A Rough – Neuro Model for Classifying Opponent Behavior in Real Time Strategy Games

M. Z. Rashad

Faculty of Computers and Information sciences,
Mansoura University, Egypt
Magdi_z2011@yahoo.com

ABSTRACT:

Real Time strategy games offer an environment where game AI is known to conduct actuality. One feature of realistic behavior in game AI is the ability to recognize the strategy of the opponent player. This is known as opponent modeling. In this paper, a classification Rough-Neuro hybrid model of the RTS opponent player behavior process is proposed. As a mean to achieve better game performance, reduction of the agent decision space and better high-level winning of real-time strategy games. The Rough-Neuro methodology allows the classification model to some extent simulate opponent behavior in playing RTS games. The methodology incorporates a two-stage hybrid mechanism. Rough sets for reduction of relevant attributes and artificial neural networks for classification opponent behavior during game playing. The proposed hybrid approach has been tested on an open source 3D RTS game called Glest. From our results we can deduce that the tactic may be successfully used for foretelling the demeanor of contender in the Glest game.

KEYWORDS:

Real-Time Strategy Games, rough sets, attribute reduction, opponent modeling, neural network.

1. INTRODUCTION

Real Time Strategy (RTS) [8][24] video game genre began to appear roughly two decades ago. RTS are games in which several players choose races and struggle against enemy factions by harvesting resources scattered over a terrain, producing buildings and units, and fighting one another in order to set up economies, improve their technological skill and level, and win battles, until their enemies are extinct. The better equation you get amidst economy, technology and force the more opportunity you have to earn. Many studies exist on learning to win games with comparatively small search spaces, while few studies exist on learning to win complex strategy games.

An important factor that influences the choice of strategy is the strategy of the opponent. For instance, if one knows what types of units the opponent has, then typically one would choose to build units that are strong against those of the opponent [3].

DOI : 10.5121/ijcsit.2012.4515

To make classifications about the opponent's strategy whether he is defensive or offensive, an AI player can establish an opponent model. Many researches mention the importance of designing the contender's tactic and illustrate that contender samples are grudgingly needed to handle the complexities of state of the technique video games [9][10].

Opponent modeling is an important research area in game playing, it concerns establishing models of the opponent player, and utilizing the models in actual play. In general, an opponent model is an abstracted description of a player [21] or of a player's behavior in a game. The goal of [15] opponent modeling is to improve the capabilities of the artificial player by allowing it to adapt to its opponent and exploit his weaknesses. Albeit a game theoretical best settlement to a game is known a computer program which has the ability to design its conductor's demeanor may gain a higher prize.

Setting up effective conductor samples in RTS games is a specially challenge that is why the reduction of perfect information of the game environment. In traditional board games the entire [5][21] board is being seen to the player; a player can notice all the actions of the contender. Hence; estimating the contender's strategy and setting up a contender sample is potential by default, for proposal by using case based reasoning techniques.

In RTS games, however, the player has to deal with imperfect information Typically, [16] the player can only observe the game map within a certain visibility range of its own units. This renders constructing opponent models in an RTS game a difficult task.

Research specially focused on the topic of opponent-modeling search started in 1993, Carmel and Markovitch [5] investigated in depth the learning of models of opponent strategies. While ,Iida et al. [12] investigated potential applications of opponent-model search.

In the year 1994, Uiterwijk and Van den Herik [17] invented a search technique to speculate on the fallibility of the opponent player.

In the 2000s, Donkers et al. [10][11] defined probabilistic contender samples, which seek to avoid the traps of contender designing by insertion the player's uncertainly about the contender's strategy.

Houlette [21], Charles and Black [6], Charles et al. [7], and Bohil and Biocca [3] discussed the challenges of opponent modelling in game environments, and suggested possible implementations of opponent modeling. A challenge for contender designing in games is that samples of the contender player have to be set in a comparatively realistic and complex game environment with an ideal little time for noticing and often with only partial noticeable of the environment.

While, Rohs [18] was able to model accurately the preferences of opponent players in the game Civilization. Van der Heijden et. al. [19] applied opponent modeling in RTS games to increase the effectiveness of strategies in a simple game mode of the ORTS game. For adding, researchers inserted techniques to foretell successions of user actions as the place of contender players in first person shooters [4] [23]and in the game Warcraft [2].

