World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:12, 2011

A Robust Al-Hawalees Gaming Automation using Minimax and BPNN Decision

Ahmad Sharieh and R Bremananth

***Abstract*—**Artificial Intelligence based gaming is an interesting topic in the state-of-art technology. This paper presents an automation of a tradition Omani game, called Al-Hawalees. Its related issues are resolved and implemented using artificial intelligence approach. An AI approach called mini-max procedure is incorporated to make a diverse budges of the on-line gaming. If number of moves increase, time complexity will be increased in terms of propositionally. In order to tackle the time and space complexities, we have employed a back propagation neural network (BPNN) to train in off-line to make a decision for resources required to fulfill the automation of the game. We have utilized Leverberg-Marquardt training in order to get the rapid response during the gaming. A set of optimal moves is determined by the on-line back propagation training fashioned with alpha-beta pruning. The results and analyses reveal that the proposed scheme will be easily incorporated in the on-line scenario with one player against the system.

***Keywords*—**Artificial Neural Network; Back propagation Gaming; Leverberg-Marquardt; Minimax procedure

I. Introduction

VISUALIZATION of gaming is an interesting research in the field of artificial intelligence. There are many popular games in the Sultanate of Oman, which the people inherited across the generations. These games represent the needs of the individual for physical activity and recreation for the spirit. Each game holds in itself the value and inheritance, intellectually and culturally, and has different forms. Numerous popular games under different names are practiced by men, women and children, which contribute to the leisure activation of physical and mental development. Al-Hawalees is one among the popular games known in several states in the Sultanate of Oman and world community of Omani. It is like a game of chess requiring skill and intelligence exercised by the people of the sea more than others especially on the beach. This is a board type game with two players. In this paper, we deal with the possibility of implementing Al Hawalees game using Minimax and BPNN decision. The Al-Hawalees game can be implemented such that one human player can compete with intelligence computer player. Fig. 1 shows the semantic flow of the proposed Al-Hawalees game. A back propagation neural network (BPNN) is utilized to estimate the resource requirement of the on-line game. For example, memory requirement and number of related moves are more essential for the gaming in order to take a decision against human player in the automation [5]. Next, the Minimax Game Tree is employed to find the shortest cost effective computation for each stage of the game. In the computation, however, the outcome of the Minimax, a minimum cost is also estimated using back propagation network to generate most latitude of winning the game against the human player [1] [2] [6] [7]. A game can be defined by the initial state that illustrates how the Al-Hawalees board is set up. The players define the legal moves of the Al-Hawalees. A terminal test and timer is fashioned to proclaim when the game is over. A utility or payoff function is an essential one which says who won, and by how much score and sort out rank of the players.

In Al-Hawalees game with ideal moves computation, in turn, the Minimax algorithm can determine the best optimal move for a player by in receipt of shortest path cost computation from BPNN and assuming that the opponent plays perfectly by enumerating the entire game tree. In addition, alpha-beta algorithm has been incorporated to prune away branches of the search tree that it can prove are superfluous to the final outcome [8][9].

The BPNN is mainly utilized for estimating the number of resources to optimally accomplish the game in the on-line scenario. Time and space complexities are predicted according to the number of seashells which are involved for the playing. After alpha-beta pruning, the unnecessary branches are removed and verified whether the shortest cost is obtained or not. If not, then the BPNN is called again to estimate the most possible resources.

The remainder of the paper is organized as follows. Section II describes criteria of the Al-Hawaless game. The BPNN based Minimax and the alpha-beta pruning of the Al-Hawalees game is enlightened in Section III. Section IV depicts the results and performance analyses. Concluding remarks are given in Section V.

II. Criteria and how to play the Al-Hawalees

Al-Hawalees game board consists of 28 holes. Each player takes 14 holes. The holes are distributed into four rows and there are seven holes in each row. The initial phase of the game is to fill two seashells in each hole. There will be 56 seashells in 28 holes. Each player has 28 seashells to move around the board. One of the players should start to pick up seashells from any one of the holes randomly and starts to place one seashell in each hole either in clockwise or anti-clockwise direction. The play is repeated until entire holes will be exhausted. Game will be over when one player's holes become without any seashell. In our paper, we form a set of rules which are required for the game and implementation has been carried out to play by a human player against computer player.

Ahmad Sharieh is Professor, with Information Systems and Technology Department, Sur University College, Affiliated to Bond University Australia, Sur, Oman (e-mail: ahmadsharieh@suc.edu.om).

