Computational Intelligence in Mind Games

Jacek Mańdziuk

Faculty of Mathematics and Information Science
Warsaw University of Technology, Poland.
mandziuk@mini.pw.edu.pl

Summary. The chapter considers recent achievements and perspectives of Computational Intelligence (CI) applied to mind games. Several notable examples of unguided, autonomous CI learning systems are presented and discussed. Based on advantages and limitations of existing approaches a list of challenging issues and open problems in the area of intelligent game playing is proposed and motivated.

It is generally concluded in the paper that the ultimate goal of CI in mind game research is the ability to mimic human approach to game playing in all its major aspects including learning methods (learning from scratch, multitask learning, unsupervised learning, pattern-based knowledge acquisition) as well as reasoning and decision making (efficient position estimation, abstraction and generalization of game features, autonomous development of evaluation functions, effective preordering of moves and selective, contextual search).

Key words: challenges, CI in games, game playing, soft-computing methods, Chess, Checkers, Go, Othello, Give-Away Checkers, Backgammon, Bridge, Poker, Scrabble.

1 Introduction

Playing games has always been an important part of human activities and the oldest mind games still played in their original form (Go and Backgammon) date back to 1,000 - 2,000 BC.

Games also became a fascinating topic for Artificial Intelligence (AI). The first widely known "AI approach" to mind games was noted as early as 1769 when Baron Wolfgang von Kempelen's automaton Chess player named *The Turk* was presented at the court of Empress Maria Theresa. *The Turk* appeared to be a very clever, actually unbeatable, Chess player who defeated among others Napoleon and the Empress Catherine of All the Russias. It took a few decades to uncover a very smart deception: a grandmaster human player was hidden inside the Turk's machinery and through a complicated construction of levers and straddle-mounted gears was able to perceive opponent's moves and make its own ones. The history of *The Turk* was described independently by several people, including the famous American novelist Edgar Allan Poe[78]. Although *The Turk* had apparently nothing in common with AI, the automaton is a good illustration of humans' perennial aspiration for creating intelligent machines able to defeat the strongest human players in popular mind games.

Serious, scientific attempts to invent "thinking machines" able to play mind games began in the middle of the previous century. Thanks to seminal papers devoted to programming Chess [93, 110, 74] and Checkers [82] in the 1950s., games remained through decades an interesting topic for both classical AI and CI based approaches.

One of the main reasons for games' popularity in AI/CI community is the possibility to obtain cheap, reproducible environments suitable for testing new search algorithms, pattern-based evaluation methods or learning concepts.

On the other hand the "human aspect" of game playing should not be underestimated. This is why from the very beginning of AI "involvement" in games, it was Chess - the queen of mind games - that attracted special attention and in 1965 was even announced "*the Drosophila of Artificial Intelligence*" by the Russian mathematician Alexander Kronrod.

The focus of this chapter is on the most popular mind games, such as Chess, Checkers, Go, Othello, Backgammon, Bridge, Poker and Scrabble. The reason for choosing these particular games is two-fold: (1) they are all very popular and played all over the world, (2) for decades they have been a target for AI/CI research aiming at surpassing human supremacy. Certainly, there are many other interesting and highly competitive mind games (e.g. Shogi, Chinese Chess, Hex, Amazons, Octi, Lines of Actions, Sokoban), which do not appear in this chapter, mainly due to their lesser popularity - although, some of them are becoming more and more prominent. Also other types of computer games different from mind games, such as skill, adventure, strategic, war, sport, negotiating, racing and others are not considered in this chapter.

In order to make the notation clear and concise, henceforth any reference to CI systems (approaches) will address soft-computing-based systems (*neural networks, genetic or evolutionary algorithms, fuzzy systems, reinforcement learning, Bayesian methods, probabilistic reasoning, rough sets)* capable of learning and autonomous improvement of behavior[1].

It seems worth noting that the aim of this chapter is by no means to criticize the achievements of traditional AI methods in the game playing domain. On the contrary the hitherto accomplishments of AI approaches are undisputable and speak for themselves. Our goal is rather to express the belief that other, alternative ways of developing "thinking machines" are possible and urgently needed. These methods include cognitive, knowledge-free approaches capable of learning from scratch based merely on an unguided training process,

[1] Certainly, a distinction between AI and CI is to some extent a matter of convention. The above proposal is consistent with the author's point of view.

e.g. evolutionary or reinforcement-type, or based on an agent's experience obtained gradually through (self-)playing or in a supervised training process, e.g. with neural nets, but again without explicit implementation of human experts' knowledge.

In our opinion the need for further development of these, knowledge-free methods is unquestionable, and the ultimate goal that can be defined is building a truly autonomous, human-like multi-game playing agent. In order to achieve this goal several challenging problems have to be addressed and solved on the way.

The chapter is organized as follows. In the next section a brief description of state-of-the-art accomplishments in the most popular mind games are presented. Section 3 starts with a general discussion on the challenging issues in the game playing domain and presents further motivation for pursuing this research topic. Next, in several subsections particular challenges are considered one-by-one in more detail. Conclusions are presented in the last section.

2 State-of-the-Art Playing Programs

In this section some of the best playing programs in the most popular games are briefly introduced. In some games (Scrabble, Chess, Checkers, Othello, Backgammon) the supremacy of the presented systems over humans and other artificial agents was officially acclaimed either by gaining the respective World Champion title or by defeating the best human players. In the remaining games considered here (Poker, Bridge, Go) humans are still far ahead of machines.