So in this paper we will try to predict opponent's strategy through modeling opponent behavior in an imperfect-information RTS-game environment. We will introduce a proposed hybrid rough-neuro model applied on Glest RTS game and implemented using Rosetta and Matlab applications. Our model will try to reduce uncertainty in predicting opponent behavior through using neural networks (NN) predicting capabilities and rough set abilities in solving NN problems.

The paper is organized as follows. In Section (2) we will present a brief overview of rough set and neural networks. In Section (3) a detailed description of our proposed Rough-Neuro model. Model results will be in Section (4) and finally in Section (5), conclusions are drawn.

2. ROUGH SET- NEURAL NETWORK: A BRIEF OVERVIEW

2.1 Rough Set

RS theory is to some extent a new smart technique for managing uncertainly that is used for the detection of data dependencies in order to estimate the importance of features to find out types in data, to minimize repetitions [25][26][27] and to realize and grade objects. Over and above, it is being used for the extraction of basis from databases where one vantage is the induction of readable if then bases. Such bases have a possibility to detect previously unfound types in data. On the other hand, it also collectively tasks like a classifier for invisible samples.

Unlike other computational smart techniques, rough set analysis requires no outer parameters and uses only the information offered in the personal data. One of the fine advantages of rough set theory [1][26] is that it can tell if the data is complete or not count on the data itself. Whether the data is imperfect, it will offer that more information about the objects is wanted.

Moreover, whether the data is full rough sets are capable of detection if there are any repetitions in the data and discover the least data needed for rating. This matter of rough sets is very important [20][27] for applications where sphere knowledge is very bounded or data collection is costly painful because it makes sure the data collected is just enough to set a good ranking sample without victimization the fineness or losing time and effort to collect additional information about objects.

2.2 Neural Networks

Artificial neural networks (ANN) are computational models of nervous systems. A neural network is a system composed of many simple processing elements called neurons operating in parallel whose function is determined by network structure, connection [22] strengths, and the processing performed at computing elements or nodes. Neurons are groued into layers or slabs. The neurons in each layer are the same type. Each neuron is connected to other layers by means of interconnections or links with an associated weight. The behaviour of an ANN depends on both the interconnections and the input-output function (transfer function) that is specified for the units.

Neural networks is trained rather than programmed. They have perfect ability to learn the connection between input/output designation from a given dataset without any information or proposition about the statistical apportionment of data [22][29]. This ability of learning from data without any advanced information makes spooky networks especially for classification favorable and regression functions in workable situations and in most fiscal and manufacturing applications. The neurons in every category are the same pattern.

Neural networks are also originally non-linear that makes them more workable and precise in designing complex data types as contrary to many classical ways which are linear.

3. ROUGH-NEURO OPPONENT MODELING

A lot of algorithmic and architectures ways have been given to design contenders. Some of them are so complicated or expensive to perform. It still unsolved problem to discover workable solutions.

Here we focus on a particular method well suited for predicting opponent behavior in RTS games. Neural networks is one of the most popular techniques in classification that can solve opponent modeling and reduce decision space of AI player.

But there are issues related to using neural networks in RTS games as time, storage and training cost required in neural networks of a higher input dimensions [we had 13 inputs to model opponent in Glest]. Neural Networks also have a slow learning rate, Complex network structure and umbiguous meaning of network.

But we mentioned in section 2.1 that rough set has the ability to evaluate the importance of attributes and to reduce them. The result of attribute reduction called *reduct* [14]which gives the same result as the whole attributes do. So we tried to make benefit of attribute reduction based rough set and neural networks classification cabapilities. Using rough set we can reduce number of arributes and so reduce the amont of data which solves the neural networks highly time and storage training cost. So we proposed a Rough-Neuro model consists of two main steps, Figure (1):

