A Pendulum Effect of Expert Playing in Games

Dimitris Kalles
School of Science and Technology
Hellenic Open University
Patras, Greece
Email: kalles@eap.gr

Panagiotis Kanellopoulos
CTI "Diophantus"
and University of Patras
Patras, Greece
Email: kanellop@ceid.upatras.gr

***Abstract*—When learning how to play a strategy board game, one can measure the relative effectiveness of the learned policies by assessing how often a player wins and how easily these wins are scored. Experimental evidence also shows that when one of the competing players is trained by a sophisticated tutor, performance benefits also flow to the opponent. We present comprehensive experimental evidence that the level of tutor effectiveness is best demonstrated by the improvement of the tutored players opponent; this performance change is termed the pendulum effect.**

***Keywords*—board games; reinforcement learning; neural networks; function approximation; temporal difference learning**

I. Introduction

Motivation: Quite a few artificial intelligence techniques have made significant inroads into problems that demonstrate a human-computer interaction component. Learning how to play games is a prime example of such an application; Shannon [1] and Samuel [2] provided the first stimulating examples; Deep Blue defeated Kasparov at chess in 1997 [3] and, more recently, Schaeffer's team solved checkers completely [4], though the latter two are more indicative of the strength of search techniques. Today, advances in machine learning allow us to present "syllabus" of experience to a generic learning mechanism and expect that it can formulate playing knowledge.

Arguably one of the strongest contributions to that problem was the development of the TD(λ) method for temporal difference reinforcement learning [5] [6], which was successfully demonstrated in TD-Gammon for the game of backgammon [7]; therein using reinforcement learning techniques and self-playing, a performance comparable to that demonstrated by backgammon world champions was achieved.

Implementing a computer's strategy is a key point in strategy games. By the term strategy we broadly mean the selection of the computer's next move based on a variety of factors which can include its current situation, the opponent's situation and consequences of that move (or, possible next moves of the opponent). In our research, we use a strategy game to gain insight into how we can evolve game playing capabilities, as opposed to how we program such capabilities (maybe, using a heuristic). Although the operational goal of achieving improvement (measured in a variety of ways) is usually achieved in several experimental settings [8] [9] [10], the actual question of which training actions help realize this improvement is central if we attempt to devise an optimized training plan. A possible instantiation of this optimizing problem might be: given the possibility of turning to an expert for a limited number of games, is it better to ask the expert to provide guidance at the start of a learning session, at the end of it, or at intermediate points?

Exploiting expert interaction in trainable systems has been recently described according to three major directions [11]. In the first one, inspired by mainstream personalization, one collects raw items of user behavior and attempts to elicit habit, structure, or intention from such data [12]. The second direction is based on how control is exerted to generate actions that minimize some measure of a knowledge gap [13]. A third dimension is whether the expert tutor is a part of the training mechanism at all times, maybe with continuous but simple feedback [14].

A popular combination of these directions is to use an expert for focused guidance and, subsequently, to use some mechanism of self-improvement up to a point where, again, the expert will be called into action to provide a correction of direction, if required; this can go ad infinitum. So, the line between what constitutes sparseness and what constitutes density of learning examples (when the computer acts as a student) is quite fine [15].

Our contribution: Our workbench on this field is a simple board game, RLGame; for legacy reasons, the name draws on reinforcement learning being used as the basic learning mechanism [16]. Earlier evidence suggested that, when one of the competing players is trained by a sophisticated tutor (for example, as implemented by minimax), the other player also benefits since it is forced into unexplored ground and manages to learn new tactics that allow it to swiftly improve its performance [17]. To stress the interestingness of such a clearly observable change in performance between two competing players, this was called the pendulum effect.

So, the contribution of this paper is the design and carrying out of a comprehensive experimentation to investigate the pendulum effect, associating it with the rational intuitions that a tutor's impact is best assessed by its absence and that one learns better by facing a strong opponent. To arrive at these results we have designed and carried out several millions of games with diverse configurations in a distributed fashion over a grid computing infrastructure.