Bremananth R is Asst. Professor with Information Systems and Technology Department, Sur University College, Affiliated to Bond University Australia, Sur, Oman (e-mail: bremresearch@gmail.com/bremananth@suc.edu.om).

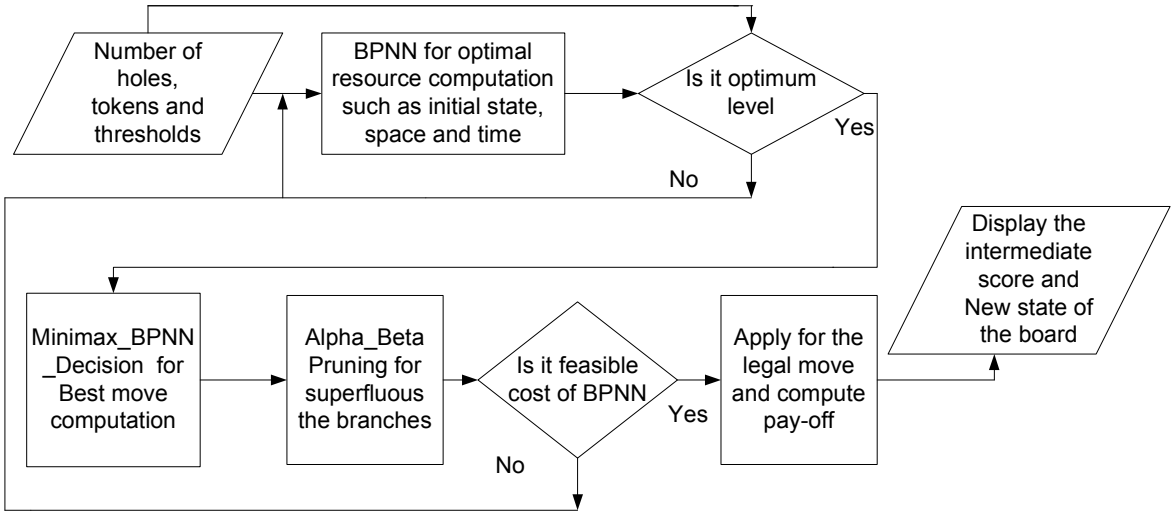

Fig. 1 A proposed semantic flow of the Al-Hawalees game with optimal moves

A. Rules of the Game

The principle of the game depends on moving seashells from one hole to one in left direction, for example, and eliminating opponent's seashells from the board [3][4]. To start playing the game, one of the two players has to transfer two seashells from a hole in the first row. There will be three numbers of seashells in each of the second and the third holes. Then, this player will take all seashells from the third hole, there will be three seashells in the first round, and distribute them into the next three holes. The same player will move the three seashells from the last holes and distribute them to the next three consecutive holes. This player will stop if the last shell was to be put or dropped in an empty hole. If this hole is in the second row, the seashells opposite to this hole in the side of the second player will be eliminated. Then, game comes to the second player turn. The second player will do similar as the first player. The process will be continued until the seashells of either player are completely eliminated from the hole. The player whose seashells entirely eliminated will be the loser and pay-off function generates -1. The other player will be the winner and utility function produces 1 to indicate the winning. There will be certain crucial chance of dice the game due diverse faulty moves of both players, the pay-off function will alarm as 0.

The seashells eliminated from the board are called Makool, which means they are removed from the game. If the last dropped seashell was in an empty hole and there is no seashell in the opponent side facing this hole, then this is called Albayute. This is when one player needs to stop playing and gives turn to his opponent. If the last dropped seashell was in an empty hole, in the second row, and an opponent seashell to be eliminated, then this called Akel. The game also depends on counting principles that means the player should know counting when his game will finish. An intelligent player should compute and know which hole to start with. The goal is to eliminate the opponent seashells before he/she eliminates the opponent's seashells. Fig. 2 illustrates the initial start of the game with Max and Min representations. Each player should start playing the game from first row, for example, and from either clock or anti-clockwise direction. Next time a player can play from any row also from any appropriate hole, but move seashells from right to left is an appropriate one.