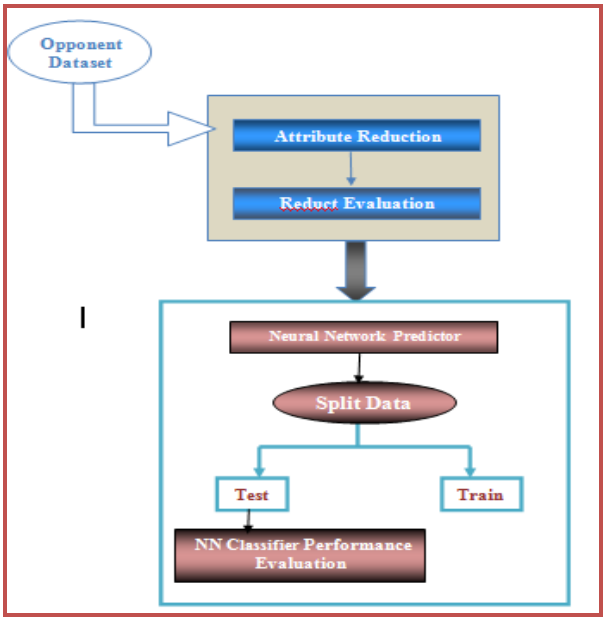

Fig.(1): Opponent Modeling Rough-Neuro Model

- Rough Set Phase
- Neural Network Classifier Phase

We applied these two steps of our model on opoonent features extracted from a 3D RTS games called Glest [24][28] where you control the armies of two different factions: Tech, which is mainly composed of warriors and mechanical devices, and Magic, that consists of mages and summoned creatures in the battlefield. Glest is not just a game, but also a Free Software (cross-platform), fully modifiable engine to create strategy games based on XML and a set of tools.
We had 13 feature extracted from Glest we used in modelling opponent and its behavior especially strategic behavior, e.g., the opponent's preference of unit type, the focus of an opponent's technological development, the strength of his economy, and the aggressiveness of the opponent, can generally be inferred from observing the feature values during actual play. As Glest is a typical RTS game, the defined features may be generalized to similar strategic games. The 13 defined features are given in table (1).

3.1 Rough Set Phase

3.1.1 Attribute Reduction

There often exist conditional attributes that do not provide “almost” [5] any additional information about the objects. These features need to be removed to minimize the complexity and value of decision process, but discovering all the reducts [13] is NP-complete but luckily in applications it is always not needful to discover all of them , one or a few of them are enough
Generally, rough set theory presents helpful techniques to minimize beside the point and superfluous features from a large database with lots of features. The dependency degree (or approximation quality, classification quality) and the information entropy are two most common attribute reduction measures in rough set theory. Table (1) displays a sample of opponent dataset and two reducts of it.

3.1.2 Reduct Evaluation

Selecting the best reduct is important. Choosing the best reduct is significant; the selection is based on the ideal standard with the features. If a cost task could be assigned to features then the selection can be depended on the collective least cost standards. In this paper, we adopt three criteria [3].

Table (1): A Sample of Opponent Data set and two different reducts

# of Units	Gold(a)	2000	500	350	150
	Wood(b)	300	400	300	300
	Stone(c)	1500	1000	2000	1500
	Food(d)	30	60	30	60
	Castle(e)	0	1	1	0
	defense_tower(f)	2	3	0	1
	Worker(g)	2	3	1	1
	Swordman(h)	1	1	3	3
	Archer(i)	1	0	3	2
	Guard(j)	3	2	1	1
	battle_machine(k)	1	2	4	3
	Armor(l)	40	100	40	25
	Sight value(m)	15	10	25	30
Type (s)		Deffinsive	Deffinsive	Offensive	Offensive

# of Units	(a)	2000	500	350	150
	(f)	2	3	0	1
	(g)	2	3	1	1
	(h)	1	1	3	3
	(i)	1	0	3	2
	(j)	3	2	1	0
	(k)	1	2	4	3
(s)		Deffinsive	Deffinsive	Offensive	Offensive

# of Units	(a)	2000	500	350	150
	(c)	1500	1000	2000	1500
	(e)	0	1	1	0
	(f)	2	3	0	1
	(h)	1	1	3	3
	(i)	1	0	3	2
	(k)	1	2	4	3
(s)		Deffinsive	Deffinsive	Offensive	Offensive

Cardinality means the number of attributes of the reduct. The less the cardinality the better the reduct. But the cardinality measure alone did not shrink our list of available reduction algorithms a lot so we needed more measurements.