II. A brief background on a game workbench

The RLGame [16] is played by two players on an $n \times n$ square board. Two $\alpha \times \alpha$ square bases are on opposite board corners; these are initially populated by β pawns for each player, with the white player starting off the lower left base

and the black player starting off the upper right one. The goal is to move a pawn into the opponent's base or to force all opponent pawns out of the board.

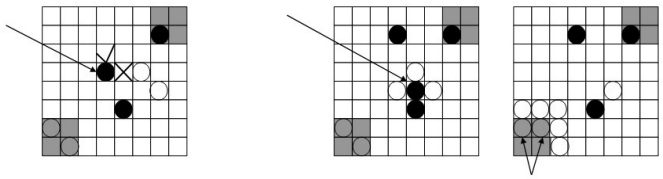

Fig. 1. Examples of game rules application

The base is considered as a single square, therefore a pawn can move out of the base to any adjacent free square. Players take turns and pawns move one at a time, with the white player moving first. A pawn can move vertically or horizontally to an adjacent free square, provided that the maximum (of the horizontal and the vertical) distance from its base is not decreased. A pawn that cannot move is lost (more than one pawn may be lost in one move).

The leftmost board in Fig. 1 demonstrates a legal ("tick") and an illegal ("cross") move for the pawn pointed to by the arrow, the illegal move being due to the rule that does not allow decreasing the maximum distance from the home (black) base. The rightmost boards demonstrate the loss of pawns, with arrows showing pawn casualties. A "trapped" pawn automatically draws away from the game; so, when there is no free square next to the base, the rest of the pawns of the base are lost.

A. Learning considerations

We use reinforcement learning to decide the next move. Since each player observes the full board, RL uses true "states" of the environment; this ideal case underpins a lot of the associated theory [6].

Temporal difference (TD) learning [5] is a combination of Monte Carlo and dynamic programming ideas. TD methods update estimates based in part on other learned estimates, without waiting for a final outcome (*bootstrapping*). Whereas Monte Carlo methods must wait until the end of the episode to determine the increment to $V(s_t)$ (only then is the reward known), TD methods need wait only until the next time step. *Eligibility traces* are one of the basic mechanisms of reinforcement learning and can be seen as a temporary record of the occurrence of an event, such as the visiting of a state. When a TD error occurs, only the eligible states or actions are assigned credit or blame for the error.

The value function on the game state space is approximated with neural networks [6] [16], where each next possible move and the current board configuration are fed as input and the network outputs a score that represents a belief degree that one will win by making that move. Learning employs an ϵ-greedy policy with $\epsilon = 0.1$ (the system chooses the best-valued action with a probability of 0.9 and a random action with a probability of 0.1). All non-final states are assigned the same initial value which, after each move, is updated through TD($\lambda = 0.5$), thus charting a middle course between considering the influence of the value most recently observed and the influence of a final value [5]; credit assignment is thus practically restricted to the last $6 - 7$ moves.

Going into some details in the neural network implementation, we note that we actually use two networks, one responsible for each player's behavior, because the value of each state depends on which player has the move token. This facilitates further experimentation, because we can try different configurations for the two opponents.

Now, for each state s, where s_t is the state selected at time t, we have calculated its eligibility trace as follows [18]:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s), & s \neq s_t \\ 1, & s = s_t \end{cases}$$

We used $\gamma = 0.95$ and $\lambda = 0.5$. Additionally, we used "vanilla" back-propagation for the neural networks. We updated the network weights vector by $\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha\delta_t\vec{e}_t$, where δ_t is the TD error defined as $\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$ and $\vec{e}_t$ is a vector of eligibility traces, one for each component of $\vec{\theta}_t$. The eligibility traces are updated according to $\vec{e}_t = \gamma\lambda\vec{e}_{t-1} + \nabla_{\vec{\theta}_t} V_t(s_t)$, with $\vec{e}_0 = \vec{0}$. Rewards are assigned for loss/win of games and for loss/capturing of pawns.