Scrabble. One of the first programs that achieved a world-class human level in a non-trivial game was *Maven* - a Scrabble playing program written by Brian Sheppard [94]. In the 1990s. Maven successfully challenged several world top Scrabble players including (then) North America Champion Adam Logan and world champion Joel Sherman (both matches took place in 1998). Strictly speaking the supremacy of Maven was demonstrated only in North America - i.e. for US English, but adaptation of Maven to another dictionary is straightforward. Actually, it was later on adapted to UK English, International English, French, Dutch and German.

Scrabble is a game of imperfect information with a large branching factor, and as such is very demanding for AI research. The key to Maven's success lies in efficient, selective move generation and perfect endgame play supported by B^* search algorithm[2]. Maven, similarly to TD-Gammon described in sect. 3.1, uses game scenarios simulation or "rollouts", which proved to be a strong evaluation technique. The implementation details are presented in [94].

[2] As soon as the bag is empty, Scrabble becomes a perfect information game, since one can deduce the opponent's rack by subtracting the tiles one can see from the initial distribution.

In [94] Sheppard stated: "*There is no doubt in my mind that Maven is superhuman in every language. No human can compete with this level of consistency. Anyone who does not agree should contact me directly to arrange a challenge match*". So far, no-one tried ...

Chess. Presumably the most striking achievement of AI in games was *Deep Blue II's* victory over Garry Kasparov - the World Chess Champion (at the time the match was held) and one of the strongest Chess players in the history. This event ended a nearly 50-year era of Chess programming efforts which started in Shannon's paper [93]. The evaluation function of Deep Blue II was composed of over $8,000$ features implemented in a single chess chip. 480 such chips formed an extremely fast, massively parallel search system based on 30-node cluster allowing for total search speed between 100 million and 330 million positions per second depending on their tactical complexity [20] or 50 billion positions in three minutes - the average time allotted for each move [48]. Certainly, except for tuning thousands of weights in the evaluation function, a lot of other problems concerning massively-parallel, non-uniform, highly-selective search or creation and analysis of the extended opening book and endgame database had to be solved in order to achieve the final result. There in no doubt that Deep Blue II is a milestone achievement from an engineering and programming point of view [45, 20]. From a CI viewpoint much less can be said since the system did not take advantage of any learning or self-improvement mechanisms.

The victory of Deep Blue II attracted tremendous interest among game playing researches and also had an undisputed social and philosophical impact on other people, not professionally related to science (New York's Kasparov vs. Deep Blue II match was followed on the Internet by thousands of people all over the world). On the other hand the result of the match should not lessen further research efforts aiming at developing an "intelligent" Chess playing program equipped with cognitive skills similar to those used by human Chess grandmasters.

Since Deep Blue's era several other Chess programs, e.g. *Shredder, Fritz, Deep Junior* or the recent Chess supercomputer - *Hydra* played successfully against human grandmasters, but none of these matches gained comparable public attention and esteem.

Checkers. The first computer program that won a human world championship was *Chinook* - the World Man-Machine Champion developed by Jonathan Schaeffer and his collaborators from the University of Alberta [87, 89, 84]. Chinook's opponent in both 1992 and 1994 was Dr Marion Tinsley - the ultimate Checkers genius, who was leading the scene of Checkers competitions for over 40 years losing during that period as few as only 7 games (including the 2 lost to Chinook)! As Schaffer stated: Tinsley was "*as close to perfection as was possible in a human*" [89].

Similarly to Deep Blue, Chinook can be regarded as a large scale AI engineering project including all vital aspects of AI design: efficient search, well-tuned evaluation function, opening book and endgame database. Special care was taken over specific tactical combinations (e.g. exchanging one of our own pawns for two of an opponent's) and these situations were carefully analyzed and coded in special tables. The evaluation function was linear and composed of over 20 major components, each of which having several heuristic parameters. All evaluation function weights were hand-tuned. During the re-match in 1994 Chinook was equipped with a complete 7-piece endgame database (i.e. exact solutions of all endings of 7 pieces or less) and with a 4×4 subset of an 8-piece database (i.e. all endings in which each side was left with exactly 4 pieces) [89]. At the time of writing this chapter the 9-piece database is completed and the 10-piece one is on the way [85].

The ultimate goal of Schaeffer and his group is to build the perfect Checkers player by solving the game of Checkers. Recently it was announced that another opening (already the second one) has been solved - proven to be a draw [85].

Othello. Another game in which computers outperformed over humans is Othello. In 1997, just a few months after Deep Blue's victory, Michael Buro's program *Logistello* [19], running on a single PC machine, decisively defeated the then Othello World Champion Takeshi Murakami with the score $6 - 0$. Taking into account that Logistello was not implemented in a special hardware and considering the convincing result of the match as well as post-mortem analysis of the games which showed that program was not in trouble in any of the six games played, further advances Buro's achievement.