- Number of generated rules: it uses the same concept as the reduct cardinality, i.e. the less the number of rules the better [14].
- Support: is a total number of correctly classified objects divided by total number of objects to be classified (that is size of a training set). Hence, the highier the support the better the reduct.

3.2 Neural Networks Classifier Phase

In an attempt to validate the feasibility of the reduct, a classifier based on a feed forward neural network (FFNN) was implemented A series of experiments was performed in order to determine the best possible configuration, i.e., network's structure, activation functions and training algorithms.

The main goal of this stage of the project was to perform a preliminary analysis and comparison of the behavior of the FFNN-based classifier that uses only those inputs that were included in the RS-derived reduct.

A feed forward neural network with one hidden layer was used. The number of neurons in the layer varied from the ½ of the number of inputs to the double number of inputs + 1. The training algorithm that proved to be the most effective during preliminary experiments, and was used in final computations was the Levenberg-Marquardt backpropagation algorithm.

$$MSE= 1/n \sum_{i=1}^{n} (P^i_e - P^i_a)^2$$

where P^i_e is the success probability of the ith output estimated by the Levenberg-Marquardt neural network, P^i_a is the actual probability of the ith output, and n is the total number of proposals in the testing set. Logsig activation function was used, which is usually performed better than others.

$$f(x)= \frac{1}{1+ \exp(-x)}$$

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Rough Set Phase

4.1.1 Attribute Reduction

Using RSs attributes reduction algorithms programmed in Aleksander Øhrn ROSETTA [4], we tried to reduce the number of measured attributes through applying them on each of the eight reduction algorithms, generating each algorithm reducts and rules. The data is available as Microsoft Access database, loaded in ROSETTA as an ODBC [4] and applied in each reduction algorithm.

4.1.2 Reduct Evaluation

As we mentioned previously in section 3.1.2 that reduct evaluation is performed according to three criteria , we can see from table (2) that ManualReducer has the lowest number of rules but it also has the lowest number of support which eliminats it from our list of choice. After ManualReducer elimination we found that JohnsonReducer has the lowest no of rules and cardinality and the highest support. Which tells us that JohnsonReducer is the best choice for our system .

Table (2): The evaluation measurements of reducts and rules

No.	Reduction algorithms	No. of reducts	No. of rules	*cardinalities* of reduct	Support of reduct
1	SAVGeneticReducer	21	450	5,6,7	89
2	JohnsonReducer	1	21	5	100
3	Holte1RReducer	8	61	6	35
4	ManualReducer	1	15	5	1
5	RSESDynamicReducer	64	982	6,7,8	1,59,52,24
6	RSESExhaustiveReducer	21	553	5,6,7	1
7	RSESJohnsonReducer	5	42	5,7	1
8	RSESGeneticReducer	17	348	6,7,8	1

Johnson's algorithm is a typical greedy algorithm with a natural tendency to find a single prime implicant of minimal length. In our experiment it provides only one reduct with five attributes (the amount of gold , the number of defense_tower units, , the number of sowrdman units, the number of archer units, , the number of battle-machine) that gives the same decision as the whole 13 attributes. It provides a reduction method that is fast and efficient with the minimum overhead, and using Rosetta software makes it faster, easier and even more efficient.

Let A be a "universal " set of n elements, $=\{S_1,S_2,\ldots S_k\}$ a collection of subset U formaing a cover for it, and c:$\rightarrow Q^+$ a cost function . Johnson´s approximation
algorithm finds a sub-collection of $ covring all the elements of U at minimal cost
Input : C =0, T=0
Output : T

1. Step1 let C=0 , T =0
2. Step2 while C≠U do
   - Find S € $ such that c(S)\|S\C| is minimum
   - ¥ x € s, define cost(x)= C(S)\ |S\C|.
   - C←C U S , T ←C U {S}
3. Step3 Result = T

Figure 3 : JohnsonReducer algorithm

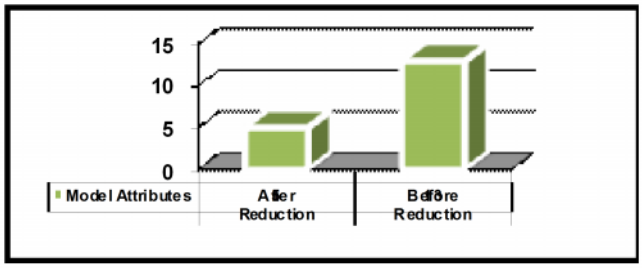

Figure 4: displays the number of attributes before and after reduction

4.2 Neural Network Classifier Phase

In order to start NN classifier phase we split our opponent dataset to two parts 50 opponent data for training and 30 for network classification performance testing. A MATLAB environment with the Neural Networks Toolbox was used to perform classifier training and testing. Figures (5),(6),(7) and (8) display the MSE test results for FFNN classifiers with different number of hidden neurons at 3000 and 4000 epochs. Each classifier training and testing were performed before and after input reduction.

