*International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-6, September 2015*
*ISSN: 2395-3470*
*www.ijseas.com*

# FEED -FORWARD NEURAL NETWORK ARCHITECTURE FOR SOLVING DOUBLE DUMMY BRIDGE PROBLEM IN CONTRACT BRIDGE

**M Dharmalingam[1] and R Amalraj[2]**

[1]Associate Professor, Dept of Computer Science, Nandha Arts and Science College, Erode-638052,
Bharathiar University, Tamil Nadu, India,
emdharma@gmail.com

[2]Associate Professor, Dept of Computer Science, Sri Vasavi College, Erode,
Bharathiar University, Tamil Nadu, India

**ABSTRACT**: Contract Bridge is an intelligent game, which enhances the creativity with multiple skills and quest to acquire the intricacies of the game, because no player knows exactly what moves other players are capable of during their turn. The Bridge being a game of imperfect information is to be equally well defined, since the outcome at any intermediate stage is purely based on the decision made on the immediate preceding stage. One among the architectures of Artificial Neural Networks (ANN) is applied by training on sample deals and used to estimate the number of tricks to be taken by one pair of bridge players is the key idea behind Double Dummy Bridge Problem (DDBP) implemented with the neural network paradigm. This paper mainly focuses on Elman Neural Networks (ENN) which is used to solve the Bridge problem by using Back-Propagation (BP) algorithm and Work Point Count System.

***Keywords:*** *Artificial Neural Network, Elman Neural Network, Back-Propagation algorithm, Activation functions, Contract Bridge, Double Dummy Bridge Problem, Work Point Count System.*

## 1. Introduction

The game of bridge is one of the well known card games played worldwide and drawing attention of many of the researchers and Computational Intelligence (CI) methods are normally applied to games and mainly focused on the aspect of learning in the game playing systems [1]. Artificial Neural Networks (ANN) are based on non-linear activation function approximations which make them suitable for most applications, since the outcome of the game could only be foreseen, but can't be stated earlier at any stage. However, the training is usually a cumbersome process and requires proper tuning of the learning algorithm, and in practice based on the knowledge and expertise acquired so far in the problem domain.

ANN are classified under a broad spectrum of Artificial Intelligence (AI) that attempts to imitate the way a human brain works and the Feed-Forward Neural Networks (FFNN) are one of the most common types of neural networks in use and these are often trained by the way of supervised learning supported by Elman Neural Network architecture using in Back Propagation algorithm. Many FFNN

were trained to solve the Double Dummy Bridge Problems (DDBP) in bridge game [2][3][4][5] and they have been formalized in the best defense model, which presents the strongest possible assumptions about the opponent. This is used by human players because modeling the strongest possible opponents provides a lower bound on the pay off that can be expected when the opponents are less informed. The new heuristics of beta-reduction and iterative biasing were introduced in the general tree search algorithm capable of providing consistent performance in the actual game. The effectiveness of these heuristics, particularly when combined with payoff-reduction mini-maxing results in an iprm-beta algorithm which makes fewer errors than the human experts and it represents the first general search algorithm capable of consistently performing at and above the expert level on a significant aspect of bridge card play [6].

A Point Count method and Distributional Point methods are the two types of hand strength in human estimators. The Work Point Count System (WPCS) is an exclusive, most important and popular system which is used to bid a final contract in Bridge game. Many of the neural network architectures are used to solve the double dummy bridge problem in contract bridge. Among the various networks, Elman neural networks architecture with supervised learning (ENN) is focused in this paper with Back-Propagation (BP) algorithm and the activation functions were used to train and test the data and results are compared.

## 2. Problem description

In bridge games, basic representation includes value of each card (Ace (A), King (K), Queen (Q), Jack (J ), 10, 9, 8, 7, 6, 5, 4, 3, 2) and suit as well as the assignment of cards into particular hands and into public or hidden subsets, depending on the game rules. In the learning course, besides acquiring these basic information, several other more sophisticated features need to be developed by the learning system [7][8].

