MP-Draughts: a Multiagent Reinforcement Learning System based on MLP and Kohonen-SOM Neural Networks

Valquiria Aparecida Rosa Duarte*, Rita Maria Silva Julia†, Ayres Roberto Araujo Barcelos*, and Alana Bueno Otsuka‡
*Graduate student at the Computer Science Department
Federal University of Uberlandia - UFU, Uberlandia, Brazil
Email: valquiriaduarte@gmail.com, abarcelos@gmail.com
†Professor at the Computer Science Department
Federal University of Uberlandia - UFU, Uberlandia, Brazil
Email: rita@ufu.br
‡Undergraduate student at the Computer Science Department
Federal University of Uberlandia - UFU, Uberlandia, Brazil
Email: alana.otuska@gmail.com

***Abstract*—This paper presents MP-Draughts (MultiPhase-Draughts): a multiagent environment for Draughts, where one agent - named IIGA- is built and trained such as to be specialized for the initial and the intermediate phases of the games and the remaining ones for the final phases of them. Each agent of MP-Draughts is a neural network which learns almost without human supervision (distinctly from the world champion agent Chinook). MP-Draughts issues from a continuous activity of research whose previous product was the efficient agent VisionDraughts.Despite its good general performance, VisionDraughts frequently does not succeed in final phases of a game, even being in advantageous situation compared to its opponent (for instance, getting into endgame loops). In order to try to reduce this misbehavior of the agent during endgames, MP-Draughts counts on 25 agents specialized for endgame phases, each one trained such as to be able to deal with a determined cluster of endgame board-states. These 25 clusters are mined by a Kohonen Network from a Data Base containing a large quantity of endgame board-states. After trained, MP-Draughts operates in the following way: first, VisionDraughts is used as IIGA; next, the endgame agent that represents the cluster which better fits the current endgame board-state will replace it up to the end of the game. This paper shows that such a strategy significantly improves the general performance of the player agents.**

***Index Terms*—Reinforcement Learning, Draughts, Machine Learning, Neural Networks, Temporal Difference, Clustering,Kohonen Self-Organizing Maps, Games.**

I. Introduction

The choice of Draughts as an application domain is due to the fact that it presents significant similarities with several practical problems, such as interaction problems issue of the dialogue human/machine [1] and Urban vehicle traffic control [2]. Furthermore, it presents a level of complexity that demands the use of powerful techniques of states space representation, search and learning. Chinook is the world man / machine champion for Draughts [3], [4], [5] [6]. Chinook counts on two databases: one for initial phases of games and the other one for final phases. It deals with five evaluation functions manually adjusted. The human interference has been strongly significative for the high level of efficiency of Chinook.

This paper presents a multiagent environment for Draughts - MP-Draughts (MultiPhase Draughts)- whose goal is to generate a competitive automatic agent of Draughts practically without human interference, that is, just counting on powerful techniques of Artificial Intelligence (distinctly from Chinook). MP-Draughts is a multiagent system, composed of 26 agents, where an agent (IIGA - Initial/Intermidiate Game Agent) is built and trained such as to be specialized for initial and intermediate phases of a game (here characterized by board-states with at least 13 pieces) and the remaining ones for final phases of a game (here characterized by board-states with at most 12 pieces). MP-Draughts issues from a continuous activity of research whose previous product was the efficient player agent VisionDraughts [7]. Each agent of MP-Draughts presents the same general architecture of VisionDraughts, that is, a MLP (Multi-Layer Perceptron) neural network which uses temporal differences (TD(λ)) and alpha-beta pruning algorithm, combined with transposition table and iterative deepening, to learn almost without human supervision. Despite its good performance, VisionDraughts, like other good agents for Draughts that learns by reinforcement - such as Anaconda [8], NeuroDraughts [9], [10] and LS-Draughts [11]- frequently do not succeed in final phases of a game, even being in advantageous situation compared to its opponent (for example, getting into endgame Loop). In order to attack this misbehavior of the agent, MP-Draughts counts on 25 agents specialized for endgame phases, each one trained such as to be able to deal with a determined cluster of endgame board-states. These 25 clusters are mined from a Data Base (DB) containing a thousand of 12-pieces board states generated during real tournaments played by VisionDraughts. A Kohonen Network