A player may be chosen randomly to be first or second player. There are several possibilities of choosing the costing function. One possibility will be $f(n)=n,$ where n is the number of seashells to be purged (Akel) from opponent in each turn of the game. Another one is $h(m)=m,$ where m is the number of seashells that will be eradicated from player after he/she finishes his/her turns. A third one could be $d(n,m)=|n-m|$. If we choose $f(n)$, for example, the computer player will assess the values of $f(n)$ for picking all holes to play one by one and store the value of $f(n)$ corresponding to each move. The hole that leads to the highest value of the 14 values of $f(n)$ will be the first candidate to start the playing. In case of dice, the first with highest value of $f(n)$ will be picked; however, cost function value will be dependent on the estimation function of the BPNN.

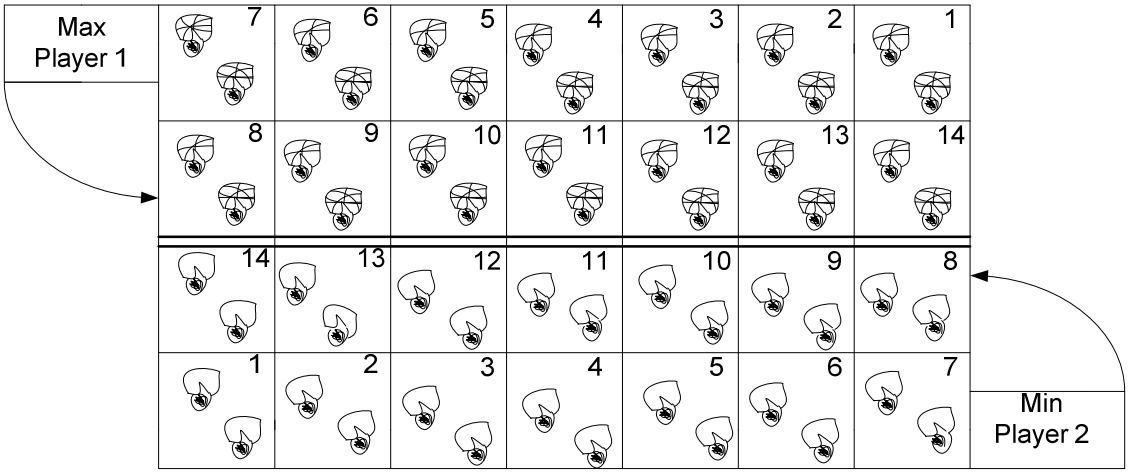

Fig. 2 A board positions of Al-Hawalees game, illustrates Max and Min players one and two, respectively

III. Proposed Methodology

The Minimax algorithm is designed to determine the optimal strategy for MAX [9], and thus to decide what the best first move is. In our proposed method, BPNN receives players' requirements in order to compute the robust resource planning. For that, we have utilized two-stage hidden layer network to train network in the supervised manner. That means, we predefined the various possible moves and requirement in an off-line manner then network will be trained to obtain an optimal state of requirements before starting the game. The initial state computation is an essential one to possibly win the game against the human player. We fixed in the game Max as the system player which plays against the Min, human player.

BPNN is the generalization of the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions [1]. In our gaming problem, input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by the players. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities. Standard BPNN is a gradient descent algorithm, as is the Widrow-Hoff learning rule [1], in which the network weights are moved along the negative of the gradient of the performance function. The term BPNN refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods [1]. Properly trained BPNN tends to give reasonable optimal values for initial state of the game when presented with inputs that they have never seen. Typically, a new input leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented.

Neural network training can be made more efficient if you perform certain preprocessing steps on the network inputs and targets. This section describes several preprocessing routines that you can use. The most common of these are provided automatically when you create a network. Network-input processing functions transform inputs into a better form for the network use. Processing functions associated with a network output transform targets into a better form for network training, and reverse transformed outputs back to the characteristics of the original target data. During the training, we have used the following preprocessing functions in order to obtain the robust estimation as in (1) and (2).

$$a = \text{sim}(\text{net}, p) \quad (1)$$

$$[m, b, r] = \text{postreg}(a, t) \quad (2)$$

The *sim* and *net* denote the simulation of neural network and the BPNN, respectively, and p represents number of input values to the BPNN. The t denotes target of the training. The m, b, and r represent slope of the linear regression, y intercept of the linear regression, regression value, respectively. $r = 1$ means perfect correlation. During training, we obtain values such as $m = 0.6640$, $b = 24.2165$, and $r = 0.8788$.