### 2.1. The game of contract bridge

Contract bridge, simply known as bridge, is a trick-taking card game, where there are four players in two fixed partnerships as pairs facing each other [9] and referred according to their position at the table as North (N), East (E), South (S) and West (W), so N and S are partners playing against E and W. A standard fifty two pack is used and the cards in each suit rank from the highest to the lowest as *Ace (A), King (K), Queen (Q), Jack (J ),* 10, 9, 8, 7, 6, 5, 4, 3, 2. The dealer deals out all the cards one at a time so that each player receives 13 of them. The team who made the final bid will at the moment try to make the contract. The first player of this group who mentioned the value of the contract becomes the declarer. The declarer's partner is well-known as the dummy shown in Fig. 1.

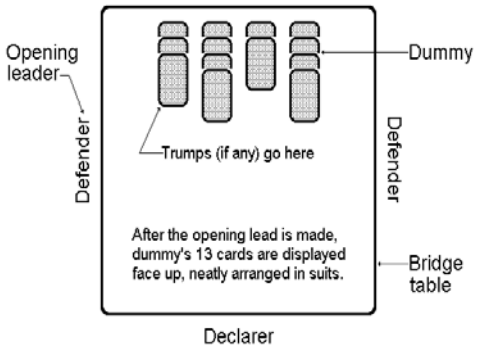


**Fig. 1.** A Bridge playing table

The player to the left of the declarer leads

to the first trick and instantly after this opening lead, the dummy's cards are shown. The play proceeds clockwise and each player must, if potential, play a card of the suit led. A trick consists of four cards and is won by the maximum trump in it or if no trumps were played by the maximum card of the suit led. The champion of a trick leads to the next stage and the aim of the declarer is to take at least the number of tricks announced during the bidding phase when the opponents try to prevent from doing it. In bridge, special focus in game representation is on the fact that players cooperate in pairs, thus sharing potentials of their hands [10].

To estimate the number of tricks to be taken by one pair of bridge players in DDBP, an attempt is made to solve the problem in which the solver is presented with all four hands and is asked to determine the course of play that will achieve or defeat at a particular contract. The partners of the declarer, whose cards are placed face up on the table and may be played by declarer. The dummy has few rights and may not participate in choices concerning the play of the hand and estimating hands strength is a decisive aspect of the bidding phase of the game of bridge, since the contract bridge is a game with incomplete information. This incompleteness of information might allow for many variants of a deal in cards distribution and the player should take into account all these variants and speedily approximate the predictable number of tricks to be taken in each case [11].

The fifty two input card representation deals were implemented in this architecture. The card values were determined in rank card (2, 3, K, A) and suit card (♠ (S), ♥ (H), ♦ (D), ♣(C)). The rank card was transformed using a uniform linear transformation to the range from 0.10 to 0.90. The Smallest card value is 2(0.10) and highest card value is A (0.90). The suit cards were a real number using the following mapping: Spades (0.3), Hearts (0.5), Diamonds (0.7) and Clubs (0.9).There were 52 input values and each value represented one card from the deck. Positions of cards in the input layer were fixed, i.e. from the leftmost input neuron to the rightmost one the following cards were represented: 2♠, 3♠, K♠, A♠, 2♥, A♥, 2♦, A♦, 2♣,. .. , A♣ Fig. 2. A value presented to this neuron determined the hand to which the respective card belonged, i.e. 1.0 for North, 0.8 for South, −1.0 for West, and −0.8 for East. The game then proceeds through a bidding and playing phase. The purpose of the biding phase is to identification of trumps and declarer of the contract. The playing phase consists of 13 tricks, with each player contributing one card to each trick in a clockwise fashion with another level bid to decide who will be the declarer. A bid recognizes a number of tricks and a trump suit or no-trump. The side which bids highest will try to win at least that number of tricks bid, with the specified suit as trumps. There are 5 possible trump suits: spades (♠), hearts (♥), diamonds (♦), clubs (♣) and "no-trump" which is the term for contracts played without a trump. After three successive passes, the last bid becomes the contract.