[12], [13] is used to mine these board-states. In this way, the dynamics that the system MP-Draughts adopts for playing is the following one: VisionDraughts is the agent that plays from the complete initial board-state of a game up to a 13-board-state (that is, it corresponds to the IIGA). From this point on, the endgame agent that represents the cluster which better fits the current endgame board state - also detected by the Kohonen Network- will replace VisionDraughts (this agent is named EGA - EndGameAgent) . This paper shows by means of tournaments that MP-Draughts significantly improves the general performance of VisionDraughts.

II. Background in Computer Programs for Draughts

The first great experiment in automatic learning for draughts is devoted to Samuel in 1959 [14]. He used a deterministic learning procedure where boards were evaluated based on a weighted polynomial comprising features considered important for generating good moves. The use of features (NET-FEATUREMAP mapping) aims to provide qualitative measures to better represent the properties of pieces on a board. More details about these features can be found in [14], [15], [9]. That work remains a milestone in artificial intelligence research, since it was the pioneering automatic draughts player that succeeded competing against good human players.

Chinook is the most famous and strongest draughts player in the world [3], [4] and [6]. It is the result of research in strategy for games started in 1989 by Schaeffer et al [3]. The project had two objectives that have been achieved: to develop a program capable of winning the world human champion and solving the draughts games. Chinook is the world man-machine draughts champion and uses linear handcrafted evaluation functions (whose role is to estimate how much a board state is favorable to it) that considers several features of the game, for example: quantity of pieces and kings, trapped kings, turn and runaway checkers. In addition, it has access to a library of opening moves obtained from games played by grand masters and to an endgame database, which is a computer-generated collection of positions with a proven game-theoretic value (actually, a collection has 39 trillion possible board states with 10 pieces). To choose the best action to be executed, Chinook uses a parallel iterative alpha-beta search with transposition tables and the history heuristic [4]. At the U.S. Open [3], the program averaged 20-ply searches (a ply is one move executed by one of the players).

Fogel [8], with his player Anaconda, explored a co-evolutionary process to implement an automatic draughts player which learns without human expertise. He focused on the use of a population of neural networks, each one counting on an evaluation function to estimate the quality of the current board state. According to Fogel, the effort spent on Automatic learning should minimize the human intervention in the learning process of the player. In a tournament of 10 games against a novice player version of Chinook, its best neural network obtained 2 wins, 4 losses and 4 draws, which allowed it to be ranked as expert. It is interesting to point out that Fogel did not try to attack the problem of endgame loops, since in Anaconda they are considered as being a case of draw.

Mark Lynch's Draughts player, the Neuro-Draughts, consists of a MLP neural network whose weights are updated by TD(λ) Reinforcement Learning methods. The NET-FEATUREMAP is used to represent the board-states in the network input. The network output corresponds to a real number (prediction) that indicates to what extent the input state is favorable to the agent. The agent is trained by self-play coupled with a cloning technique. The Minimax algorithm is used to choose the best action to be executed considering the current game board-state [10] [9]. In NeuroDraughts, the features are manually selected and do not vary (the player is trained only for a fixed set of characteristics). Mark Lynch [9] developed his Draughts player trying to use the minimum of human intervention as possible. EndGame Loops are very frequent in this player.

In order to improve the NeuroDraughts abilities, Neto and Julia proposed LS-Draughts [11]: an automatic draughts player which extends the architecture of NeuroDraughts with a genetic algorithm that automatically generates a concise set of efficient and sufficient features for representing the game board states and for optimizing the training of the neural network by TD(λ). The improvement obtained was confirmed by a tournament of seven games where LS-Draughts, with a more concise set of features, scored two wins and five draws playing against NeuroDraughts.