On network representation issues, each of the $n^2 - 2\alpha^2$ empty squares is assigned two input nodes, one for each player, which describe whether there is a pawn on that square. Two more nodes show if a pawn has captured the enemy base, and eight more nodes serve to show the number of pawns that still reside in the bases; we check if this number exceeds $\beta/4$, $\beta/2$ or $3\beta/4$ and "turn on" the appropriate input node(s). Thus, we have a total of $2n^2 - 4\alpha^2 + 10$ input nodes. We use $h = n^2 - 2\alpha^2 + 5$ hidden nodes and apply the standard logistic sigmoid function to all of them. We also apply the same logistic sigmoid function to the output node. We can, thus, view the output as the probability of winning the game starting from a particular state.

B. Reviewing the effects of expert involvement

Earlier experimentation culminated in the design of a metric to measure the relative effectiveness of two distinct policies [19]. Assuming that one has available a player, X, with its associated white and black components, W_X and B_X (the components being the neural networks), one can compare it to Y by first pairing W_X with B_Y for a CC^{1000} session (CC^{1000} stands for 1000 computer-vs-computer games), then pairing W_Y with B_X for a further CC^{1000} session, and subsequently calculating the number of games won and the average number of moves per game won.

The above metric captures shades of playing quality: for example, a (W_X+B_X):(W_Y+B_Y) result of $200 : 1800$ demonstrates a sizeable difference while a result of $900 : 1100$ less so. It has been also useful subsequently in testing a minimax algorithm for the moves of the white player [17].

Since minimax can be used with a different look-ahead set-up, initial experiments consisted of short sessions for various look-ahead values (note that a look-ahead of $2n + 1$, denoted by MC_{2n+1}, indicates $n + 1$ moves for the white player using minimax interleaved with n moves for the black player not using minimax). Each look-ahead experiment consisted of

100 MC_x games (totaling 500 games) and, since previous experimental evidence [16] suggests that RLGame is fair, we did not implement minimax for black for this type of experiments.

One examines MC_x experiments with the expectation that the white player's performance will be improved the longer experimentation is allowed to carry on. There is a simple reason for that: the white player is better situated to observe winning states and then update its learning data structures accordingly, also for those states that lead to a win but have not been yet reached via minimax. This behavior was observed for all initial experiments. Subsequently, a CC^{1000} session was run on the output of each of the MC_1, MC_3, MC_5, MC_7 and MC_9 sessions (totaling a further $5,000$ games) and this is where the pendulum effect was observed for the first time; namely, an abrupt reversal on performance, with the black player dominating the white one.

Specifically, it was observed that the minimax tutor for the white player was actually training the black one by forcing it to lose; when the tutor was drawn out of the game, the black player overwhelmed the white one, which then had to adapt itself again due to black pressure as fast as possible [17].

III. Detecting the pendulum effect

We now review the experiments that we designed and carried out to investigate the "pendulum effect" hypothesis, as the original experimental session was too short for any conclusive claim despite being indicative.

For a fixed combination of board size (n), base size (α), number of pawns (β) and exploitation-exploration trade-off (ϵ), we ran a CC^{10000} session, to serve as the fundamental self-playing benchmark. We also ran three MC^{100} sessions, each one with a different look-ahead value (1, 3, 5). Subsequently, we ran three further CC^{10000} sessions, basing each one of them on one of the concluded MC^{100} sessions (so, every MC_x^{100} session was followed by a CC^{10000} session). Such an arrangement eventually delivers seven logs in total.