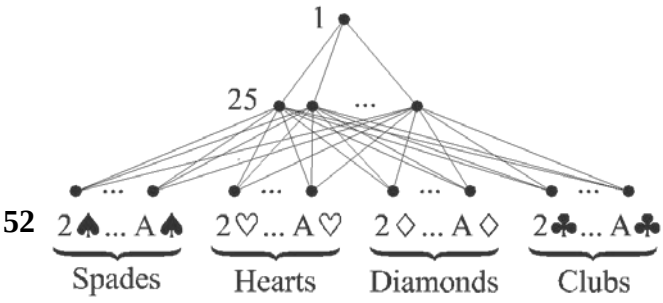

**Fig. 2.** Neural network architecture with 52 input neurons

Layers were fully connected, i.e., in the 52 − 25 − 1 network all 52 input neurons where connected to all 25 hidden ones, and all hidden neurons were connected to a single output neuron.

**2.2. The bidding and playing phases**

The bidding phase is a conversation between two cooperating team members against an opposing partnership which aims to decide who will be the declarer. Each partnership uses an established bidding system to exchange information and interpret the partner's bidding sequence as each player has knowledge of his own hand and any previous bids only. A very interesting aspect of the bidding phase is the cooperation of players in North with South and West with East. In each, player is modeled as an autonomous, active agent that takes part in the message process [12][13].

The play phase seems to be much less interesting than the bidding phase. The player to the left of the declarer leads to the first trick and may play any card and instantly after this opening lead, the dummy's cards are exposed. The play proceeds clockwise and each of the other three players in turn must, if found potential, play a card of the same suit that the person in-charge played. A player with no card of the suit may play any card of his selection. A trick consists of four cards, one from each player, declared won by the maximum trump in it, or if no trumps were played by the maximum card of the suit. The winner of a trick leads subsequently with any card as the dummy takes no active role in the play and not permitted to offer any advice or observation. Finally, the scoring depends on the number of tricks taken by the declarer team and the contract [14][15][16].

**2.3 No-trump & Trump-suit**

A trick contains four cards one contributed by each player and the first player starts by most important card, placing it face up on the table. In a clockwise direction, each player has to track suit, by playing a card of the similar suit as the one led. If a heart is lead, for instance, each player must play a heart if potential. Only if a player doesn't have a heart he can discard. The maximum card in the suit led wins the trick for the player who played it. This is called playing in no-trump. No-trump is the maximum ranking denomination in the bidding, in which the play earnings with no-trump suit. No-trump contracts seem to be potentially simpler than suit ones, because it is not possible to ruff a card of a high rank with a trump card. Though it simplifies the rules, it doesn't simplify the strategy as there is no guarantee that a card will take a trick, even Aces are useless in tricks of other suits in no-trump contracts. The success of a contract often lies in the hand making the opening lead. Hence even knowing the location of all cards may sometimes be not sufficient to indicate cards that will take tricks [17]. A card that belongs to the suit has been chosen to have the highest value in a particular game, since a trump can be any of the cards belonging to any one of the players in the pair. The rule of the game still necessitates that if a player can track suit, the player must do so, otherwise a player can no longer go behind suit, however, a trump can be played, and the trump is higher and more influential than any card in the suit led [18].

**2.4 Work Point Count System**

The Work Point Count System (WPCS) which scores 4 points for Ace, 3 points for King, 2 points for Queen and 1point for a Jack is followed in which no points are counted for 10 and below. During the bidding phase of contract bridge, when a team reaches the combined score of 26 points, they should use WPCS for getting final contract and out of thirteen tricks in contract bridge, there is a possibility to make use of eight tricks by using WPCS.