The main factors contributing to this strong victory were: (1) new, efficient way of feature selection for the evaluation function [17]. Starting from a set of predefined, atomic features, various Boolean conjunctions of these simple features were considered and their weights calculated by the linear regression based on several million training positions labelled either by their true mini-max value or an approximation of it. (2) Forward pruning method ProbCut (and Multi-ProbCut) capable of cutting out the most probably irrelevant subtrees with predefined confidence [16]. In short, the method generalizes from shallow search results to deeper search levels by statistically estimating the coefficients in the linear model approximating the relationship between shallow and deep mini-max search results. (3) Automatic opening book development, which takes advantage of the search results along the promising lines not played so far and consequently allows potentially interesting opening alternatives in the future [18].

All three above mentioned aspects of game playing are game independent and possibly applicable to other two-player board games. On a more general note these methods are in line with a human way of playing which includes building up the evaluation function, performing selective search or looking for new variants in the known game openings.

Backgammon. The state-of-the-art Backgammon playing program is *TD-Gammon* [103, 104, 105] written by Gerald Tesauro. The program implements

Temporal Difference learning and is one of the archetypal examples of successful CI approaches in games. The main features of TD-Gammon are discussed in more detail in sect. 3.1.

In the above mentioned games the human supremacy has already been successfully challenged by AI/CI programs. Among the most popular games there are only three left in which humans have not been conquered - (yet!). These are: Poker, Bridge and Go.

Poker. According to the results of the World Poker Robot Championship that took place in July 2005 the world's best playing Poker program is *PokerProbot*^{TM} [40] - the Amateur Robot Champion and, at the same event, the winner of the match with *Poki-X* [86] written by Jonathan Schaeffer and his group from the University of Alberta.

Since PokerProbot, written by Hilton Givens, is commercial software, very little is known about its internal characteristics and the history of its development. On the contrary the knowledge behind its main opponent Poki-X has been revealed [13]. Both programs were also confronted with one of the game's most accomplished professionals and World Series of Poker champion, Phil Laak - and both lost by a large margin.

The reasons why Poker is so difficult for AI/CI is related to the fact that it is an *imperfect information* game since the other player's cards are hidden. Additionally Poker players often use various kinds of deception or bluffing. Non-accessibility to the whole information (as opposed to Chess, Checkers, and other board games not involving elements of chance) requires (1) modeling of the opponents, (2) applying risk management techniques and (3) using a dynamical, context sensitive and in most cases probabilistic evaluation function rather than a static one. These issues are expanded in [13] with regard to the Poki system. Poki uses a sophisticated, multi-stage betting strategy which includes the evaluation of effective and potential hand strength, opponent's modeling and probabilistic simulations based on selective sampling[3]. All the above issues are essential for successful machine Poker playing. For the scope of this chapter the problem of *opponent modeling* (discussed further in sect. 3.6) is of particular interest.

Bridge. Since 1997 the World Computer-Bridge Championship has been organized each year by The American Contract Bridge League. The regular participants in this annual event are *Jack*, *Bridge Baron*, *WBridge5*, *Micro Bridge*, *Q-Plus Bridge* and *Blue Chip Bridge*. Each of these programs enjoyed some success in previous contests but the most renowned one is the Dutch program named Jack by Hans Kuijf and his team [53]. Jack won the title in 2001, 2002, 2003, 2004 and was placed second in 2005 after WBridge5, though was in the first place in the so-called Round Robin - the aggregated result of direct pairwise comparison.

[3] The idea of selective sampling in a world-class playing programs was also applied to Backgammon [105], Scrabble [94] and Bridge[38].

An interesting phenomenon among top bridge programs is GIB (Ginsberg's Intelligent BridgePlayer) written by Matthew L. Ginsberg - historically the first strong bridge playing program. GIB uses partition search (the cutting tree technique defined by Ginsberg) and Monte Carlo sampling techniques for both the bidding and cardplay phases [39].

The level of play of the best computer programs is gradually improving and currently they are able to play on equal terms against intermediate human players.

Go. The game of Go is widely considered as the most demanding, grand AI/CI challenge in the mind games domain. Despite simple rules and no pieces differential, playing the game well is yet a non-achievable task for machines. The most advanced Go programs can still be easily beaten by intermediate amateur human players [4]. There are several reasons for this situation. First of all, Go has a very high branching factor, which effectively eliminates brute-force-type exhaustive search methods. But the huge search space is not the only impediment in efficient play. The very distinctive feature that separates Go and other popular board games is the fact that static positional analysis of the board is orders of magnitude slower in Go than in other games [73]. Additionally, proper positional board judgement requires performing several auxiliary tactical searches oriented on particular tactical issues [73]. Due to the variety of positional features and tactical threats it is highly probable that, as stated in [73], "*no simple yet reasonable evaluation function will ever be found for Go*". Another difficult problem for machine play is the "pattern nature" of Go. On the contrary to humans, who posses strong pattern analysis abilities, machine players are very inefficient in this task, mainly due to the lack of mechanisms (either predefined or autonomously developed) allowing flexible subtask separation. The solutions for these subtasks need then to be aggregated - considering complex mutual relations - at a higher level and provide the ultimate estimation of the board position. Instead, only relatively simple pattern matching techniques are implemented in the current playing programs [72, 73].

Due to the still preliminary stage of Go playing programs' development it is hard to point out the stable leader among them. Instead, there exists a group of about ten programs playing on a more or less comparable level. These include: *Many Faces of Go*, *Go4++*, *Handtalk*, *GoIntellect*, *Explorer*, *Indigo* and a few more. A detailed discussion on the development of Go playing agents can be found in [14, 73]. An interesting proposition for researchers aiming to write their own Go program is the open source application GnuGo [15].