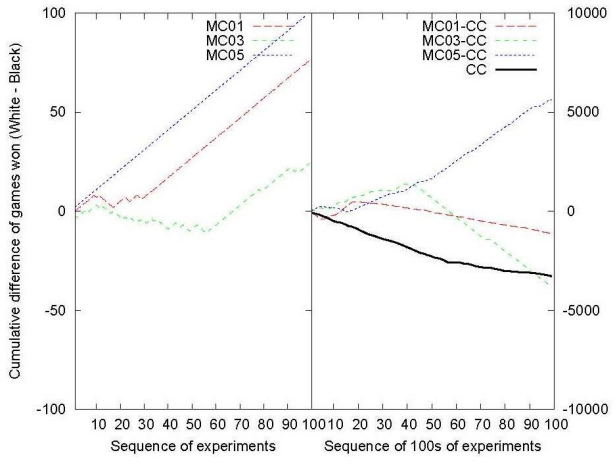

Fig. 2. Number of games won by each player in each session

Fig. 2 shows an indicative cumulative wins graph. The left part has a vertical axis spanning from -100 (all games won by black) to 100 (all games won by white), since any MC_x session is at most 100 games long. The right part has a vertical axis spanning from -10000 to 10000 to account for the length of the CC sessions. However, both parts are aligned with respect to their zero-value in their respective vertical axes. Dashed lines of varying density differentiate between MC_x and MC_x-CC variants; a solid line (only available in the right part) serves as the CC benchmark.

A group of seven sessions such as the above totals 300 MC_x games (3×100) and $40,000$ CC games ($4 \times 10,000$). To substantiate the existence of the pendulum effect, we experimented with a multitude of parameter combinations, as detailed in Table I; additionally for each such valid configuration we experimented with three values for the exploitation-exploration trade-off, ϵ: 0.1, 0.3 and 0.5.

TABLE I. A description of game configurations

Board size ($\mathbf{n}$)	$5, 6, 7, 8, 9, 10$
Base size (α)	$2, 3, 4$
Number of pawns (β)	$1, 2, 3, 4, 5, 6, 7, 8, 9, 10$

We did not experiment with ($n = 5$, $\alpha = 2$, $\beta = 1$) since it contains a forced win for the first player in 4 moves (or 7 semi-moves); still the remaining 357 configurations (of seven sessions each) correspond to over 10^5 MC_x games and over $14 \cdot 10^6$ CC games.

The pendulum effect is a manifestation of a power shift; whereas a player had a clear advantage over a series of games, that advantage is at some point eliminated by its opponent, which then scores successive wins, until the trend is reversed again, maybe *ad nauseam*. Detecting the pendulum effect implies that we also have a reference collection of results that can be clearly shown *to not be* an instance of that event; moreover, it also implies that we have access to enough samples to strengthen our claim. The standalone CC session in Fig. 2 (solid line) is a good example of a result that belongs to that reference collection; it is a pattern that that has been overwhelmingly observed across all standalone CC sessions (referred to as tabula rasa elsewhere in this paper).

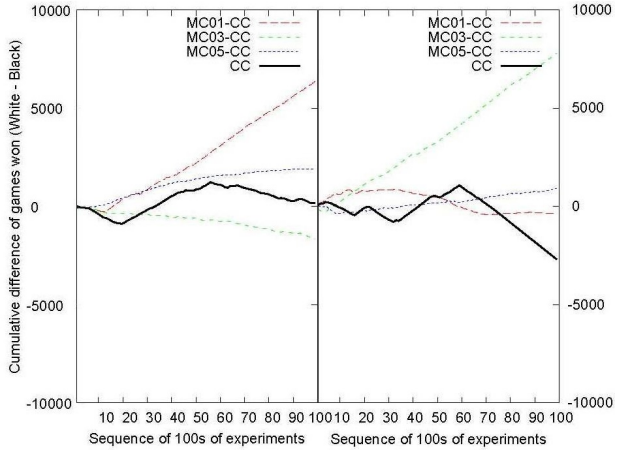

Fig. 3. Examples of the pendulum effect in *tabula rasa* experiments

While it may be straightforward to agree on what constitutes a relatively straight line, describing what constitutes instances of the pendulum effect at the standalone CC level is subtler. In Fig. 3 we show the basic alternatives (we have

combined two experiments to save space); therein the left part demonstrates how the pendulum effect brings about a balance of power, whereas the right part demonstrates an example where, for the length of the recorded session, one of the players eventually gains a sustainable edge (observe the rightmost end of the solid black lines with respect to the y-axis 0 value).

It was while inspecting cumulative performance graphs that our attention was drawn to the abrupt changes in performance; that triggered the investigation of the pendulum effect. This section reviews the results of that visual inspection which was fundamental to offering some high-level data to motivate further analysis.