## 3. The data representation of GIB library

The data used in this game of DDBP was taken from the Ginsberg's Intelligent Bridge (GIB) Library, which includes 7,00,000 deals and for each of the tricks, it provides the number of tricks to be taken by N S pair for each combination of the trump suit and the hand which makes the opening lead[19]. There are 20 numbers of each deal i.e. 5 trump suits by 4 sides as No-trumps, spades, Hearts, Diamonds and Clubs. The term 'No-trump' is used for contracts played without trump in the four sides West, North, East and South.

## 4. Artificial Neural Networks

Artificial Neural Network consists of several processing units which are interconnected according to some topology to accomplish a pattern classification task. An Artificial Neural Network is configured for a precise application, such as pattern recognition or data classification through learning process. ANNs are non-linear information processing devices, which are built from organized elementary processing devices called neurons [20].

In Artificial Neural Network following the supervised learning, each input vector requires a matching target vector, which represents the desired output. The input vector along with the target vector is called training couple. In supervised learning, a supervisor is necessary for error minimization. Consequently the network trained by this method is said to be using supervised learning methodology. In supervised learning, it is assumed that the correct target output values are known for each input pattern [21].

## 5. Elman Neural Network Architecture

Elman neural network is a partial recurrent neural network model first proposed by Elman [22] It is a special kind of feed-forward neural network, which has extra local memory neurons and feedback loop. The Elman neural network is capable of approximating a nonlinear system without an explicit physical model. An Elman neural network has four kinds of layers Input layer, hidden layer, context layer and output layer [23] The context layer is utilized to constitute the back-forward loop, from which the hidden layer selects input. In comparison with other forms of feed forward neural network, the Elman neural network is sensitive to history of input data by this mechanism.

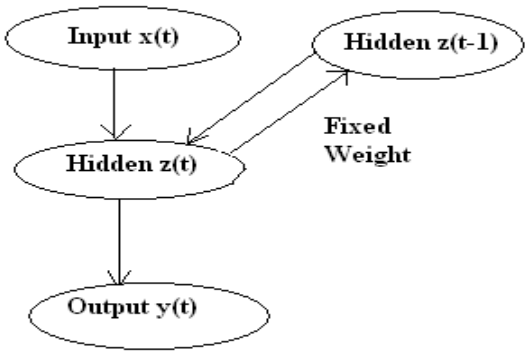

**Fig.3** Architecture of Elman Neural Network

In Fig.3 shows architecture of an Elman neural network, with the addition of a set of context units in the input layer. There are connections from the hidden layer to these context units fixed with a weight. At each time step, the input is propagated in a standard feed-forward approach, and then a supervised learning rule is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units.

## 6. The Back-Propagation (BP) algorithm

The Elman neural network is a widely used type of network architecture, Instead of just adjusting the weights in a network of fixed topology, with supervised learning. This network consists of an input layer, a hidden layer, an output layer and two levels of adaptive connections [24]. It is also fully interconnected, i.e. each neuron is connected to all the neurons in the next level. The overall idea behind back propagation is to make large change to a particular weight, '*w*' the change leads to a large reduction in the errors observed at the output nodes. The equation (1) let '*y*' be a smooth function of several variables $x_i$, we want to know how to make incremental changes to initial values of each $x_i$, so as to increase the value of y as fast as possible. The change to each initial $x_i$ value should be in proportion to the partial derivative of y with respect to that particular $x_i$. The equation (2) Suppose that '*y*' is a function of a several intermediate variables $x_i$ and that each $x_i$ is a function of one variable '*z*'. Also we want to know the derivative of '*y*' with respect to '*z*', using the chain rule.

$$\Delta x_i \propto \frac{\partial y}{\partial x_i} \tag{1}$$

$$\frac{dy}{dz} = \sum_i \frac{\partial y}{\partial x_i}\frac{dx_i}{dz} = \sum_i \frac{dx_i}{dz}\frac{\partial y}{\partial x_i} \tag{2}$$