The network output and the corresponding targets are passed to *postreg* . It returns three parameters. The first two, m and b, correspond to the slope and the y-intercept of the best linear regression relating targets to network outputs. If there were a perfect fit (outputs exactly equal to targets), the slope would be 1, and the y-intercept would be 0. In this example, you can see that the numbers are very close. The third variable returned by *postreg* is the correlation coefficient (R-value) between the outputs and targets. It is a measure of how well the variation in the output is explained by the targets. If this number is equal to 1, then there is perfect correlation between targets and outputs. In the example, the number is very close to 1, which indicates a good fit.

Fig. 3 illustrates the analysis values provided by *postreg* for Al-Hawalees game. The network outputs are plotted versus the targets as open circles. The best linear fit is indicated by a dashed line. The perfect fit (outputs equal to targets) is indicated by the solid line. In this example, it is difficult to distinguish the best linear fit line from the perfect fit line because the fit is so good.

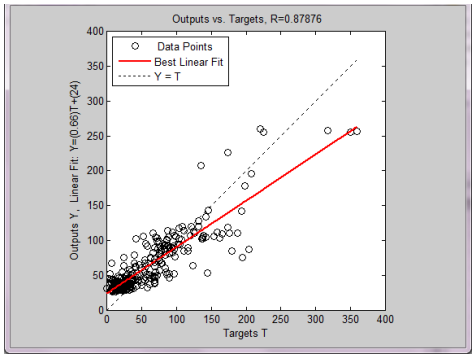

Fig. 3 Best Linear fit of the Al-Hawalees game

In order to tackle the time complexity, in this paper, we have employed a faster training algorithm called the Levenberg-Marquardt algorithm. It is designed to move toward second-order training speed without having to compute the Hessian matrix [1]. When the performance function has the form of a sum of squares, then the Hessian matrix can be approximated. Levenberg-Marquardt is performed as in (3) (4) and (5).

$$net = newff(p,t,3,\{\},'trainlm') \quad (3)$$
$$net = train(net,p,t), \quad (4)$$
$$y = sim(net,p), \quad (5)$$

where '*trainlm*' denotes the function of Levenberg-Marquardt and *newff* is the BPNN feed forward network.

A. Minimax and BPNN decision

Minimax function [10] is invoked after BPNN training is accomplished. The BPNN generates the optimal values for initial state to start the game based on the number of seashells, holes, and memory/space complexities of the game. Fig. 4 shows the pseudo code for the Minimax_BPNN_Decision and Minimax_value functions.

The following is the algorithm for the gaming

Step 1: Initial state will be estimated by the BPNN. It is trained based on the number of seashells, positions of the board, and threshold which are maintained by the player.

Step 2: After finding the set of optimal parameters to start the game, the whole game tree will be generated all the way down to the terminal states.

Step 3: Apply the utility function to each terminal state to get its value.

Step 4: Use the utility of the terminal states to determine the utility of the nodes one level higher up in the search tree.

Step 5: Continue backing up the values from the leaf nodes toward the root, one layer at a time.

Step 6: In due course, the backed-up values reach the top of the tree; at that point, MAX chooses the move that leads to the highest value. This is called the Minimax decision**,** because it maximizes them utility under the assumption that the opponent will play perfectly to minimize it.

Step 7: Cut off the search at some point and apply an evaluation function that gives an estimate of the utility of a state. Usually, it is not feasible to consider the whole game tree (even with alpha-beta).

Step 8: Most likelihood of the game can be handled by an extension to the Minimax algorithm that evaluates chance nodes by taking the average utility of all its children nodes, weighted by the probability of the child.

B. Optimal decision in the game

An optimal decision of Minimax is based on the BPNN training. However, Max therefore must find a contingent strategy, which identifies Max's move in the initial state. Then Max's moves in the states resulting from every possible response by Min. Then Max's moves in the states resulting from feasible response by Min to the moves. This process is repeated until the consistent move is expected to obtain for the player. An Optimal strategy leads to outcome at least as good as any other strategy when one is playing a fail-safe adversary. Fig. 5 shows a sample Minimax tree of the Al-Hawalees**,** that initial state is optimally computed by the BPNN. Player 1 treats as a Max player and player 2 takes as a Min player.