Still in order to improve NeuroDraughts, Caixeta and Julia proposed VisionDraughts [7]. VisionDraughts is an automatic draughts player which replaces a very efficient tree-search routine which uses alpha-beta pruning, transposition table and iterative deepening for the minimax search module of NeuroDraughts. The experimental results confirm the improvements obtained: the runtime required for the search algorithm of VisionDraughts is 95% less then that required by NeuroDraughts, which allowed to increase the depth search end to obtain a much better player in a reasonable training time. Even though in VisionDraughts there is a significant reduction of endgame loops compared to NeuroDraughts, they still represent a problem which compromises the efficiency of VisionDraughts during endgames.

III. Theoretical Foundations

In the same way as NeuroDraughts [9], LS-Draughts [11] and VisionDraughts [7], the agent of MP-Draughts consist of MLP networks that learns by TD(λ) in self-play training with cloning, where the board-states are represented in the input of the networks by NETFEATUREMAP.

Similarly to VisionDraughts, MP-Draughts also performes the search for the best move corresponding to a current board-state by means of an alpha-beta pruning algorithm combined with Table Transposition and Iterative Deepening [16].

It is interesting to note that NeuroDraughts, Anaconda, LS-Draughts and VisionDraughts are competitive multiagent systems, since whenever a player B tries to maximize its performance measure, it minimizes the performance measure of its opponent [17]. Furthermore, despite the fact that all of them counts on several neural networks during the training, the final trained player consists on a single neural network: that one which better learned during training tournaments. Similarly to these automatic players, MP-Draughts is also a competitive multiagent system. Nevertheless, as said before, its final player system differs from theirs, since it counts on 2 cooperatives MLP neural networks: the best network of VisionDraughts (IIGA) and the EGA (chosen among the 25 endgame agents, by means of a Kohonen Network), responsible for the initial/intermediate and for the final phases of a game, respectively. The next subsections resume the principal techniques and tools used in the implementation of MP-Draughts. Considering that the main contribution of this work is focused on the clustering process and on the selection of the most appropriate EGA, both proceedings performed by a Kohonen Network, this kind of neural network will be presented in section IV, which details the architecture of MP-Draughts.

A. *MLP Artificial Neural Networks*

A Neural Network is a computational model based on biological neural networks which consists of a network of basic units called neurons (nodes). These nodes are the artificial neurons, originally intended to simulate the operation of a single brain cell. The same algorithm runs on each neuron, witch communicates with a subset of other neurons of the same network [18]. More details can be seen in [11], [17], [8], [12], [19], [13].

As said before, the agents of MP-Draughts correspond to MPL networks. A multilayer perceptron is a feedforward artificial neural network model that maps sets of input data onto an appropriate output set. It has at least one hidden layer of neurons. It is a modification of the standard linear perceptron in that it uses three or more layers of neurons (nodes) with nonlinear activation functions,what makes it more powerful than the perceptron, since it can distinguish data that are not linearly separable. More details about MLP can be seen in [11], [17], [8].

B. *Reinforcement Learning*

The Reinforcement Learning paradigm has been of great interest for the learning machine due to the fact that it does not require a 'coach' during the learning process. This fact is particularly appropriate in complex areas where the collection of examples for learning is difficult or impossible [17].

Among the Reinforcement Learning methods, one can spot the learning methods of the temporal differences, TD(λ): they are widely used with highly efficient results, including in the construction of agents capable of learning to play Draughts, Chess, Backgammon and other games, [9], [14], [4], [20], [21].

TD learning [22], [23] has emerged as a powerful reinforcement learning technique for incrementally tuning parameters. The basic idea is: a learning agent receives an input state that is continuously modified by means of the actions performed by the agent. Each current state is evaluated based on the previous one. At the end of the process, it outputs a signal and then receives a scalar *reward* from the environment indicating how good or bad the output is (reinforcement). That is, the learner is rewarded for performing well (positive reinforcement) and, otherwise, it is punished (negative reinforcement).