[4] Unless otherwise stated in the whole paper we will refer to the game played on a regular 19 x 19 board.

3 The Challenges

Recent advances of AI in the most popular mind games, which led to spectacular challenging the human supremacy in Chess, Checkers, Othello or Backgammon provoke the question: "*Quo vadis mind games research?*". Do we still need to pursue mind game research or maybe defeating human world champions is (was) the ultimate, satisfying goal?

Naturally, when considering the quality of machine playing in particular game as the sole reference point the only remaining target might be the further extension of the machines' leading margin in the man-machine competition (since it is doubtful that the improvement of human players would be adequate to the one of computer players). But improvement of efficiency is not the only and not even a sufficient motivation for further research.

I would argue that good reasons for game research concern ***the way*** in which high playing competency is accomplished by machines. On one side there are extremely powerful AI approaches in which playing agents are equipped with carefully designed evaluation functions, look-up tables, perfect endgame databases, opening databases, grandmaster game repositories, sophisticated search methods (e.g. B*[11], SSS*[99], NegaScout [80], MTD(f) [76, 77], conspiracy numbers [65]) or search enhancements (e.g. singular extensions [2], null moves [9, 44], ProbCut [16], and other [83]) and a lot of other predefined, knowledge-based tools and techniques that allow making high quality moves with enormous search speed. On the other side there are soft, CI-based methods relying mainly on knowledge-free approaches, extensive training methods including reinforcement learning, neural networks, self-playing and even learning from scratch based merely on the final outcomes of the games played. Application and development of these soft techniques pose several challenging questions which are discussed in the remainder of this section. First, in section 3.1 some well-known successful examples of **autonomous learning** in game playing and challenging, open problems related to this type of learning are discussed. Section 3.2 addresses the issue of **creativity** understood as *ad hoc* **knowledge discovery** which may emerge as a result of deliberately designed training process. Section 3.3 is devoted to **intuition**, implementation of which in artificially built systems seems to be one of the grand challenges not only in game playing domain. Section 3.4 considers the problem of **abstraction and generalization** of knowledge possessed during the learning process. In particular the problem of how to generalize from shallow-depth search is still unsolved and considered a challenge. Another challenging problem is efficient **pre-ordering of moves** in the search algorithms (section 3.5). Although a lot of results have been published in this area, the problem - addressed generally - still remains open. Section 3.6 touches on the problem of **opponent modeling** which is one of the central issues in games with imperfect information, especially those in which deception and bluffing are inherent elements, such as Poker or Perudo. Finally, section 3.7 concerns **universality of approaches** and tools applied within CI. The development of game independent, universal training processes applicable to a wide range of games is one of the relevant current research problems. Possible approaches include **multitask learning** and **lifelong learning**.

3.1 Autonomous Learning

One of the most distinctive features of CI-based systems is the ability to improve themselves through a (self)-learning process. Unlike classical AI approaches which rely on carefully designed, hand-crafted evaluation functions reflecting expert knowledge about various game aspects, the CI systems, given some initial knowledge, are able to improve their performance through learning or evolution.

Construction of game playing agents capable of learning based on experience is one of the challenging issues. There are several notable examples of such systems, e.g. Tesauro's Neurogammon and TD-Gammon, Baxter's KnightCap, Schaeffer's TDL-Chinook, Thrun's NeuroChess, or Fogel's Anaconda - to mention only a few of them. A brief description of the above seminal achievements is presented in the remainder of this section, followed by a general discussion on their strengths and weaknesses as well as related open problems.

Certainly the systems described below by no means pretend to be a complete catalogue of CI achievements in games. They are rather a partial collection of milestone accomplishments subjectively chosen by the author. Other CI approaches to most popular games include for example [112, 109, 36, 50, 41, 34] in Chess, [1, 61] in Checkers, [63, 75, 54] in Give-Away Checkers, [69, 113] in Othello, [52] in Rummy, [26, 22, 92] in Iterated Prisoner's Dilemma, [79] in Backgammon, [4, 5] in Poker, [90, 91, 29, 81] in Go.

Neurogammon and **TD-Gammon**. The first world-class accomplishment in the field of CI in games was TD-Gammon program [103, 104, 105] and its predecessor - Neurogammon [102], both written by Gerald Tesauro for playing Backgammon - an ancient two-player board game. In short, the goal of the game is to move one's checkers from their initial position on the one-dimensional track to the final position (players make moves in the opposite directions). The total distance that pieces belonging to one player can move at a given turn depends on the score of the two dices which are thrown by a player at the beginning of a move. Rolling dices introduces randomness into the game. Based on the dices' score the player makes a decision regarding which pieces to move forward and of how many fields. The game has a high branching factor (due to dices' throwing) and when played by masters becomes a highly complex battle, full of tactical and positional threats including sophisticated blocking strategies.

The evaluation function in Neurogammon was implemented by Multilayer Perceptron (MLP) neural network trained with backpropagation, having as the input the location of pieces on the board and a set of game features carefully designed by human experts. Board positions were extracted from the corpus of master-level games. Neurogammon achieved a steady human intermediate level of play which allowed it to convincingly win the computer olympic competition [102].