Table II reviews the results of the visual inspection. Note that, while CC sessions are overwhelmingly likely to demonstrate a smooth behavior, increased occurrences of the pendulum effect are associated with larger exploitation (smaller exploration) values. This suggests that when the winner attempts to exploit what has been learnt and, so, is more likely to follow paths that will lead it to win again, the opponent is increasingly likely to learn and rebound. This is hardly a surprise since, when wandering, there are fewer things to exploit and fewer opportunities to learn from a winner.

TABLE II. RELATIVE FREQUENCY OF THE PENDULUM EFFECT IN $(119 \times 3 = 357)$ *tabula rasa* CC SESSIONS AND MC_x SESSIONS

	$\epsilon = \mathbf{0.1}$	$\epsilon = \mathbf{0.3}$	$\epsilon = \mathbf{0.5}$
Aggregate (CC)	11%	6%	4%
Aggregate (MC)	15%	9%	11%

The telling test for the pendulum effect is to investigate what happens when a tutor is absent. Essentially, we are interested in whether any MC_x-CC session in some configuration demonstrates more complex behavior (pendulum effect) than the respective CC baseline, also taking into account the MC_x session that preceded it. With reference to Fig. 2, it is obvious that the sequence of the MC_5 and MC_5-CC sessions does not demonstrate a pendulum effect compared to its CC reference; however, seeing MC_1-CC and its respective predecessor, MC_1, it is obvious that they demonstrate a more "unstable" behavior compared to the CC baseline (the group of MC_3 and MC_3-CC demonstrates a similar behavior too). However, in this context, exploitation does not drive learners as strongly as the minimax tutor does, which actively maintains a frontier of promising alternatives during game play.

Table III reviews these results; now, while all sessions are quite likely to demonstrate the pendulum effect, a smaller exploitation drive magnifies the importance of the minimax tutor and makes the pendulum effect more pronounced. Viewing these results vis-à-vis Table II makes it apparent that the MC^{100} sessions are associated with the subsequent emergence of the pendulum effect.

TABLE III. RELATIVE FREQUENCY OF THE PENDULUM EFFECT IN (1071) COMBINED MC_X-CC SESSIONS

	$\epsilon = \mathbf{0.1}$	$\epsilon = \mathbf{0.3}$	$\epsilon = \mathbf{0.5}$
$\mathbf{MC_1}$**-CC**	73%	90%	92%
$\mathbf{MC_3}$**-CC**	81%	88%	83%
$\mathbf{MC_5}$**-CC**	76%	85%	87%
Aggregate	76%	88%	87%

IV. ON THE VALIDITY AND THE IMPLICATION OF THE RESULTS

Millions of individual games have been carried out, across diverse game size configurations and exploitation-exploration policies to demonstrate the pendulum effect. By the relative inspection of CC and MC_x-CC sessions, those that initially look like an interesting minority in standalone CC sessions eventually turn up as the overwhelming majority when there is a previous tutoring (MC) stage.

It is justifiable to expect that the pendulum effect should arise. When two players interact in a vacuum devoid of other interactions (against other players), none of them has a clue as to whether the other player is inherently good or bad, apart from somehow measuring wins vs. losses. This is elementary opponent modeling. It is also the basis of co-evolutionary learning, since both players respond to the opponent's move while also trying to learn how to best respond. If both players are honest and none of them attempts to trick the opponent by not playing what it thinks it is best for itself, then each player gets an as accurate as possible picture of what its opponent thinks about the state of the game (exploitation is synonymous to honesty; exploration is subtler though). As a result, the weaker player is guided through value function updates which will decrease its performance gap compared to the stronger player. The pendulum effect arises when one of the players has access to a winning policy, since this creates a strong drive away from the stability point; that drive is subsequently compensated by a comparably strong drive towards the stability point again. And, obviously, a tutor (like minimax, for example) does tend to create a winning policy.