Tesauro [22] introduced the use of this techniques in games proposing the TD-Gammon, a very efficient automatic gammon player corresponding to a neural network that learned by TD(λ). In chess, KnightCap [23] learned to play chess against humans on the internet also using TD. In the draughts domain, in spite of the undeniable efficacy that Chinook obtained by manual adjustment of its evaluation function during several years, Schaeffer [5] showed that TD learning is capable of competing with the best human effort in the task of adjusting the evaluation function. In fact, Schaeffer's automatic player based on a neural network trained by TD proved to be a very good draughts player [5].

C. *Alpha-Beta Pruning and Transposition Table*

The Draughts game can be thought as a tree of possible future states of the board. Each node in the tree represents a board state and each branch represents one possible move [18].

In this sense, the alpha-beta algorithm selects the best action to be executed considering the current board-states. This algorithm can be summarized as following: it is a recursive algorithm for choosing the best move performing a simple left-to-right depth-first traversal on a search tree [24]. The mechanism of alpha-beta pruning removes sessions of the search tree witch surely do not comprise the best move.

In order to provide more efficiency to the search mechanism, the alpha-beta algorithm can be used jointly with transposition table. The transposition table stores board-states already evaluated in such a way as to indicate whether a game tree must be expanded or not (usually, the board states available in the transposition table do not need to be evaluated). Therefore, considering that the alpha-beta algorithm does not keep a history of solutions previously obtained and that the iterative deepening can be extremely repetitive during the search process [17], the use of a transposition table associated with them improves a lot the search method [16].

IV. MP-Draughts Player Architecture

MP-Draughts is a multiagent system for Draughts. Figure 1 illustrates the general architecture of MP-Draughts, where: an agent - IIGA - is built and trained such as to be specialized for initial and intermediate phases of a game and another one - EGA - for final phases of a game. The EGA is selected, by Kohonen networks, from a set of 25 trained endgame agents, each one trained such as to be able to deal with a certain cluster of endgame board- states. These 25 clusters are mined from a DB containing a thousand of 12-pieces board states generated during real tournaments played by VisionDraughts. A Kohonen Network is used to mine these board-states.

Each agent of MP-Draughts presents the same general architecture of VisionDraughts, that is, a MLP neural network which uses temporal differences and alpha-beta algorithm

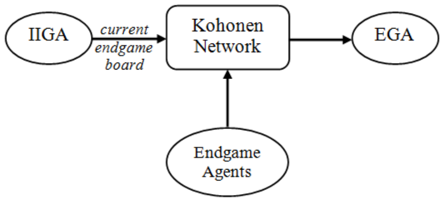


Fig. 1. MP-Draughts Architecture

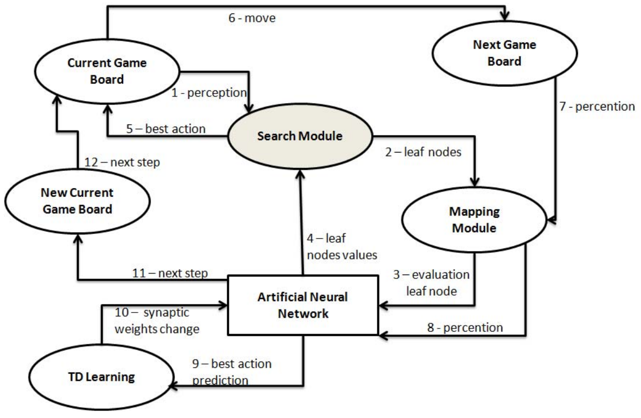


Fig. 2. Learning Process of VisionDraughts

combined with transposition table and iterative deepening to learn almost without human supervision.

The subsections below resume the architecture of the agents of MP-Draughts.