Quite a different approach was adopted in TD-Gammon - the successor of Neurogammon. TD-Gammon was also utilizing the MLP network, but it differed from Neurogammon in three key aspects: (1) instead of backpropagation training the temporal difference learning introduced by Sutton [100, 101, 49] was used; (2) the input to the network was a raw board state without any expert features[5]; (3) training was essentially based on self-playing as opposed to training based on board positions that occurred in games played by experts.

Initially, i.e. in a knowledge-free approach, TD-Gammon reached an intermediate human level of play roughly equivalent to Neurogammon's level. In subsequent experiments - still in a self-playing regime, but with the input layer extended by adding expert board features (the ones used in Neurogammon) to the raw board data - the level of play eventually became equivalent to the best world-class human players.

Following Tesauro's work, various attempts to repeat his successful TD approach in other game domains were undertaken, but none of the subsequent trials reached as high level of playing competency in any other game as TD-Gammon did in Backgammon. One of the possible reasons of TD-Gammon's striking efficiency is the stochastic nature of the game which allows broad search of the entire state space and a real-valued, smooth, continuous target evaluation function, as opposed to discrete and discontinuous functions in most of the popular perfect information, deterministic games. Another reason is ascribed to the impact of TD learning strategy on the course of neural net's training: first simple linear associations were learnt, and only then a representation of nonlinear, context-sensitive concepts and exceptional cases was built [105].

KnightCap. Another well known example of TD-type learning in games is Chess playing program KnightCap written by Baxter, Tridgell and Weaver [6, 7, 8]. The authors applied TDLeaf(λ) method - a variant of TD(λ) introduced in [10][6]. As opposed to Samuel [82], Tesauro [103], Thrun [106], Beal and Smith [10] and later on Schaeffer et al. [88] KnightCap's designers found self-playing to be a very poor way of learning and preferred the use of external trainers instead. Hence, the TDLeaf(λ) learning was carried out by playing on the Internet Chess site. The program started from the blitz rating of 1650 and required only three days of playing (308 games) to reach the blitz rating of 2150, which is roughly equivalent to master candidate player. Afterwards the rating curve entered a plateau.

[5] Some experiments with adding expert features to the input vector were carried out in subsequent studies.

[6] Although the idea of TDLeaf(λ) was first presented in [10], the algorithm's name was coined in Baxter et al.'s papers.

The success of KnightCap laid, according to the authors, in appropriate choice of TD learning parameters, and first of all in "intelligent material parameters" initialization, which reflected the common knowledge of the pieces' values in Chess. Additional contribution to rapid rating increase was attributed to the fact that the weights of all other (i.e. non-material) parameters were initially set to zero, and therefore even small changes in their values potentially caused a relatively significant increase in the quality of play.

The main lesson from KnighCap's experiment was that the choice of initial weights in the evaluation function is crucial for the speed and quality of training. Another conclusion concerned the choice of training opponents who, according to authors' suggestions, should be comparable in playing strength to the learning program. This observation is in line with common human intuition that too strong or too weak opponents are not as valuable as the ones playing on approximately the same level.

The weakest feature of KnightCap was playing in the opening phase[7]. One possible remedy to this problem is the idea of "permanent brain" introduced in Crafty [46] - the strongest publicly available freeware Chess program and a direct descendant of a former Computer Chess Champion - Cray Blitz [47]. "Permanent brain" stores a number of losing positions and their evaluations in a hash table, which is used in every search. Thus the program avoids playing into these unfavorable lines.

TDL-Chinook. Jonathan Schaeffer, the author of Chinook (the Man-Machine Checkers Champion described in sect. 2), together with Markian Hlynka and Vili Jussila applied TD learning to Checkers [88] in order to verify its efficacy in another (after Backgammon and Chess) demanding game. The authors used the TDLeaf(λ) learning scheme. Their direct goal was a comparison between Chinook's evaluation function and the TD-based learnt one. In order to make this comparison the TD learning player was initially equipped with Chinook's evaluation function but with a different set of weights assigned to its components. Two main approaches were considered: in the first one, Chinook served as the training opponent for the TD player whereas the second approach relied on self-playing. In the first case training was performed in a predefined regime involving the use of some number of standard Checkers openings (afterwards the game was continued and finally completed by the players).

Surprisingly enough it turned out that by applying the TDLeaf(λ) learning scheme the program was capable of reaching the level of play comparable to the teacher's even though Chinook evaluation function's weights had been carefully tuned for more than five years. More surprisingly, the other approach (self-playing) also led to the Chinook caliber program, which implies that

[7] Due to the way TD learning is performed the relatively poorer play in the openings is common to practically all TD implementations regardless of the choice of the game.

external teacher is not indispensable for achieving the human championship level of play in a complex game as Checkers is!

It is interesting that the weighting of features in the evaluation function of the learning program was very different from that of Chinook. Closer examination of weights developed during training revealed several interesting insights into how some "human-type" features (i.e. the ones which very rarely occur in machine vs machine play) are compensated by other components in the evaluation function.