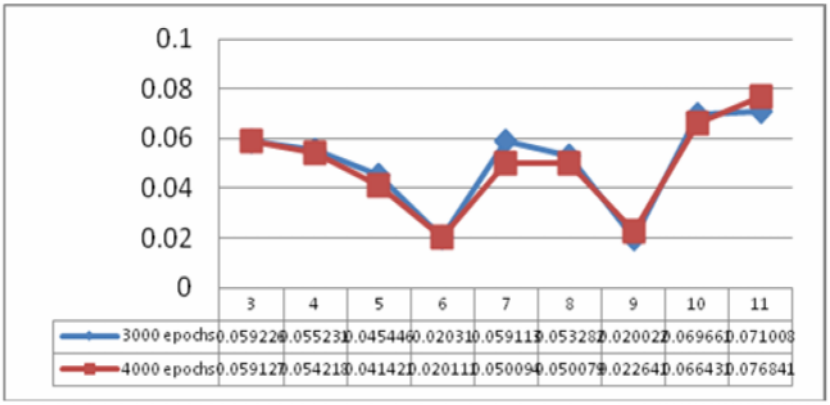

Figure 5: displays after reduction MSE for the logsig output activation function for LR=0.5

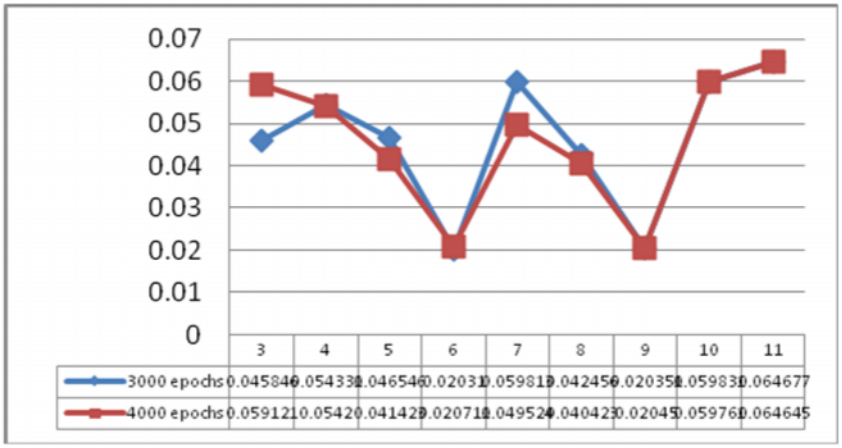

Figure 6: displays before reduction MSE for the logsig output activation function for LR=0.5

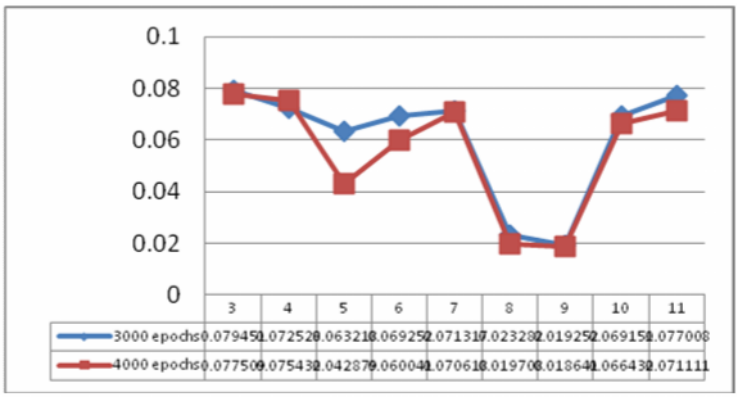

Figure 7: displays before reduction MSE for the logsig output activation function for LR=1

