ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

D-VisionDraughts: a Draughts Player Neural Network That Learns by Reinforcement in a High Performance Environment

Ayres Roberto Araújo Barcelos[1], Rita Maria Silva Julia[1] and Rivalino Matias Júnior[1]

1- Federal University of Uberlandia - Computer Science Department
Av Joao Naves de Avila 2121, Uberlandia MG - Brazil

Abstract. This paper describes D-VisionDraughts, a distributed player agent for draughts which is based on Neural Networks trained by Temporal Differences. D-VisionDraughs is trained in a high performance environment and achieves a high level of play without expert game analysis and with minimum human intervention. D-VisionDraughts corresponds to a distributed version of the efficient agent player VisionDraughts. In this way, the main contributions of this paper consist on substituting the distributed Young Brothers Wait Concept algorithm (YBWC) for the serial alpha-beta search algorithm used in VisionDraughts and on measuring the impact of a high performance environment into the non-supervised learning abilities of the player. Evaluative tests proved that even a modest distributed version counting just on ten processors is able to reduce from about 83% the search runtime and to increase from 15% its capacity of winning.

1 Introduction

This paper presents the D-VisionDraughts, a distributed and improved version of the successful agent for draughts - based on a multilayer perceptron Neural Networks (MLP) that learns by reinforcement- named VisionDraughts [2]. The Reinforcement Learning methods have been a subject of great interest in the machine learning area, since it does not require an intelligent instructor to provide training examples. Therefore, it is a suitable tool for dealing with complex domains where it is hard or even impossible to obtain such examples [8]. Among the reinforcement learning methods, Temporal Differences (TD) stands out. It has been widely used with highly efficient results, including in the construction of agents capable of learning to play draughts, chess, backgammon, go and other similar games [5], [14] and [12]. Such agents have demonstrated that games are a very suitable domain to study and to check the efficiency of the machine learning techniques.

Particularly, draughts has been chosen as a test bed to evaluate appropriate learning methods because it presents significant similarities with several practical problems (e.g., human-machine dialogue [15], and urban vehicle traffic control problems [16]). Further, draughts game presents many of the research challenges of other board games like chess, but without the unnecessary complexity [11]. The Schaeffer's player Chinook [10], [11] is the current man-machine world champion in draughts. Its evaluation functions are manually adjusted and the system counts on databases to optimize the choice of the best move. It means that the learning process of Chinook, differently from the one of D-VisionDraughts, is strongly supervised.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

The previous version of D-VisionDraughts - VisionDraughts - uses an efficient serial algorithm with alpha-beta pruning to perform the search and counts on two versions: in the first one [2], the alpha-beta algorithm follows a limited-depth strategy of search, whereas in the second version it expands the current state based on an iterative-deepening process. In the second version, a transposition table [2] is used to store the nodes that have already been evaluated, speeding up the iterative deepening search (where the same nodes may repeatedly be expanded [8]), and to order the search-tree. In this paper, the authors distribute the search algorithm used in the first version of VisionDraughts by using the Young Brother Wait Concept (YBWC)[3] and the inter-process communications mechanism MPI (Message Passing Interface) [7].

2 Temporal Differences methods in games and Related Works

This section explains how TD Reinforcement Learning methods can be used by a player Neural Network. The idea is that the Network is rewarded for a good performance (receiving from the environment a positive reinforcement corresponding to the endgame state, in case of victory) and it is punished for a bad performance (receiving from the environment, in case of defeat, a negative reinforcement corresponding to the endgame state). For all the intermediate game board states (between the starting board and the final board) represented in the input layer of the Network, as no specific reward is available, the TD mechanism calculates the prediction P of victory by means of the following equation:

$$P = g(in^{output}), \tag{1}$$

where g is the hyperbolic tangent function and in^{output} is the local induced field on the neuron of the Network output layer [5], [8]. It means that the value of P depends on the Network weights. A prediction P corresponds to a real number belonging to the interval [-1,1] that indicates how much the game board state represented in the input of the Network is favorable to the agent. Each time the agent in a current state S must move a piece, a search algorithm is called in order to build a search tree whose evaluation will indicate the best move in S. Next, each leaf board-state of the tree is represented into the input of the Network such that this Network estimates its prediction value. These prediction values are used by the search algorithm in order to indicate to the agent which is the best action to be chosen and executed in S. Whenever the agent executes a move, the Network weights are updated according to equation 2 [14]:

$$\Delta w_t = \alpha(P_t - P_{t-1})\sum_{k=1}^{t-1} \lambda^{(t-1)-k} \nabla_w P_k, \tag{2}$$

where P_t is the prediction corresponding to the current game board state, P_{t-1} is the prediction corresponding to the previous game board state, each P_k represent the prediction corresponding to an earlier game board state, α is the learning rate (defined according to how fast the system will update the Network weights), λ is a constant

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

defined according to how much the system will consider the impact of an earlier state P_k in the weight updating process and $\nabla_w P_k$ correspond to the partial derivate of P_k with respect to the variable *w* (weight).

A relevant contribution in the field of Reinforcement Learning was given by Gerald Tesauro when he applied the TD methods to train an evaluation function of a board game [14]. Tesauro's program, TD-Gammon, is a backgammon player that, in spite of having very little knowledge about backgammon, is able to play as efficiently as the greatest world players. The principles of TD methods have first been applied by Samuels, who pioneered the idea of updating evaluations based on successive predictions in a checker program [9].

3 The Architecture of D-VisionDraughts

D-VisionDraughts is an agent that learns to play draughts by reinforcement on a high performance environment, what differentiates it from the other non-supervised agents cited in section 2 (note that Chinook, in spite of also being parallelized, is a supervised agent). The architecture of D-VisionDraughts uses the MPI [7] as inter-process communication mechanism. In order to estimate the impact of parallelizing an agent that learns by reinforcement, D-VisionDraughts corresponds to a distributed version of the efficient player VisionDraughts [2] where the serial alpha-beta search algorithm of the later is replaced by the distributed algorithm Young Brothers Wait Concept (YBWC). Figure 1 shows the architecture of D-VisionDraughts. The YBWC was chosen because it showed the best tradeoff between performance and programming constraints and it naturally fits the distributed scenarios found in many practical applications. Other options like APHID [1] or DTS [1] were not considered due to additional technical difficulties. DTS, for example, requires a shared memory architecture. Concisely, whenever the agent must choose a piece to move, the current board state, B1, is presented to the distributed Alpha-Beta search algorithm (YBWC), #1 (step 1 in the figure). The search module performs a limited depth-first search corresponding to B1, #2. Each leaf board-state of the search tree is converted into a feature-based representation - that is, it is represented by means of a set of functions (called features) that captures relevant knowledge about the domain of Draughts [2] – and is presented to the input of the MLP, #3. The MLP will evaluate each of these leaves and will output a value (prediction) that indicates to which extent it is favorable to the agent player. This value is returned to the search algorithm, #4, in order to allow it to point out the best move to be performed from B1, #5. The agent then executes the move, #6. The new board state B2 is converted into a feature-based representation, #7, and presented to the MLP to be evaluated, #8. If the agent plays a training game, the prediction calculated by the MLP for B2 is used by the TD learning module as an input parameter to update the MLP weights, #9. The weights of the MLP are updated, #10, and the cycle begins again for the new current board state B2, #11.The agent is trained by self-play with cloning technique [2]. Note that for a non-training match, D-VisionDraughts presents the same architecture described above, just cutting off the TD learning module and links #9, and #10.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