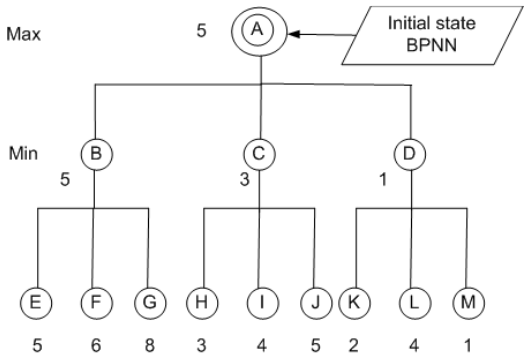

Fig. 5 A sample Minimax tree for the Al-Hawalees game with optimal values

If we assume that both players play optimally from there to the end of the game, then perceptibly, the Minimax value of a terminal state is just its utility function. Furthermore, given an option, Max desires to move to a state of maximum value, whereas Min prefers a state of minimum value described by (6).

$$Mini\max(s) = \begin{cases} Utitlity(s), & if\ Ter\min al\_test(s) \\ Max\ Mini\max(result(s,a)), & if\ player(s) = MAX \\ Min\ Mini\max(result(s,a)), & if\ player(s) = MIN, \end{cases} \quad (6)$$

Minimax_BPNN-Decision (Al-Hawalees) returns an operator

Invoke BPNN functions (1)-(5) to obtain the optimal state to start the game
for each *op* **in** ***operator [Al-Hawaless]*** do
value [op] = ***Minimax_value (Apply(op, Al-Hawalees), Al-Hawalees)***
end
return the *op* with the highest value[op]

Minimax_value (state, Al-Hawalees) returns a utility value

if Terminal_Test [***Al-Hawalees***].(state) **then**
return Utility*[Al-Hawalees].(state)*
else if ***MAX*** is to move in *state* **then**
return the highest ***Minimax_value*** of Successors(state)
else
return the lowest ***Minimax_value*** of Successors(state)

Fig. 4 Pseudo code for implementing Minimax_BPNN and Minimax_value functions.

where *s* denotes the states and *a* represents the state labels. The first Min node labeled B. It has three successor states named E, F, G with values 5, 6, 8 values respectively. So B's Minimax value is 5. Similarly, the other two Min states such as C and D have the minimum values of 3 and 1, respectively. The root node is a Max state whose successor states are B, C and D have the values of 5, 3 and 1 respectively. So that A has Minimax value of 5. We can also recognize the Minimax decision at the root that B is the optimal choice for MAX because it leads to the state with highest Minimax value. If the maximum depth of the game tree is l, and there are *b* legal moves at each point, then the time complexity of the Minimnax algorithm is $O(b^l)$. The algorithm is a depth-first search, although here, we employed the implementation through recursion rather than using a queue of nodes, so its space requirements are only linear in l and *b*. By using the alpha-beta pruning [7], we can examine only require nodes which are suitable for winning the game. So, order of complexity becomes $O(b^{l/2})$ instead of $O(b^l)$. This means that the effective branching factor of Al-Hawaless is $\sqrt{b}$ instead of b. Thus the factor becomes five instead of 25 in the case of five possible moves.

IV. Results And Performance Analyses

The proposed approach has been implemented in Visual Basic 6.0 and results were analysed using Matlab7.7. After optioning the number of holes and seashells that player wants to play, the BPNN is trained to get the feasible solution to achieve the game to the MAX's player. For this test, the BPNN with input neurons are specified unambiguously and weights are computed.

For example, if p is a set of input values and t is a set of target values that game wants to achieve after training, then we can specify the Feed-forward networks consist of N_1 layers using the dot operator weight function, *NETSUM* net input function, and the specified transfer functions. Hidden layer has received weights from the previous layer called Input layer both input and hidden layers have the biases. The final layer is the network output. Each layer's weights and biases are initialized. Adaption is done with *TRAINS* which updates weights with the specified learning function. Training is done with the specified training function. Performance is measured according to the specified performance function. Fig. 6 shows a sample training process using neural network training using Leverberg-Marquardt fast training for the BPNN. For example, *P* = [5 6 8 3 4 5 2 4 1] is a possible initial values of the Al_Hawalees game and target is specified as *T* = [2 2 3 4 3 5 1 8 6 4]. A network is created with one hidden layer of five neurons as, *net = newff(P,T,5)*.

The neural network is trained and plotted against the target to verify whether training process achieves the best possible move for obtaining MAX's as conqueror of the game or not. Training function can be described as *Y = sim(net,P)*; and plotted its trained values as shown in the Fig. 7. Fig. 7(a) is the default training process of the game and Fig. 7(b) illustrates after 50 epochs training close proximity with target of the optimal values which are possible require for the game.