An alternative way of viewing the pendulum effect is to consider learning and un-learning (or, forgetting). Learning *and* forgetting do co-exist in a reinforcement learning context; good paths need occasional reinforcement so that learned (useful) value functions do not degrade in approximation quality when alternative paths are explored. Again, however, this seems to be linked to the relative density of high value advice (for example, expert playing), which underlines the conceptual proximity of reinforcement learning to the scaffolding concept in the situated learning approach [20]. A similar problem exists when a tutor unavoidably displays a behavior that is prone to deviations, maybe due to physical peculiarities (for example, when piloting a helicopter, a "best" trajectory from a departure point to a destination can be realized as a selection of partially similar, real trajectories as followed by a human pilot). In such a context, the learning mechanism has to be designed with a view to subsequently factor out the deviations [21]. This can be quite expensive, since the existence of many similar experiences may decrease the amount of "new" space we can afford to explore and thus result in significantly slower performance improvement.

Detecting the occurrence of the pendulum effect right now does not offer insight into how the pendulum effect may be associated to the relative sparseness or density of alternatives. Specifically, matches of one-pawn-per-player are rather simple and their resilience to game tree pathology is inferior to variants utilizing many pawns. Additionally, such resilience is further compromised by small boards where shorter paths to game conclusion amplify the relative importance of errors in the minimax estimates [22].

V. Conclusions and future directions

This paper presented carefully designed experiments, at a large scale, to support the claim that expert tutoring, which improves the performance of computer players in a board game, is mostly evident when the tutor resigns and the impact of its absence is examined.

We assigned such a tutor role to minimax and we elaborated on our experimental setup using a key statistic: number of games won by each player throughout the tutoring session and throughout a subsequent non-tutoring session. Calculating this statistic is trivial but associating it with the depth of the tutoring expertise to measure learning effectiveness is not obvious. This is mostly because both competing players attempt to fine tune their performance vis-à-vis their opponent, so differences sometimes smooth out. Our experimental evidence and intuition both suggest that the larger the difference, the stronger the smoothing trend. This is the pendulum effect, robustly demonstrated in our experiments.

The importance of the pendulum effect will be strengthened if it can be demonstrated across similar zero-sum two-player games, possibly by employing more sophisticated and faster playing and by investigating more complex tutors. Another significant contribution would be to improve the automatic classification of pendulum effect cases. Besides devising more accurate formulae, one might also employ a machine learning approach, using data series and expert judgment as input. Currently, we are investigating both options and progress on that direction would likely facilitate subsequent analysis on the effectiveness of learning systems at large.

The pendulum effect promotes interactive evolutionary learning. Therein, one would ideally switch from focused tutor-based training (expert-play) to autonomous crawling (self-play); we first generate an initial advantage for one of the players and then explore the state space aiming to catch up on behalf of the other. However, as we have discovered during this work, the interactivity requirements of this iterative process also demand a suitable environment to supports this development. To put the tutor in the loop we are currently working to integrate our development with scientific workflow systems over grids [23] [24] [25].

The pendulum effect raises a fundamental question: when do we stop learning according to a policy and try something else? If the pendulum effect can be computationally traced, then we can detect performance plateaus where players learn little and should divert to explore new learning opportunities. In artificial agent societies, where the concepts of competition, co-evolution and co-learning are inherent, a central theme is to select a good partner with which to co-learn for some time, before switching partners. Viewing an opponent as a partner is key in co-evolutionary learning and being able to estimate whether such a partnership is profitable will be invaluable.

References

[1] C. Shannon (1950). "Programming a Computer for Playing Chess", Philosophical Magazine 41(4): 265-275.

[2] A. Samuel (1959). "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research and Development 3: 210-229.

[3] F-H. Hsu (2002). Behind Deep Blue: Building the Computer that Defeated the World Chess Champion. Princeton University Press.

[4] J. Schaeffer, Y. Bjoernsson, N. Burch, A. Kishimoto, M. Mueller, R. Lake, P. Lu, and S. Sutphen (2007). "Checkers is Solved", Science 317(5844): 1518-1522.