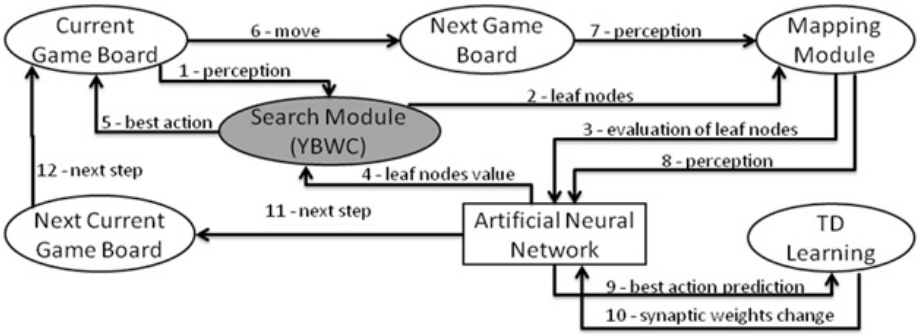

Fig. 1: D-VisionDraughts Learning Process

The Artificial Neural Network (ANN) module is the core of the agent and corresponds to a three layer feedforward network whose output layer is composed of a single neuron. As said before, the role of this network is to evaluate to which extent a board state is favorable to the agent (prediction).

The main idea of the algorithm YBWC used in D-VisionDraughts is to search the first sibling node, referred as the *eldest brother*, before spawning the remaining siblings in parallel, the *younger brothers*. This is based on the observation that the previous evaluation of the eldest brother will more likely produce either a convenient pruning on the search tree (which will avoid an unnecessary evaluation of the younger brothers), or a convenient narrowing on the search window that will speed up the search [3]. If the evaluation of the eldest brother does not produce a cut-off, then the remaining moves are searched in parallel. As in the original sequential alpha-beta algorithm the exploration of a node depends on the priori knowledge obtained during the expansion of the eldest nodes to prevent searching those parts of the tree that have no influence on the final results, the task of distributing this algorithm is not trivial, since a parallel search will traverse different parts of the tree, simultaneously, with no guaranty that such a priori knowledge is available. That is the main difficulty to be overcome when distributing the alpha-beta algorithm. The YBWC tries to solve this problem in the following way: instead of waiting to have the search window value corresponding to the evaluation of the eldest brothers for exploring a node N, a master processor P_1 distributes its exploration to an idle slave processor P_2 with the current available search window and, as soon as P_1 detects a narrowing in that window, it communicates these new bounds to P_2. This later, then, checks whether these new bounds allow a pruning of N. If it does, P_2 aborts the processing of N.

D-VisionDraughts operates with a dynamic number *n* of processors (particularly here, 10 processors were used). It uses a stack data structure to appropriately represent the state space, given that it keeps track of the nodes to be explored. The depth of the stack is the same of the nodes being currently explored in the search tree. Thus, all nodes of a certain level of each sub tree are pushed into the same depth of the stack. Each processor keeps its own local stack. Whenever this stack is empty, the processor requires a task to the other processors. When a processor P receives a task of exploring a node N, placed in the depth *d*, it stores N in its local stack in the same depth *d* and begins to explore it. Each time D-VisionDraughts starts off a search, the current board state, N_0, is pushed into the local stack of a main processor P_0 (at that

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

moment, the local stacks of the other processors are empty). The system then performs a serial expansion of the left-most sub tree of N_0 up to the maximum depth, being able to start the distributed expansion of the remaining sub-trees of N_0.

4 Experimental Results

This section presents comparative real tests whose results confirm the improvement obtained with the distributed search of D-VisionDraughts - counting on two Intel Core 2 Quad Q6600 CPU and one Intel Core 2 Duo E6600 CPU connected by a Gigabit Ethernet switch - compared to the serial search performed by VisionDraughts.