NeuroChess. Another interesting application of CI methods in games is NeuroChess program written by Sebastian Thrun [106], which combines TD learning with Explanation-Based Neural Network learning (EBNN) described in [68, 107]. The evaluation function in NeuroChess is represented by a neural network which inputs and outputs are board features (defined by a human expert) of the current position and the one expected after the next two half-moves, respectively. The challenge of Thrun's approach is to learn the evaluation function (i.e. weights of a network) with TD algorithm based solely on the final outcomes of the training games. Training is also supported by self-playing. The role of EBNN is to speed up the training process by allowing better generalization. This goal is accomplished by defining a separate neural network called the *Chess model* which represents the domain knowledge, obtained based on a large number of grandmaster games.

Although NeuroChess never reached the level of play of GNU-Chess (being its test opponent) defeating it in about 13% of times, the experiment pointed out some important advantages and weaknesses of TD learning based on the final games' outcomes. First of all NeuroChess's ability of playing openings was very poor, which was the consequence of increasing inaccuracy of position estimation from the final position backwards to the opening one. Another characteristic feature of NeuroChess play was mixing very strong moves with schoolboy mistakes, which according to Thrun happened quite frequently.

The main conclusion from Thrun's work is that learning based solely on observation of grandmaster play (TD learning in here) is not efficient enough and may lead to several artifacts in agent's evaluation function. An example of such inefficiency it the tendency of NeuroChess (when trained without self-playing) to move its queen into the center of the board in the early stage of the game. This behavior was learnt from grandmasters' games, but the program was unable to observe that grandmasters make such moves only when the queen is safe from being harassed by the opponent. In other words the basic idea of EBNN, i.e. using domain knowledge for finding *explanations* for a given set of examples in order to generalize based on them is not sufficient in the game of chess since some moves cannot be fully explained based exclusively on the accessible domain theory, *ergo* cannot be properly learnt.

Anaconda (vel Blondie24). Kumar Chellapilla and David Fogel carried out an experiment in which an ensemble of feed-forward neural networks, each representing an evaluation function for the game of Checkers, was evolved in appropriately designed evolutionary process. The input data for each network consisted of locations of pieces on a game board. This data was further decomposed in the first hidden layer into all possible subsets of size 3×3, 4×4, $\ldots$, 8×8 of the entire board. The two subsequent hidden layers operated on features originated in the first hidden layer. A single output neuron represented the evaluation of a Checkers' position presented in the input.

In each generation offspring networks were created and then each network (being either parent or offspring) played against five randomly selected opponents from that population. The best networks constituted the population for the next generation. After 250 generations the top network was tested against human competitors on the Internet site where it achieved the rating of an A-class player (immediately below the expert level) [22, 23]. After another 590 evolutionary generations the best network achieved the rating of an expert player (just below the master level) according to the U.S. Chess Federation rating system on the same Internet gaming site [25, 33]. This network was also tested against three characters (Beatrice, Natasha, and Leopold) from the *Hoyle's Classis Games* - commercially available software - winning a six game match with the score 6 : 0 [24]. Chellapilla and Fogel used two names for their network: Anaconda and Blondie24. The former one was related to the system's style of playing (this issue is further discussed in the next section) while the latter - most probably - to attract other player's attention on the Internet gaming zone.

It should be underlined that except for the sum of all board inputs (reflecting difference in material), which was presented as an additional input value directly to the output neuron, no expert knowledge about the game of Checkers was incorporated into the neural networks or the evolutionary process. The only "knowledge" available during the process was the location of pieces on the board, the rules of making (generating) all legal moves in a given position and the minimax heuristic for selecting the most favorable move at a given search depth. In majority of the games the search depth was defined to be equal to 6 or 8. The search was extended further for non-quiescent positions.

The common feature of all playing agents described above is the ability to autonomously improve their playing strength basing on experience (games played). This improvement is achieved either in the self-playing regime or in the course of playing against external opponents. Interestingly, the above mentioned experiments and other works presented in the literature are inconclusive with regard to whether it is more profitably to favor self-playing or rather to train with external opponents. Hence, one of the interesting and challenging issues is further investigation and formalization of the strengths and weaknesses of both training approaches.

In the case of training with external opponents additional key issue is the choice of the opponent players and the scheme of training [6, 63]. According to intuition, too strong or too weak opponents may not lead to expected improvement since weak opponents play badly and the strong ones are too good to be followed by the learner. Also the training scheme, when playing against external opponents, may have a great impact on the speed and quality of the learning process. In particular, in TD learning one may consider updating weights of the evaluation function after each game or only after the games lost or drawn. Another possibility is to update the weights regardless of the game's outcome, but with elimination of weak moves which most probably may be misleading for the training process [6, 7]. One may also consider playing against stronger opponent a few times in a raw if only the learner keeps losing against that opponent. The results for the game of Give-Away Checkers presented in [75] suggest the superiority of such approach over classical TD learning based on either all games played or only the ones not won by the learner. Other constructions of the learning scheme, e.g. the tournament choice of the opponents, can also be considered. The issue of how to define the optimal training scheme deserves further investigation and hopefully new conclusions across various game domains will come into light.

Another relevant issue is the choice of initial weights in the evaluation function. Regardless of the training method (being either TD, neural nets or evolutionary approach) the choice of the starting point is in most cases crucial for the final outcome of the learning process. Usually these initial settings are based on human expert knowledge. Another possibility would be to define a universal, game-independent procedure allowing the development of "reasonable" initial settings that approximate the relative importance of particular features or their combinations.