A. VisionDraughts - IIGA

Figure 2 resumes the learning process of VisionDraughts [7], that is: the tree-Search Routine selects, using an efficient search routine (alpha-beta pruning and transposition table), the best move corresponding to the current board state. The TD Learning module is responsible for training the player neural network. The Artificial Neural Network module is the core of agent. It corresponds to a three layer feedforward network whose output layer is composed of a single neuron. The role of this network is to evaluate to what extent the board state,represented by NET-FEATUREMAP in the input layer, is favorable to the agent (it is quantified by a real number comprised between 0 and 1, called prediction, which is available in the output of the third layer neuron).

B. EndGameAgent - EGA

MP-Draughts counts on 25 agents specialized for endgame phases, each one trained such as to be able to deal with a determined cluster of endgame board-states. These 25 clusters are mined from a DB containing a thousand of 12-pieces board states generated during real tournaments evolving VisionDraughts. As MP-Draughts uses VisionDraughts as IIGA, it is coherent that it also uses the same agent to generate the endgame board-states to be stored in the DB.

The next subsection details the main proceeding concerning the generation of the endgame agents of MP-Draughts.

1) Representations of the Boards in MP-Draughts: MP-Draughts adopts the following two representations for the board-states:

- A vector based representation, where a vector stores the 32 positions corresponding to the complete board. This precise representation is very appropriate to represent the endgame board-states to be stored into the DB, since it allows to distinguish clearly a certain board from another one, which will be very useful during the process of converting this first representation into the second one- more appropriate for the learning process- called feature-based representation, described below.
- A feature-based representation: several experiments trying to find a more convenient way to represent board-states in order to optimize the learning of Draughts player agents have shown that a feature-based representation, even being approximate, produces much better results then full vector-based representation [11] [9]. That is why this representation is very appropriate during the learning process of a Kohonen-SOM responsible for clustering a set of *B* board-states. In this case, a collection of *n* features provides a qualitative representation for them. Each feature f_i indicates whether a certain board satisfies or not a determined constraint required by f_i, that is, it can be defined as a mapping:
  f_i : B $\longrightarrow$ {0,1}, where:
  f_i(x) = 1, if a board-state *x* satisfies the constraint expressed by f_i;
  f_i(x) = 0, otherwise.
  The value returned by $f_i(x)$ is called the *attribute* of the board-state *x* with respect to the i-th feature f_i. For example, considering f_i as being the feature "PieceAdvantage" conceived to check whether a given board *b* presents or not pieces in advantage with respect to one of the players, an attribute f_1*(b) = 1* would indicate that the considered player has pieces in advantage in *b*, whereas the attribute f_1*(b) = 0* would indicate the opposite.

As explained above, in order to deal with the clustering process of the Kohonen-SOM, MP-Draughts converts the vector-based representation of the endgame boards of the KB into a feature-based representation. It performs it by applying a conversion mapping C to each board-state to be clustered. The mapping C is defined such as shown below:

C: B $\longrightarrow \{0,1\}^n$, where:

C(x) = $<$ f_1(x),..., f_n(x) $>$

As mentioned before, *n* is the quantity of features to be used in the feature-based representation and, particularly here, it is equal to 28 such as proposed by Samuel [14] [15]. It means that each board-state *b* will be represented, during the clustering process, by a n-tuple composed of the *n* attributes of *b* with respect to the features f_1, , f_n . The next section shows how this board representation is used in the clustering process.