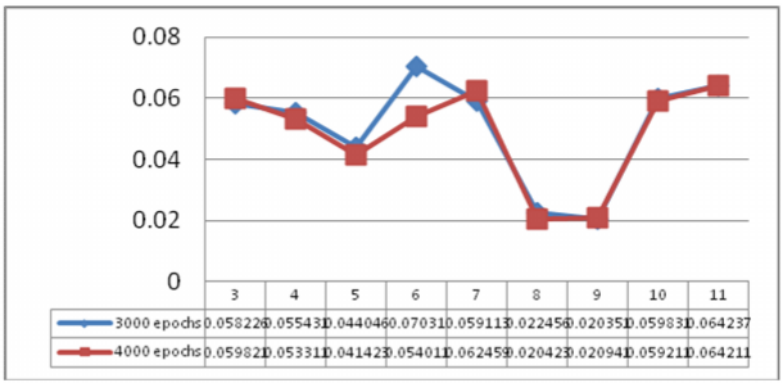

Figure 8: displays before reduction MSE for the logsig output activation function for LR=1

From the results we can observe that the rough sets based reduction of the attribute space improves the efficiency of the classifier. Tables (4) and (5) show that the classification error is at least comparable, if not smaller, when the set of inputs reduced. Also, since it is possible to obtain it with a smaller neural network (5 Input- 9 Hidden- 2 Output) trained at 4000 epochs with learning rate (LR=1) 1, the convergence time and storage will significantly reduced.

From all the previous we can say that our model can model opponent through predictiog if the opponenet is offensive or defensive to allow the AI player to face the opponent and beat him. Our model make use of the rough set to solve ANN storage and time problem to make it effective for the game real-time environment.

5. CONCLUSIONS

In this paper we proposed a hybrid intelligent opponent modeling system based rough sets and artificial neural networks. Rough sets provide a method for easily having an effective reduct of given data and its belonging rules using Johnson's reducer algorithm programmed in Aleksander Øhrn application ROSETTA. Hence; we become able to minimize attributes count which are used in our game to 62% reduction in complexity. This reduction solves the neural network problem with the increasing size of training data. As the five inputs neural network classifier

gives the same conclusion as the 13 inputs do besides saving memory and time. Experimental results from testing the proposed model in the game environment were particularly encouraging, showing that this method is capable of handling uncertainty better than other soft computing techniques.

6. REFERENCES

[1] A. J. J. Valkenberg, "Opponent Modelling in World of Warcraft". Bachelor's thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands,(2007).

[2] A. Hassanien, A. Abraham, J. Peters and J. Kacprzyk, (2001). "Rough Sets in Medical Imaging: Foundations and Trends". In Proceedings of CRC Press LLC. Pp. 9-58.

[3] C. J. Bohil, F. A. Biocca, "Cognitive Modeling of Video Game Players". Tech. Rep., Media, Interface, and Network Design Labs (MINDLab), Dept. of Telecommunication, Information and Media, Michigan State University, East Lansing, Michigan, USA, (2007).

[4] C. J. Darken, B. G. Anderegg, "Particle Filters and Simulacra for More Realistic Opponent Tracking". In Proceedings of, Charles River Media, Inc., Hingham, MA., USA, pp. 419-428, (2008).

[5] D. Carmel, S. Markovitch, "Learning Models of Opponent's Strategy in Game Playing". In Proceedings of AAAI Fall Symposium on Games Planning and Learning, Raleigh, NC, 140-147, (1993).

[6] D. Charles, M. Black, "Dynamic Player Modeling: A Framework for Player-centered Digital Games". In Proceedings of Computer Games Artificial Intelligence, Design and Education (CGAIDE 2004), University of Wolverhampton, Wolverhampton, UK, pp.29-35,(2004).

[7] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kucklich, A. Kerr, Player-centered Design: "Player Modeling and Adaptive Digital Games". In Proceedings of the DiGRA Conference.Changing ViewsWorld in Play, pp. 285-298,(2005).

[8] F. Safadi, R. Fonteneau and D. Ernst ," Artificial Intelligence Design for Real-time Strategy Games". In Proceedings of NIPS Workshop on Decision Making with Multiple Imperfect Decision Makers, Sierra Nevada, Spain, December 16th, (2011).