Naturally, the problem of how to define the optimal set of features that compose the evaluation function for a particular game is also a challenge. A more demanding question would be how to define the human-guided, but semi-autonomous and *game-independent* process that would have led to the construction of suboptimal set of board (game) features. This issue was discussed by Paul Utgoff who stated in [111]: "*Constructing good features is a major development bottleneck in building a system that will make high quality decisions. We must continue to study how to enhance our ability to automate this process*". Utgoff suggested that game features should be overlapping and form a layered, hierarchical system in which more complex features are built based on simpler ones.

Another challenge concerns autonomous learning with zero initial knowledge (except for the rules of the game). The majority of game playing programs rely on carefully designed expert features reflecting positional and tactical nuances of the game. A good counterexample is Anaconda which does not rely on built-in human knowledge at all, and as such is an apparent, successful example of learning from scratch using Computational Intelligence techniques. Chellapilla and Fogel's success contradicts Allen Newell's opinion (supported also by Marvin Minsky): "*It is extremely doubtful whether there is enough information in 'win, lose, or draw' when referred to the whole play of the game to permit any learning at all over available time scales*" [67].

3.2 Creativity - Knowledge Discovery

One of the long-term goals of CI in game playing is development of **creativity** mechanisms which implemented in the playing program might lead to spontaneous **knowledge discovery**.

Some successful examples of such “emerging intelligent behavior” have already been presented in the literature however, according to the author’s knowledge, all of them were merely “the side effects” of the training process. The most famous example is probably Tesauro’s TD-Gammon described in the previous section, which according to former Backgammon world champion Robertie, came up with genuinely novel strategies that no one had used before. TD-Gammon’s play caused revision in human positional judgement in this game leading, for example, to invention of new opening moves - proposed by TD-Gammon and subsequently proved (in exhaustive, statistical analysis as well as tournament play) to be successful. Another interesting observation concerning TD-Gammon is the development of spatial weight patterns in the MLP, responsible for representation of particular game concepts, which *were not explicitly presented in the course of training* [103].

Similar observations about *ad-hoc* feature discovery and feature representation in neural network weights were reported in [70, 62, 71] concerning the game of Bridge. The authors considered the so-called Double-Dummy Bridge Problem, which consists in answering the question about the number of tricks to be taken by a pair of players assuming perfect play of all four hands with all cards being revealed.

Several MLP networks with $0, 1$ or 2 hidden layers were trained in a supervised manner and tested based on the data from the GIB Library [37], created by Ginsberg using his GIB program [39][8]. The input layer was composed of 52 neurons and each of them was assigned to a particular card from a deal. The value of this neuron denoted the hand containing this card (e.g. $N : 1.0$, $S : 0.8$, $W : -1.0$, $E : -0.8$). A single output neuron yielded the predicted number of tricks to be taken (the output range - $[0.1, 0.9]$ was divided into 14 intervals of equal length). Besides deal assignment, no additional information e.g. the rules of the game or the strength of particular cards was provided to the network. Except for achieving satisfying numerical results the other main goal of that research was exploration of networks’ knowledge representation and search for patterns in the weight space that possibly represented particular “Bridge features” (e.g. the relative strength of cards). Examining the weights in the trained networks revealed several interesting observations.

Firstly, weights of outgoing connections from input neurons representing *aces* and *kings* always had the biggest absolute values. This feature was simple to explain (for humans) - these cards are the most important in the game of Bridge, especially in no trump contracts.

Secondly, in each trained network there were exactly four connections from input to hidden neurons with weights’ absolute values noticeably bigger than

[8] GIB is considered one of the top machine Bridge players.

all the others (about 25.0 vs less than 7.0). Not surprisingly these favored connections started from four input neurons assigned to *aces*.

Thirdly, in all networks it was possible to point out four hidden neurons focused on particular suits (one neuron per suit). Absolute values of connection weights from inputs representing the respective suit to such hidden neuron were much bigger than absolute weight values from the remaining inputs.

Finally, a very interesting feature which appeared in all trained networks with sufficient number of hidden neurons, was the presence of four hidden neurons, each of which focused on five top cards from one particular suit: *ten*, *jack*, *queen*, *king* and *ace*. In each of these five-card groups the most important connections were from *queens* and *kings*, *jacks* were less important, but still much more relevant than *aces* and *tens*. The hypothesis is that these hidden neurons were responsible for a very important aspect of the game - *the finesses*.

All the above observations are in line with human knowledge about the game of Bridge. Estimation of the strength of individual cards as well as entire suits is the basic information considered in the process of a hand's evaluation. Even though these game features are trivial to understand for human players they are not necessarily easy to discover by a neural network. Moreover, quite surprisingly, in the simple training process, the networks were also able to independently discover the notion of *the finesses*, which is a subtle mechanism - not rarely deciding about the final number of tricks taken by a playing pair.

Interesting observations concerning knowledge discovery in the game of Chess were reported in the MORPH experiment [56, 42] which implemented pattern based learning with the weights of patterns being modified through the TD(λ) method combined with *simulated annealing*. Although the strength of MORPH was far inferior to GNU Chess, the patterns learned by the system were consistent with human Chess knowledge. In particular MORPH was able to play openings on a reasonable level, despite the fact that no information about the significance of development or controlling the center of the board in the opening phase had been added to the system. On the other hand, one of the weaknesses of MORPH was poor scalability with respect to the number of patterns, due to the lack of efficient selection mechanisms. Nevertheless, the system was able to defeat human novices while searching only 1-ply.