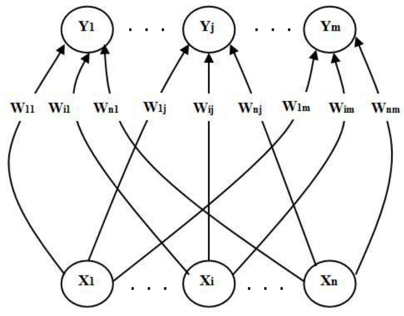


Fig. 3. Model of Kohonen-SOM

2) Clustering by Means of Kohonen-SOM: The architecture of a Kohonen-SOM, showed in figure 3,includes a n-neuron input layer and a m-neuron output layer, where all neurons X_i of the former ($1 \leq$ i $\leq$ n) is connected to all neurons Y_j of the latter one ($1 \leq$ j $\leq$ m). In this case, w_{ij} indicates the weight of the link between the *i-th* neuron of the input layer and the *j-th* neuron of the output one. Further, the *weight vector* for each neuron Y_j of the output layer is the vector composed of all *n* weights w_{ij} . Learning occurs the extent to which each datum to be clustered is submitted to the input neurons. In order to update the weights of a Kohonen-SOM such as to make it able to cluster a collection of *p* data, for each datum B_k, ($1 \leq$ k $\leq$ p) the following proceedings is repeatedly executed- each time with a decreased learning rate- until a determined stopping condition is true : the datum is submitted to the Kohonen-SOM, that is, each of its attributes A_i ($1 \leq$ i $\leq$ n) is introduced into one of the input neurons X_i; next, for each output neuron Y_j, the algorithm calculates the Distances D_j between the datum B_k and Y_j (as shown in line 4 of the pseudo-code below); the *weight vectors* corresponding to the output neuron with the minimum D_j and to the output neurons within a specified neighborhood of Y_j are updated. As soon as the training is finished for all *p* data, each datum is resubmitted to the Kohonen-SOM: it will belong to the cluster defined by the output neuron Y_j which presents the minimum distance *D* compared to the attributes of the datum. This proceedings is illustrated in the pseudo-code below:

Kohonen Algorithm

```
1:  while alpha $> 0.1$ do
2:     for each input datum $B_k$: do
3:        for each $j$, compute: do
4:           D(j) = $\sum_i (W_{ij} - A_i)^2$
5:        end for
6:        Find index $J$ such that $D(J)$ is a minimum
7:        for all units $j$ within a specified neighborhood of $J$,
          and for all $i$: do
8:           $W_{ij}$(new) = $W_{ij}$(old) + $\alpha[X_i$ - $W_{ij}$(old)]
9:        end for
10:       Update learning rate
11:       Test stopping condition
12:    end for
13: end while
```

In this way, in order to cluster the endgame board-states stored in the DB (here, p = 1000), MP-Draughts simply executes the proceedings described above for all of them, that is: first, it converts them into 28-features-based representations (by means of the conversion mapping C); next, for each one, it trains a Kohonen-SOM composed of 28 input neurons (where each one will receive one of the 28 attributes of the current endgame board-state) and of 25 output neurons (once MP-Draughts must produce 25 clusters). As soon as the training is finished for all data (endgame board-states), each one is resubmitted to the Kohonen-SOM: it will belong to the cluster defined by the output neuron Y_j which presents the minimum distance *D* compared to the attributes of the datum. As shown in the following sections, such a strategy works very well in the task of clustering the endgame board-states of Draughts, since the successful cooperation between the IIAG and the EAG chosen by the Kohonem-SOM to finalize a game, allows MP-Draughts to beat the single MLP of VisionDraughts in several tournaments.

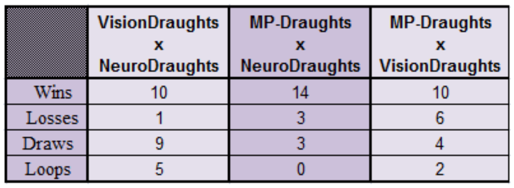
3) Generation of the EGA: As soon as the clustering process is concluded, 25 endgames agents are trained by self-play coupled with a cloning technique in order to represent the 25 clusters. Each trained endgame agent will represent a determined cluster. In order to make it possible, the training of each agent is performed by means of tournaments whose initial boards are the ones belonging to the cluster which that agent is designate to represent. Once the training of all endgame agents is over, MP-Draugths is able to play according to the following dynamics: the IIGA starts the game, playing from the complete initial board-state up to a 13-pieces-endgame board-state; from this point on, MP-Draughts submits the current endgame board-state to the trained Kohonem-Som, whose role at that moment is to detect which cluster better fits it (the output neuron Y_j which presents the minimum *D* distance compared to the attributes of the current endgame board-state will indicate this cluster). The endgame agent which represents this selected cluster is the EGA to be used to play up to the end of the game.