1) First test: D-VisionDraughts and VisionDraughts evaluated 20 distinct board-states, chosen randomly in a database composed of board-states generated in matches involving famous human players of draughts [4]. These evaluations were performed in a limited depth-search expansion of depth 14. Note that this depth (look-ahead) requires a high level of performance from the agent. For example, NeuroDraughts was trained with look-ahead of just 4, since deeper expansions would demand a very long training time in order to produce a good player [2] (in fact, in [2] it is shown that the runtime of VisionDraughts is 90% less than the one of NeuroDraughts). The objective of this first test is to estimate the average search runtime (in seconds) required for each agent. Table 2 shows that the search runtime of D-VisionDraughts is about 83% less than the search runtime of VisionDraughts.

2) Second test: in order to compare the learning capacity of the player Neural Networks of D-VisionDraughts and VisionDraughts, the best player of each system disputed an evaluative tournament of 40 matches. Both players were generated from a training tournament with the same duration of 4 hours and with the same search depth limit of 12. The improved distributed search algorithm of D-VisionDraughts allowed it to make 79 training matches during the training tournament, whereas VisionDraughts could make just 27 ones. The objective of this test is to check which system will generate the best player in the same training conditions. Table 2 shows that D-VisionDraughts obtained 15% more victories than VisionDraughts. The last but one line of table 3 indicates the quantity of draws and the last line indicates how many of these draws were caused by endgame loops.

VisionDraughts (in seconds)	D-VisionDraughts with 10 processors (in seconds)
78.08	12.99

Table 2: Experimental results

Results obtained by D-VisionDraughts playing against VisionDraughts	
Wins	16
Losses	10
Draws	14
Loops	6

Table 3: Evaluative Tournament : D-VisionDraughs X VisionDraughs

5 Conclusion and Future Works

This paper presents how much the search module based on the YBWC distributed alpha-beta algorithm implemented in D-VisionDraughts can improve the learning

process of Neural Networks. Evaluative tests showed that the distributed search version (counting just on 10 processors) was about three times faster than the serial version. This advantage allowed that D-VisionDraughts, in the same training conditions, was able to generate a much better player. As future works, the authors intend to distribute the search module of the second version of VisionDraughts (which includes Transposition Tables, Iterative Deepening and move ordering and to increase the quantity of available processors.

6 References

[1] M. G. Brockington, "A Taxonomy Of Parallel Game-Tree Search Algorithms". Journal of the International Computer Chess Association, 19(3):162–174, September 1996.

[2] 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.

[3] R. Feldmann, B. Monien, P. Mysliwietz, O. Vornberger, Distributed Game Tree Search, Parallel Algorithms for Machine Intelligence and Vision, V. Kumar, L. N. Kanal, P. S. Gopalakrishnan (Editors), Springer-Verlag, pp. 66-101, (1990) .

[4] J. Loy, “Famous checkers games”, Availabe at http://www.jimloy.com/checkers/checkers.htm#famous-20

[5] 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.

[6] T. A. Marsland, and M. S. Campbell, Parallel Search of Strongly Ordered Game Trees. ACM Computing Surveys, Vol. 14, No. 4, pp. 533-551.

[7] MPI Forum, “MPI: A Message-Passing Interface Standard Version 2.2”, Sep/2009. Available at: http://www.mpi-forum.org/docs.

[8] S. Russel, and P. Norvig, Inteligência Artificial - Uma abordagem Moderna, 2nd ed. Editora Campus, 2004.

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

[10] Schaeffer, J. et al. Chinook: World Man-Machine Checkers Champion. 2008. Perfect Play: Draw!, http://www.cs.ualberta.ca/~chinook/index.php.

[11] J. Schaeffer, R. Lake, P. Lu, M. Bryant, Chinook: The World Man-Machine Checkers Champion. AI Magazine 17(1), 21-29 (1996).

[12] N. N. Schraudolph, P. Dayan, T. J. Sejnowski, "Learning to evaluate go positions via temporal differences methods", Computational Intelligence in Games Studies in Fuzziness and Soft Computing, Spring Verlag, v.62, 2001.

[13] A. S. Tanenbaum and M. V. Steen, Distributed Systems: Principles and Paradigms, Prentice Hall, 2 edition, 2006.

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

[15] 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.

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