[5] R. Sutton (1988). "Learning to Predict by the Methods of Temporal Differences", Machine Learning 3(1): 9-44.

[6] R. Sutton, A. Barto (1998). Reinforcement Learning - An Introduction, MIT Press, Cambridge, MA.

[7] G. Tesauro (1995). "Temporal Difference Learning and TD-Gammon", Communications of the ACM 38(3): 58-68.

[8] I. Ghory (2004). "Reinforcement Learning in Board Games", Technical report CSTR-04-004, Department of Computer Science, University of Bristol.

[9] D. Osman, J. Mańdziuk (2005). "TD-GAC: Machine Learning Experiment with Give-Away Checkers", Issues in Intelligent Systems. Models and Techniques, M. Dramiski et al. (eds.), pp. 131-145.

[10] K. Walędzik, J. Mańdziuk (2010). "The Layered Learning Method and its Application to Generation of Evaluation Functions for the Game of Checkers", 11th International Conference on Parallel Problem Solving from Nature, Kraków (Poland), 543-552.

[11] A.L. Thomaz and C. Breazeal (2008). "Teachable Robots: Understanding Human Teaching Behavior to Build More Effective Robot Learners", Artificial Intelligence 172: 716-737.

[12] T.M. Mitchell, S. Wang and Y. Huang (2006). "Extracting Knowledge about Users' Activities from Raw Workstation Contents", 21st National Conference on Artificial Intelligence, Boston, MA, pp. 181-186.

[13] D. Cohn, Z. Ghahramani and M. Jordan (1996). "Active Learning with Statistical Models", Journal of Artificial Intelligence Research 4: 129-145.

[14] F. Kaplan, P.-Y. Oudeyer, E. Kubinyi and A. Miklosi (2002). "Robotic Clicker Training", Robotics and Autonomous Systems 38(34): 197-206.

[15] H.J. van den Herik, H.H.L.M. Donkers, P.H.M. Spronck (2005). "Opponent Modelling and Commercial Games", IEEE Symposium on Computational Intelligence and Games, Essex University, Colchester, UK, pp 15-25.

[16] D. Kalles, P. Kanellopoulos (2001). "On Verifying Game Design and Playing Strategies using Reinforcement Learning", ACM Symposium on Applied Computing, special track on Artificial Intelligence and Computation Logic, Las Vegas.

[17] D. Kalles, D. Kanellopoulos (2008). "A Minimax Tutor for Learning to Play a Board Game", AI in Games Workshop, 18th European Conference on Artificial Intelligence, Patras, Greece, pp. 10-14.

[18] S.P. Singh, R. Sutton (1996). "Reinforcement Learning with Replacing Eligibility Traces", Machine Learning 22: 123-158.

[19] D. Kalles (2008). "Player Co-Modelling in a Strategy Board Game: Discovering How to Play Fast", Cybernetics and Systems 39(1), 1-18.

[20] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, J. Lieberman, H. Lee, A. Lockerd, and D. Mulanda (2004), "Tutelage and Collaboration for Humanoid Robots", International Journal of Humanoid Robotics 1(2): 315-348.

[21] A. Coates, P. Abbeel and A.Y. Ng (2009). "Apprenticeship Learning for Helicopter Control", Communications of the ACM 52(7): 97-105.

[22] B. Wilson, A. Parker, D.S. Nau (2009). "Error Minimizing Minimax: Avoiding Search Pathology in Game Trees", International Symposium on Combinatorial Search, Lake Arrowhead, CA.

[23] C. Kiourt and D. Kalles (2012). "Social Reinforcement Learning in Game Playing", IEEE International Conference on Tools with Artificial Intelligence, Athens, Greece.

[24] C. Kiourt and D. Kalles (2013). "Building A Social Multi-Agent System Simulation Management Toolbox", 6th Balkan Conference on Informatics, Thessaloniki, Greece.

[25] A. Georgas, D. Kalles and V. Tatsis (2014). "Scientific Workflows for Game Analytics", Encyclopedia of Business Analytics and Optimization (in press), J. Wang (ed), IGI Global.