The standard way of measuring performance is to pick a particular sample input and then sum up the squared error at each of the outputs. We sum over all sample inputs and add a minus sign for an overall measurement of performance that peaks at o.

$$P = -\sum_s \left( \sum_z (d_{sz} - o_{sz})^2 \right) \tag{3}$$

Where 'P' is the measured performance, S is an index that ranges over all sample inputs, Z is an index that ranges overall output nodes, $d_{sz}$ is the desired output for sample input 's' at the $z^{th}$ node, $o_{sz}$ is the actual output for sample input 's' at the $z^{th}$ node. The performance measure P is a function of the weights. We can deploy the idea of gradient ascent if we can calculate the partial derivative of performance with respect to each digit. With these partial derivatives in hand, we can climb the performance hill most rapidly by altering all weights in proportion to the corresponding partial derivative. The performance is given as a sum over all sample inputs. We can compute the partial derivative of performance with respect to a particular weight by adding up the partial derivative of performance for each sample input considered separately. The equation (4) each weight will be adjusted by summing the adjustments derived from each sample input. Consider the partial derivative

$$\frac{\partial P}{\partial w_{i \to j}} \tag{4}$$

where the weight $w_{i\to j}$ is a weight connecting $i^{th}$ layer of nodes to $j^{th}$ layer of nodes. The equation (5) our goal is to find an efficient way to compute the partial derivative of P with respect to $w_{i\to j}$. The effect of $w_{i\to j}$ on value P, is through the intermediate variable $o_j$, the output of the $j^{th}$ node and using the chain rule, it is express as

$$\frac{\partial P}{\partial w_{i\to j}} = \frac{\partial P}{\partial o_j}\frac{\partial o_j}{\partial w_{i\to j}} = \frac{\partial o_j}{\partial w_{i\to j}}\frac{\partial P}{\partial o_j} \qquad (5)$$

Determine $o_j$ by adding up all the inputs to node *'j'* and passing the results through a function.

$$o_j = f\left(\sum_i 0_i w_{i\to j}\right) \qquad (6)$$

Hence,
where $f$ is a threshold function. Let

$$\sigma_j = \sum_i 0_i w_{i\to j}$$

We can apply the chain rule again.

$$\frac{\partial o_j}{\partial w_{i\to j}} = \frac{df(\sigma_j)}{d\sigma_j}\frac{\partial \sigma_j}{\partial w_{i\to j}} \qquad (7)$$

$$\frac{\partial P}{\partial o_j} \quad \frac{df(\sigma_j)}{d\sigma_j} \; o_i \; \frac{\partial P}{\partial o_k} \qquad (8)$$

$$\frac{\partial P}{\partial w_{i\to j}} = o_i \quad \frac{df(\sigma_j)}{d\sigma_j} \quad \frac{\partial P}{\partial o_j} \qquad (9)$$

Substituting Equation (8) in Equation (5), we have

$$\frac{\partial P}{\partial o_j} = \partial w_{i\to j}\frac{df(\sigma_k)}{d\sigma_k} \quad \frac{\partial P}{\partial o_k} \qquad (10)$$

$$\frac{\partial P}{\partial w_{i\to j}} = o_i \quad \frac{df(\sigma_j)}{d\sigma_j} \; w_{j\to k} \quad \frac{df(\sigma_k)}{d\sigma_k}\frac{\partial P}{\partial o_k} \qquad (11)$$

