reach the best possible solutions. The results showed that the highest maximum profit is (993423791) million dinars when using the bounded and decomposition method for the linear programming problem, which is higher than the profits achieved by the research company by (50%). Therefore, the use of this method by the liquidator will increase the percentage of profits from what is currently available by half, and by using ANN, show that the neural

network results when compared with five methods of fuzziness in a linear programming model, the best method to get maximum profit was up to 98% in a bounded and decomposition method increase from 993423791 IQD to 1943043833 IQD in a day. as it was found that by increasing the process of training the network, the results are better each time, as it appears through the results of the fifth experiment that it gave results close to the results reached in the fuzzy linear programming models. Thus, the implementation of the ANN in solving LP with a neural network will produce efficient results. such that the results showed that the bounded and decomposition is the best method because the correlation coefficient ranged between (0.993-0.997) and all test correlation coefficients is (0.98), this is what brings the company's total profits. The results were implemented by the software packages Python 3.9 and MATLAB R2019b.

9. Reference

1. Anderson , Dave , & McNeil , George,(1992) " Artificial Neural Networks Technology " , Contract Number F30602-89-C0082, Data & Analysis Center for Software .
2. Graupe, D. Principles of artificial neural networks, 2007.
3. Homayouni, N., & Amiri, A. (2011, January). Stock price prediction using a fusion model of wavelet, fuzzy logic and ANN. In International conference on e-business, management and economics (Vol. 25, pp. 277-281).
4. Isabels, K. R., & Uthra, G. (2012). An application of linguistic variables in assignment problem with fuzzy costs. International Journal of Computational Engineering Research, 2(4), 1065-1069.
5. Jayalakshmi, M., & Pandian, P. (2012). A new method for finding an optimal fuzzy solution for fully fuzzy linear programming problems. International Journal of Engineering Research and Applications, 2(4), 247-254.
6. Khadar, B., Rajesh, A., Madhusudhan, R., Ramanaiah, M. V., & Karthikeyan, K. (2013). Statistical optimization for generalised fuzzy number. Int. J. Modern Eng. Res, 3(2), 647-651.
7. Lv, C., Xing, Y., Zhang, J., Na, X., Li, Y., Liu, T., ... & Wang, F. Y. (2017). Levenberg–Marquardt backpropagation training of multilayer neural networks for state estimation of a safety-critical cyber-physical system. IEEE Transactions on Industrial Informatics, 14(8), 3436-3446.
8. Makwana, J. J., & Tiwari, M. K. (2014). Intermittent streamflow forecasting and extreme event modelling using wavelet based artificial neural networks. Water resources management, 28(13), 4857-4873.
9. Pandian, P., & Jayalakshmi, M. (2010). A new method for solving integer linear programming problems with fuzzy variables. Applied mathematical sciences, 4(20), 997-1004.
10. Reddy.V.R,Reddy.V.V,& Jaganmohan.V.C, (2018). Fuzzy Logic Controller (FLC) Implementation For Space Vector Pulse Width Modulated Induction Motor Drive. International Journal, 6(2),916-926"
11. Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., & Alves, S. R. (2017). Artificial neural networks: a practical course. Switzerland: Springer International Publishing.
12. Srinivas, H. K., Srinivasan, K. S., & Umesh, K. N. (2010). Application of artificial neural network and wavelet transform for vibration analysis of combined faults of unbalances and shaft bow. Adv. Theor. Appl. Mech, 3(4), 159-176.
13. Suzuki, K. (Ed.). (2011). Artificial neural networks: methodological advances and biomedical applications. BoD–Books on Demand.
14. Zhao, Q., Liu, Q., Cao, N., Guan, F., Wang, S., & Wang, H. (2021). Stepped generalized predictive control of test tank temperature based on backpropagation neural network. Alexandria Engineering Journal, 60(1), 357-364.