V. Experimental Results

This section presents the performance of the multiagent architecture of MP-Draughts compared to the architecture based on a single MLP-Network of VisionDraughts and Neuro-Draughts. The parameter taken into account for the comparison is the number of wins, draws and losses obtained by each one in a tournament of 20 games. It is interesting to point out that all players were trained in a similar way, that is, by self-play with cloning, during 10 tournaments of 250 games each one. Table I shows the results of the competition between them.

The results confirm the significant contribution of the proposed multiagent game architecture in the general performance of a good player agent.

In fact, as shown in table I, the multiagent architecture oh MP-Draughts significantly improves the general performance



	VisionDraughts x NeuroDraughts	MP-Draughts x NeuroDraughts	MP-Draughts x VisionDraughts
Wins	10	14	10
Losses	1	3	6
Draws	9	3	4
Loops	5	0	2

TABLE I
RESULTS OF THE TOURNAMENT BETWEEN MP-DRAUGHTS, VISIONDRAUGHTS AND NEURODRAUGHTS

of the single agent based architecture of NeuroDraughts and VisionDraughts.

Furthermore, MP-Draughts succeeded in several endgame situations in which VisionDraughts and NeuroDraughts had already proved to be inefficient, falling into infinite endgame loops even being in advantageous condition. In fact, the percentage of occurrence of endgame loops in the tournaments VisionDraughts x NeuroDraughts, MP-Draughts x NeuroDraughts and MP-Draughts x VisionDraughts were 25%, 0% and 10%, respectively. As most of the draws shown in table I was due to endgame loops, these results confirm the improvement obtained in the multiagent architecture of MP-Draughts in endgame phases.

VI. CONCLUSION AND FUTURE WORKS

This paper presented how much a multiagent player system whose agents are specialized for distinct phases of games can improve the general performance of automatic non supervised players of Draughts. The architecture of each agent of MP-Draughts consists of a neural network which uses temporal differences and alpha-beta algorithm combined with transposition table and iterative deepening to learn almost without human supervision. MP-Draughts counts on one efficient agent for starting games (VisionDraughts) and on 25 agents specialized for endgame phases, where each one of the endgame agents are trained such as to be able to deal with a determined cluster of endgame board-states. These clusters are obtained by means of a Kohonen-SOM from a DB containing endgame board-states. A competition between VisionDraughts and MP-Draughts, won by the latter one, confirms that the multiagent proposal of MP-Draughts in fact improved the architecture based on a single agent of VisionDraughts.

As future works, the authors intend to introduce in MP-Draughts a module corresponding to a genetic algorithm whose role is to select automatically a concise set of features more appropriate to represent the board-states during the clustering process (note that, in the present implementation, these features have been chosen manually). The authors believe that the automation of the feature selection process can improve the efficiency of the multiagent system MP-Draughts. Finally, the authors intend to develop appropriate interfaces that will alow MP-Draughts to play against Chinook, in order to estimate the performance of the former compared to the world champion.

ACKNOWLEDGMENT

To FAPEMIG (Fundacao de Amparo a Pesquisa do Estado de Minas Gerais) and CAPES (Coordenacao de Aperfeioamento de Pessoal de Nivel Superior) for financial support (providing equipments and scholarship, respectively)

REFERENCES

[1] M. A. Walker, “An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email,” *Journal of Artificial Intelligence Research 12*, pp. 387–416, 2000.

[2] M. Wiering, “Multi-agent reinforcement learning for traffic light control,” *Proceedings of the 17th International Conference on Machine Learning*, pp. 1151–1158, 2000.