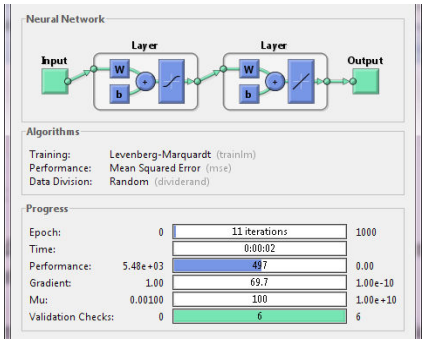

Fig. 6 BPNN based game training using Levenberg-Marquardt

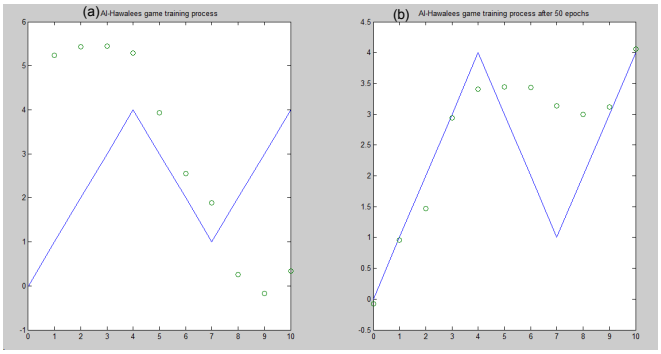

Fig. 7 Training process for the game by default and 50 epochs

A. Performance analysis

In performance analysis of choosing the initial and optimal moves of the game, best performance validation performance is attained at epoch 5. The train, validation, test and best are also shown in Fig. 8. Mean square errors between these parameters are computed for best fit of the gaming problem with respect to number of holes and seashells of the game. In order to attain rapid training process to get better time complexity as compare to the default training method, we have utilized Leverberg-Marquardt. This method achieves better possible computation of initial state fixation in the second epochs and best validation performance as 4.273 instead of 641.59 at epochs 5. This result reveals that Leverberg-Marquardt method is best suitable for gaming training approach in the estimation of initial state.

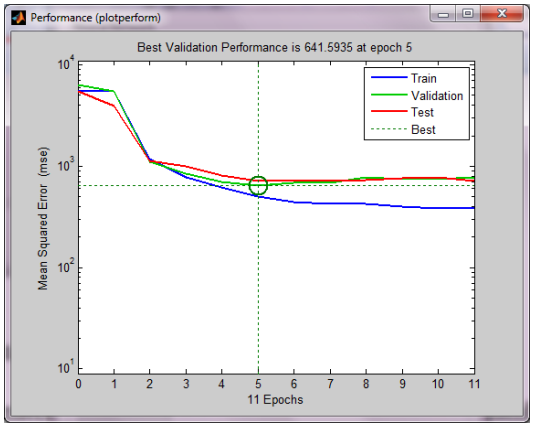

Fig. 8 Performace analysis of the game at epoch 5

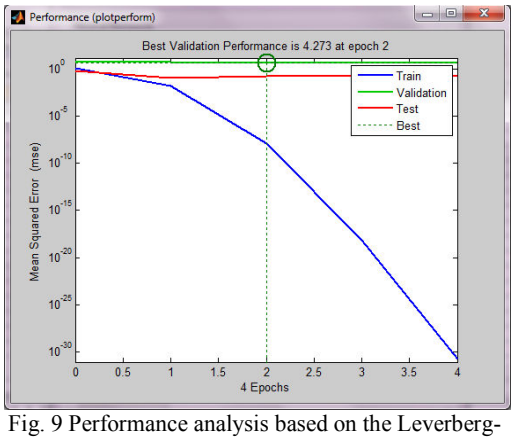

Fig. 9 Performance analysis based on the Leverberg-Marquardt fast training method

B. Training gaming state and Regression

Training game states are a computationally challenge process. The gradient, Mu (μ) and validation check are performed to illustrate the prominence of the proposed method. At the epoch 11, gradient = 69.66, μ = 100 and validation check = 6 were attained. This is shown in Fig. 10. These results show that the proposed rapid training process can effortlessly include in the on-line gaming. Our next test belongs to regression analyses of training, validation, testing and entire three processes. Training regression was 0.94037 for different target, validation regression was 0.879 and testing regression was 0.898. The overall performance of the gaming training was 0.918. Fig. 11 illustrates the regression process of the gaming.