Thus, the two important consequences of the above equations are, 1) The partial derivative of performance with respect to a weight depends on the partial derivative of performance with respect to the following output. 2) The partial derivative of performance with respect to one output depends on the partial derivative of performance with respect to the outputs in the next layer. The system error will be reduced if the error for each training pattern is reduced. The equation (14) and (15), Thus, at step's+1' of the training process, the weight adjustment should be proportional to the derivative of the error measure computed on iteration's'. This can be written as

$$\Delta w(s+1) = -\frac{n\partial P}{\partial w(s)} \qquad (12)$$

$$\left[\Delta w(s+1) = -n\frac{\partial P}{\partial w} + a\Delta w(s)\right] \qquad (13)$$

where η is a constant learning coefficient, and there is another possible way to improve the rate of convergence by adding some inertia or momentum to the gradient expression, accomplished by adding a fraction of the previous weight change with current weight change. The addition of such term helps to smooth out the descent path by preventing extreme changes in the gradient due to local anomalies. Hence, the partial derivatives of the errors must be accumulated for all training patterns. This indicates that the weights are updated only after the presentation of all of the training patterns.

## 7. Implementation

In this paper 20 sample data were used for training in MATLAB 2008a. Only one output neuron was used and in order to get the result, decision boundaries were defined the range of 0.1 to 0.9denoting particular number of tricks. The rank of the card was transformed using a uniform linear transformation to the range from 0.1 to 0.9 with biggest values to lowest values. The decision boundaries were defined a prior and target ranges for all possible

number of tricks from 0 to 13 were pair wise equal length. Gradient descent training function was used to train the data and gradient descent weight/bias learning function was used for learning the data. For training and learning the data, two activation functions viz., Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid functions were used. The results produced are represented in Table 1 and Table 2 respectively.

**Table1** Training deals sample 20

| S. No | Actual value in GIB | Calculated value in Log Sigmoid transfer function | Calculated value in Hyperbolic Tangent Sigmoid function |
|---|---|---|---|
| 1 | 0.75000 | 0.50537 | 0.56336 |
| 2 | 0.83000 | 0.76809 | 0.82961 |
| 3 | 1.00000 | 0.83654 | 0.99987 |
| 4 | 0.83000 | 0.62388 | 0.70788 |
| 5 | 0.75000 | 0.54815 | 0.70020 |
| 6 | 0.50000 | 0.50021 | 0.50224 |
| 7 | 0.58000 | 0.50046 | 0.50565 |
| 8 | 0.75000 | 0.54373 | 0.94029 |
| 9 | 0.50000 | 0.50018 | 0.50012 |
| 10 | 0.83000 | 0.84651 | 0.82685 |
| 11 | 0.58000 | 0.50195 | 0.57703 |
| 12 | 1.00000 | 0.83235 | 0.92176 |
| 13 | 0.58000 | 0.61464 | 0.57233 |
| 14 | 0.50000 | 0.52667 | 0.50134 |
| 15 | 0.91000 | 0.58635 | 0.90803 |
| 16 | 0.50000 | 0.78009 | 0.50317 |
| 17 | 0.50000 | 0.50010 | 0.50110 |
| 18 | 0.83000 | 0.50799 | 0.62475 |
| 19 | 0.66000 | 0.50299 | 0.65817 |
| 20 | 0.58000 | 0.53902 | 0.57061 |

**Table 2** Test deals sample 10 (Even)

| S. | Actual | Calculated | Calculated |
|---|---|---|---|
| No | value in GIB | value in Log Sigmoid transfer function | value in Hyperbolic Tangent Sigmoid function |
| 1 | 0.83000 | 0.94354 | 0.82368 |
| 2 | 0.83000 | 0.56256 | 0.71488 |
| 3 | 0.50000 | 0.70507 | 0.51046 |
| 4 | 0.75000 | 0.96712 | 0.86768 |
| 5 | 0.83000 | 0.64946 | 0.91212 |
| 6 | 1.00000 | 0.99577 | 0.99962 |
| 7 | 0.50000 | 0.56368 | 0.50053 |
| 8 | 0.50000 | 0.88346 | 0.59824 |
| 9 | 0.83000 | 0.88008 | 0.82525 |
| 10 | 0.58000 | 0.75167 | 0.57936 |

## 8. Results and Discussion

The results presented in the Fig 4 and Fig 5 shown that the comparison of target tricks along with Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid function. While comparing the train and test data along with target data, the result indicated that, train and test data shown significantly better results in both transfer functions, which minimized the total mean square.