[3] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, “A world championship caliber checkers program,” *Artificial Intelligence*, vol. 53, pp. 53–2, 1992.

[4] J. Schaeffer, R. Lake, M. Bryant, and P. Lu, “Chinook: The world man-machine checkers champion,” *AI Magazine*, 1996.

[5] J. Schaeffer, M. Hlynka, and V. Jussila, “Temporal difference learning applied to a high performance game-playing program,” *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 529–534, 2001.

[6] J. Schaeffer, N. Burch, Y. Bjrnsson, A. Kishimoto, M. Mller, R. Lake, P. Lu, and S. Sutphen, “Checkers is solved,” *Science Express*, vol. 317, pp. 1518 – 1522, 2007.

[7] G. S. Caixeta, “Visiondraughts -um sistema de aprendizagem de jogos de damas baseado em redes neurais, diferenas temporais, algoritmos eficientes de busca em arvores e informacoes perfeitas contidas em bases de dados,” Master’s thesis, Federal University of Uberlndia, 2008.

[8] K. Chellapilla and D. Fogel, “Anaconda defeats hoyle 6-0: A case study competing an evolved checkers program against commercially available software,” *Proceedings of the 2000 Congress on Evolutionary Computation*, pp. 857–863, 2000.

[9] M. Lynch, “Neurodraughts: An application of temporal difference learning to draughts,” Master’s thesis, Department of Computer Science and Information Systems, University of Limerick, Ireland, 1997.

[10] M. Lynch and N. Griffith, “Neurodraughts: the role of representation, search, training regime and architecture in a td draughts player,” *Eighth Ireland Conference on Artificial Intelligence*, 1997.

[11] H. C. Neto and R. M. S. Julia, “Ls-draughts - a draughts learning system based on genetic algorithms, neural network and temporal diferences,” *IEEE Congress on Evolutionary Computation*, pp. 2523–2529, 2007.

[12] T. Kohonen, “Self-organizing formation of topologically correct feature maps,” *Biological Cybernetics*, pp. 59–69, 1982.

[13] ——, “The self-organizing map,” *Proc. IEEE*, pp. 1464–1480, 1990.

[14] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal on Research and Development*, pp. 210–229, 1959.

[15] ——, “Some studies in machine learning using the game of checkers ii,” *IBM Journal on Research and Development*, 1967.

[16] G. S. Caixeta and R. M. S. Julia, “A draughts learning system based on neural networks and temporal differences: The impact of an efficient tree-search algorithm,” *The 19th Brazilian Symposium on Artificial Intelligence, SBIA*, 2008.

[17] S. Russel and P. Norving, *Inteligncia Artificial - Uma abordagem Moderna*, 2nd ed. Editora Campus, 2004.

[18] I. Millington, *Artificial Intelligence for Game*. Morgan Kaufmann, 2006.

[19] T. Kohonen, “Analysis of a simple self-organizing process,” *Biological Cybernetics*, pp. 135–140, 1982.

[20] G. J. Tesauro, “Td-gammon, a selfteaching backgammon program, achieves master-level play,” *Neural Computation*, vol. 6, no. 2, pp. 215–219, 1994.

[21] S. Thrun, “Learning to play the game of chess,” *Advances in Neural Information Processing Systems*, pp. 1069–1076, 1997.

[22] G. J. Tesauro, “Temporal difference learning and td-gammon,” *Communications of the ACM*, vol. 38, no. 3, pp. 19–23, 1995.

[23] J. Baxter, A. Tridgell, and L. Weaver, “Knightcap: A chess program that learns by combining td(λ) with game-tree search,” *Machine Learning, Proceedings of the Fifteenth International Conference*, pp. 28–36, 1998.

[24] J. Schaeffer and A. Plaat, “New advances in alpha-beta searching,” *Conference on Computer Science*, pp. 124–130, 1996.