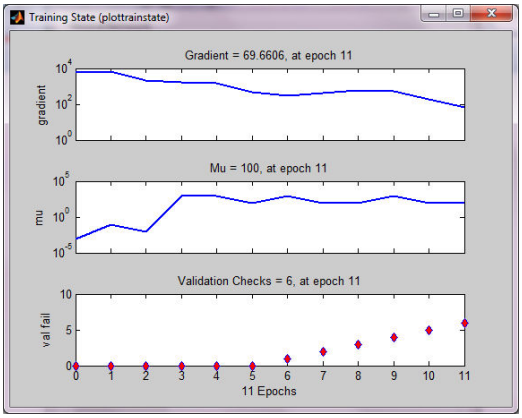

Fig. 10 Representation of game training states

C. Confusion matrices and Receiver operating characteristics

Confusion matrix is essentially used to analysis the performance variation between output and target classes. In our proposed method of training the game, we have performed three different confusion matrices such as training, validation and testing, and obtained the values 76.2%, 74.0 and 76.0, respectively. The overall performance of the confusion matrix was 75.7% as shown in Fig. 12. Receiver operation characteristics curve (ROCs) is primarily utilized to evaluate the end-user benefits due to the chance over of new methodology. In our game problem, we plotted the Training, Validation, and Testing ROCs with respect to false and true

positive rates. True and false positives are in favor of MAX and MIN players, respectively as shown in Fig. 13.

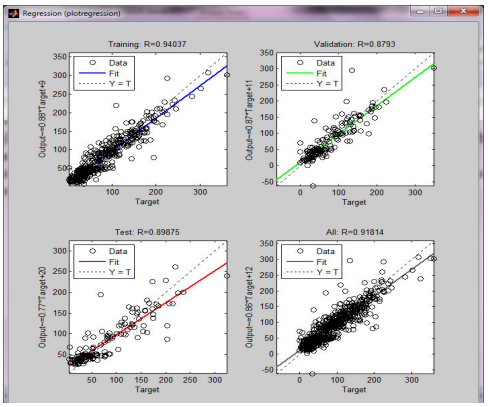

Fig. 11 Illustration of training, testing and validation regressions

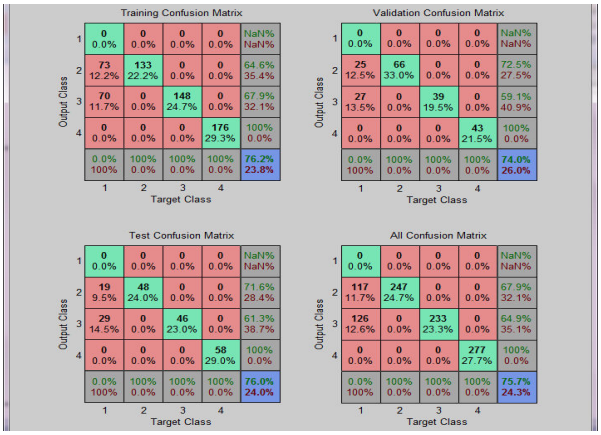

Fig. 12 Confusion matrices for the game

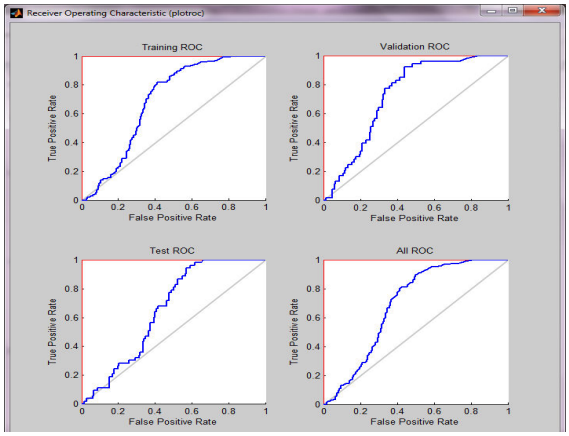

Fig. 13 Receiver operator characteristics curve of the game

V. Conclusion And Future Enhancement

In this research paper, an Artificial Intelligence based gaming issues are analyzed. We mainly dealt with the automation of a tradition Omani game named Al-Hawalees. We framed its related criteria and issues for the on-line implementation. Minimax_BPNN decision is incorporated to make a diverse nudges of the on-line gaming. In order to equip for the time, space complexities and initial state computation, we employed a BPNN to train in off-line to make a decision for resources required to fulfill the automation of the game. We utilized Leverberg-Marquardt training approach in order to obtain the speedy response during the gaming. The results and analyses based on regression, confusion, and ROCs disclose that the proposed scheme will be aptly incorporated in the on-line scenario with one player against the system. In our further research, an on-line training approach will be suggested to estimate the resources of the game in the multi-player environment with alliance perception in near future.