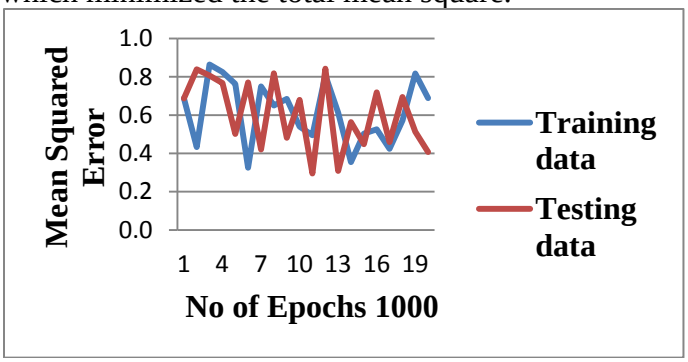


**Fig.4** Hyperbolic tangent function training deal sampling 1000 epochs

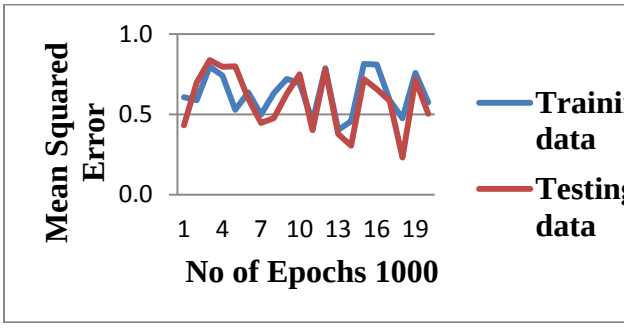


**Fig.5** Hyperbolic tangent function testing deal sampling 1000 epochs

The data trained and tested through this Hyperbolic Tangent Sigmoid function shows better performance and the time taken for training and testing the data were relatively minimum which also converged to the error steadily during the whole process.

## 9. Conclusion

ENN has a superior performance, concerning the capability of ENN to obtain the parameter easier to follow the real and the future data enhanced to take only relatively less time in order to reach minimum value. In Elman neural network architecture is used to solve DDBP. In the Work Point Count System, Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid functions were used in Back Propagation algorithm. Hyperbolic Tangent Sigmoid function produced better result when compared to Log Sigmoid transfer function which is used to bid a final contract. The Work Point Count System used in Back Propagation algorithm which produced better results and used to bid a final contract is a good information system and it provides some new ideas to the bridge players and helpful for beginners and semi professional players too in improving their bridge skills. Furthermore we would enlarge the new architecture to solve DDBP more efficiently and effectively.

## References

[1] J.Mandziuk, Knowledge-free and Learning – Based methods in Intelligent Game Playing. Springer, 2010.

[2] J.Mandziuk,K.Mossakowski, Neural Networks compete with expert human players in solving the double dummy bridge problem. In Proceedings of the Conference on Computational Intelligence and games, pp 117-124, 2009.

[3] K.Mossakowski,J. Mandziuk,Artificial Neural Networks for solving double dummy bridge problems. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh,(Eds.), Lecture Notes in Artificial Intelligence,LNAI:vol,3070. Artificial Intelligence and Soft Computing ICAISC, pp.915-921, 2004.

[4] M.Sarkar,B. Yegnanarayana,D. Khemani,Application of neural network in contract bridge bidding. In Proceeding of National Conference on Neural Networks and Fuzzy Systems,Madres, pp144-151,1995.

[5] M. Dharmalingam, R. Amalraj, Neural Network Architectures for Solving the Double Dummy Bridge Problem in Contract Bridge. In Proceeding of the PSG-ACM National Conference on Intelligent Computing, pp31-37, 2013.

[6] I.Frank,D.A Basin, Optimal play against best Defence: Complexity and Heuristics. In Lecture Notes in

Computer Science, vol, 1558, Germany, pp 50-73, 1999.