[9] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," Applied Artificial Intelligence, vol. 21, pp. 933–971, (2007).

[10] H. H. L. M. Donkers, "Searching with Opponent Models", Ph.D. thesis, SIKS Dissertation Series No. 2003-13, IKAT, Maastricht University, Maastricht, The Netherlands, (2003).

[11] H. H. L. M. Donkers, J. W. H. M. Uiterwijk, H. J. Van den Herik, "Probabilistic Opponent-Model Search". In Proceedings of Information Sciences pp. 123-149, (2001).

[12] H. Iida, J. W. H. M. Uiterwijk, H. J. Van den Herik, I. S. Herschberg, "Potential Applications Of Opponent-Model Search. Part 1: The Domain Of Applicability", In Proceedings of ICCA Journal 16 (4) pp.201-208, (1993).

[13] J.A. Starzyk , N. Dale and K. Sturtz ,"A Mathematical Foundation For Improved Reduct Generation In Information Systems" . In proceedings of Knowledge and Information Systems Journal, Springer, pp.131-147, (2000).

[14] J. Bazan, H.S. Nguyen, S.H. Nguyen, P. Synak, J.Wr´oblewski, "Rough Set Algorithms in Classification Problem". In proceedings of Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.), Rough Set Methods and Applications. Physica Verlag, Berlin, pp. 49–88,(2000).

[15] J. Beck, M. Stern and B.P. Woolf, "Using the Student Model to Control Problem Difficulty". In Proceedings of the Sixth International Conference on User Modeling, pp. 277–289.(1997).

[16] J. H. Kim, D. V. Gunn, E. Schuh, B. C. Phillips, R. J. Pagulayan, and D. Wixon, "Tracking Real-Time User Experience (True): A Comprehensive Instrumentation Solution For Complex Systems". In Proceedings of CHI, Florence, pp. 443–451, Italy, (2008)

[17] J. W. H. M. Uiterwijk, H. J. Van den Herik, "Speculative Play In Computer Chess", in: H. J. Van den Herik, I. S. Herschberg, J. W. H. M. Uiterwijk (Eds.), Advances in Computer Chess 7, Maastricht University, The Netherlands, pp.79-90, (1994).

[18] M. Rohs, “Preference-Based Player Modelling For Civilization IV”, bachelor’s thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands, (2007).
[19] M. J. M. Van der Heijden, S. C. J. Bakkes, P. H. M. Spronck, “Dynamic Formations in Real-Time Strategy Games”. In Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08), pp. 47-54,(2008).
[20] Qizhong, Zhang “An Approach to Rough Set Decomposition of Incomplete Information Systems”. In proceedings of 2nd IEEE Conference on Industrial Electronics and Applications, ICIEA2007, pp. 2455-2460,(2007).
[21] R. Houlette, “Player Modeling for Adaptive Games, in: AI Game Programming Wisdom 2”, Charles River Media, Inc., Hingham, MA., USA, pp. 557-566, (2004).
[22] R. Sassi, L. da Silva & E. Hernandez,” Neural Networks and Rough Sets: A comparative study on data classification”. Department of Electronic Systems Engineering University of Sao Paulo, Brazil, (2006).
[23] S. Hladky, V. Bulitko, “An Evaluation of Models for Predicting Opponent Locations in First-Person Shooter Video Games”. In Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08), pp. 39-46,(2008).
[24] V. K. Dimitriadis, “Reinforcement Learning in Real Time Strategy Games Case Study on the Free Software Game Glest”. Department of Electronic and Computer Engineering Technical University of Crete.China,(2009).
[25] Z. Pawlak,“Rough Sets”. In Proceedings of International J. Comp. Inform. Science, vol. 11, pp. 341-356,(1982).
[26] Z. Pawlak . “ Rough Sets- Theoretical Aspect Of Reasoning About Data” . In Proceedings of Kluwer Academic Publishers,(1991).
[27] Z. Pawlak, J. Grzymala-Busse, R. Slowinski and W. Ziarko, ”Rough Sets”. In Proceedings of Communications of the ACM, vol.38, no.11, pp.88-95, (1995).
[28] http://glest.org/en/index.php
[29] WWW.Wikipedia.org