Acknowledgement

Authors thank their students Fawziya Al- Shamakhi, Marwa Said Al-Orimi, Ahlam Al-Alawi, Iman Al-Alawi Raba Al-Maskri, Salim Al-Alawi and Habiba Al-Salti for their effort to develop the game interface and colleagues for their continuous support and consent encouragement to do this research work successfully.

References

[1] Boykin,. S., *Neural A Comprehensive Foundation*, Prentice Hall, 2008.
[2] Bowling, M., Johanson, M., Burch. N., and Szafron B, "Strategy Evaluation in Extensive game, with Importance Sampling", *ICML-08*, 2008.
[3] Bryant, B. B. and Makulainen, R., "Acquiring visibly Intelligent behavior with example-guided nouroevolation" , *AAAI 07*, 2008.
[4] Bylander, T., "Complexity Results for Serial Decomposability", *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, San Jose, California, AAAI Press, pp. 729-734, 1992.
[5] Dechter. R. and Frost, D., "Backjump-based Backtracking for Constraint Satisfaction Problems", *Journal of Artificial Intelligence*, 136(2), 147-188, 2002.
[6] Ginsberg, N., "Imperfect Information in A Computationally Challenging Game", *IJAIR*, 14, 302-358, 2001.
[7] Kierulf, A., Chen, K., and Nievergelt, J., "Smart Game Board and Go Explorer: A study in Software and Knowledge Engineering", *Communications of the Association for Computing Machinery*, 33(2): 152-167, 1990.
[8] Kumar, V., Nau, D. S., and Kanal, L. N., "A General Branch-and-Bound Formulation for AND/OR Graph and Game Tree Search"., In Kanal, L. N. and Kumar, V., editors, *Search in Artificial Intelligence*, chapter 3, pages 91-130. Springer-Verlag, Berlin, 1988.
[9] Stuart Russell, Peter Norvig, *Artificial Intelligence A Modern Approach*, Pearson Education, 2010.

Ahmad Sharieh had two bachelor degrees: one in Mathematics and one in Computer Sciences. He had master degree in Computer Science and High Diploma in Teaching in Higher Education. He had PhD in Computer and Information Sciences from Florida State University 1991. Sharieh worked as Assistant Professor in Fort Valley College / USA and The University of Jordan (UJ) / Jordan. He worked as Associate Professor with Amman Arab University for Graduate Studies (AAUGS) / Jordan and The University of Jordan. He worked as Dean of King Adbullah II School for Information Techology/Jordan. Currently, he is a professor of Computer Sciences and Dean at Sur University College (SUC), Oman. He published articles in journals (27), in conferences (22), and authored and prepared books (14). He gained grant for eight research projects from UJ and Europe. He developed several software systems such as: Teaching Sign Language, e-learning Modeling and Simulation, and Online (Automated) Exams. He is on the editorial board of several journals and conferences and a referee of several others. His research areas are Distributing Systems, Expert Systems, E-Government, E-Learning, Parallel Processing, Pattern Recognition, Software Engineering, Wire/Wireless Communication, Modeling and Simulation.

Bremananth R received the B.Sc and M.Sc. degrees in Computer Science from Madurai Kamaraj and Bharathidsan University in 1991 and 1993, respectively. He obtained M.Phil. degree in Computer Science and Engineering from GCT, Bharathiar University, in 2002. He received his Ph.D. degree in 2008 from Department of Computer Science and Engineering, PSG College of Technology, Anna University, Chennai, India. He has completed his Post-doctoral Research Fellowship (PDF) from the School of Electrical and Electronic Engineering, Information Engineering (Div.) at Nanyang Technological University (NTU), Singapore, in 2011. Before joining NTU, He was a Professor and Head, Computer Science and Applications department, SREC, India. He has 18+ years of experience in teaching, research and software development. Currently, He is an Assistant Professor in the Information Technology department, Sur University College, Sur, Oman, affiliated to Bond University Australia. He received the M N Saha Memorial award for the best application oriented paper in 2006 by Institute of Electronics and Telecommunication Engineers (IETE). His fields of research are acoustic holography, pattern recognition, computer vision, image processing, biometrics, multimedia and soft computing. Dr. Bremananth is a member of Indian society of technical education (ISTE), advanced computing society (ACS), International Association of Computer Science and Information Technology (IACIT) and IETE. He can be reached at bremresearch@gmail.com.