[7] H.Francis, A.Truscott, D. Francis, The Official Encyclopedia of Bridge, American Contract Bridge League,(5th) Edition.1994.

[8] H.W.Root, The ABCs of Bridge, Three Rivers Press, 1998.

[9] J.Mandziuk, Computational Intelligence in Mind Games, Springer, 2007.

[10] K.Mossakowski,J Mandziuk, Learning without human expertise: A case study of Double Dummy Bridge Problem. IEEE Transactions on Neural Networks, 20(2), pp 278-299, 2009.

[11] M.Dharmalingam, R.Amalraj, Supervised Learning in Imperfect Information Game. International Journal of Advanced Research in Computer Science,4(2), pp 195-200, 2013.

[12] B.Yegnanarayana,D. Khemani,M. Sarkar, Neural Network for contract bridge bidding. 21(3), pp 395-413,1996.

[13] A.Amit,S.Markovitch, Learning to bid in bridge. Machine Learning, 63(3), pp 287-327, 2006.

[14] S.J.J.Smith,D.S Nau,T.A Throop, Computer Bridge - A Big Win for AI planning. Artificial Intelligence Magazine, 19(2), pp 93-106, 1998.

[15] I.Frank, D.A Basin, A Theoretical and Empirical Investigation of Search in Imperfect Information Game. Theoretical Computer Science, 252(1), pp 217-256, 2001.

[16] T.Ando,Y. Sekiya,T. Uehara, Partnership bidding for computer bridge. Systems and Computers in Japan, 31(2), pp72-82, 2010.

[17] W.Jamroga, Modeling Artificial Intelligence on a case of bridge card play bidding. In Proceedings of International Workshop on Intelligent Information System, pp 276-277,1999.

[18] J.Mandziuk, Some thoughts on using Computational Intelligence methods in classical mind board games. In Proceedings of the International Joint Conference on Neural Networks, pp 4001-4007, 2008.

[19] M.L. Ginsberg, GIB: Imperfect Information in a Computationally Challenging Game Journal of Artificial Intelligence Research, vol. 14, pp 303-358, 2001.

[20] S.N Sivanandam and M Paulraj, Introduction Artificial Neural Networks, Chapter 5, pp 119-147, 2011.

[21] M. Dharmalingam and R.Amalraj, Artificial Neural Network Architecture for Solving the Double Dummy Bridge Problem in Contract Bridge, International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, issue.12, pp 4683-4691, 2013.

[22] J. Elman, Finding structure in time, cognitive science 14(3), pp 179-211, 1990.

[23] Y.Chen, W.Qi and J. Zhao, A new Elman neural network and its dynamics properties proceeding of international conference on cybernetics and intelligent system, pp 971-975, 2008.

[24] Rumelhart,D.E.,Hinton,G.E., Williams,R.J.(1986). Learning internal representations by error backpropagation, Parallel Distributed Processing: Ex-plorations in the Microstructure of Cognition, vol. 1, pp. 533-536.

## AUTHORS

**Dr. M. Dharmalingam** received his Under Graduate, Post Graduate, Master of Philosophy and Philosophy of doctored degrees from Bharathiyar University, Coimbatore in 2000, 2004 and 2008, 2015 respectively. He is working Associate Professor in Computer Science Nandha Arts and Science College at Erode and his research interest is Soft Computing.

**Dr. R. Amalraj** is an Associate Professor in Computer Science in the Department of Computer Science, Sri Vasavi College, Erode. He obtained his Ph.D in Computer Science from PSG College of Technology, affiliated to Bharathiar University, Coimbatore in 2003. He is a Life Member of Computer Society of India. He has been a Principal Investigator for a Minor Project sponsored by University Grants Commission, New Delhi. He has published several research papers in reputed National and International journals. His specific interests include Artificial Intelligence, Image Processing and Soft Computing.