A general approach to automatic feature generation was presented by Fawcett and Utgoff [32]. Given only domain theory and the ability to solve problems in this domain the system called Zenith was able to automatically generate a set of relevant domain features. The system started from a single feature created automatically from the problem's goal (e.g. "win of white") and by using four predefined types of transformations: decomposition, abstraction, regression and specialization, gradually extended the set of features in an iterative manner. Zenith was applied to Othello with promising results. For example, the system autonomously discovered the importance of stable pieces, i.e. the ones which cannot be reversed [32].

Another well-known example of independent feature discovery in games is Anaconda [25, 33] described in sect. 3.1, which received its name due to the "snake-like" way of playing - in most of the games won by the program its opponent was blocked and therefore forced to make a weak move. However, neither in the input data nor in the evolutionary process of Anaconda's development the concept of mobility was ever explicitly considered. Hence the importance of mobility must have been "invented" by the system or more precisely by the evolutionary process, which guided Anaconda's development.

The potential strength of neuro-evolutionary approach was also reported in Othello [69]. The evolved networks "discovered" positional features and advanced mobility issues indispensable for high-profile tournament play.

All the above examples led to discovering new features, previously unknown to the system, induced from the training data. The ultimate goal that can be put forward in this context is *autonomous* discovering of *all* relevant components of the evaluation function in a way allowing their *separation* and *explanation*. Such a requirement goes beyond Anaconda experiment and other neural or neuro-evolutionary type approaches that resulted in efficient *numerical approximation* of the board state, but lack the *feature-based formulation* of the evaluation function.

3.3 Intuition

Implementation of the concept of intuition is definitely one of the greatest challenges in computer games and also in computer science in general. Nowadays, despite the major breakthroughs made in several disciplines and despite increasingly deeper, scientific understanding of the nature, intuition - paradoxically - becomes more important than ever.

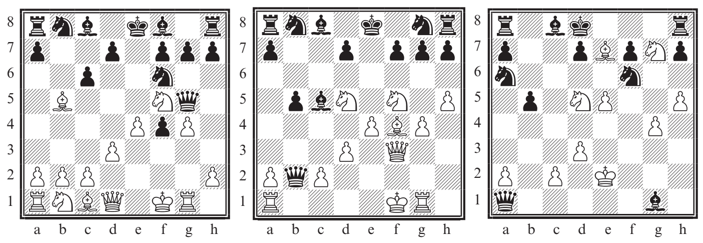
One of the most salient research studies focused on understanding (and implementation of) intuition was performed by Herbert Simon - a Nobel Prize Winner in Economics. According to Simon intuition is nothing mysterious or extraordinary and simply relates to a subconscious pattern recognition process able to immediately provide appropriate pattern(s) among those stored in the memory, based on our knowledge and experience. According to Simon, this does not mean that intuition is an irrational process - he considered it to be a rational but neither conscious nor analytical one [95].

Simon was optimistic about the potential abilities of "thinking machines" and predicted that any "intelligent" human activity (thinking, creativity, decision making, intuition and other) will ultimately be implemented in artificial systems.

In most of mind board games intuition plays a leading role at master level of play. Consider for example Chess. With a branching factor of about 30, in a 50 move (100 ply) game there are about 10^{147} contingencies, which is an enormous number for any human being (grandmasters are believed to search no more than a few hundred contingencies during the entire game). How then it is possible that Chess champions are able to play at such a high level? One of the factors is intuition which allows them to perform highly selective search in this huge space, *although in many cases they are not able* *to explain why they have chosen to search a particular contingency and skipped the others.* Moreover, when playing simultaneous games, Chess grandmasters usually need only a few seconds to make a move, which generally proves to be very strong (often optimal). This means that they have the ability to immediately find the most relevant information characterizing board position and recognize the most promising continuation (move), usually *without deep, precise calculation of its contingencies.*

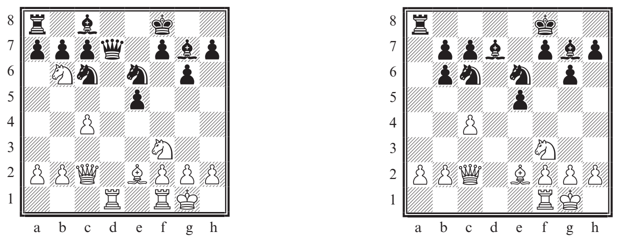
Another aspect of intuition in board games is the ability to almost instantaneous recognition of strengths and weaknesses of a given position. A grandmaster usually needs only a few seconds of board analysis in order to tell which side is in the winning or favorable position. One of the possible psychological explanations of this phenomenon is the ability of advanced players to link the new position with previously explored familiar ones and consequently to focus on moves and plans associated with these, already known, positions [27] (this topic is further discussed in sect. 3.4).

Based on the above described results of applying intuition in games, one can provide the operational definition of intuition as an instantaneous, subconscious recognition/reasoning process which does not rely on precise, deep calculations, but instead rather refers to past experiences and previously acquired general knowledge. Consequently, in most mind games, intuition is one of the main factors contributing to the beauty and attraction of the game. Its application often leads, for example, to long term material sacrifices without apparent possibility of its recovery. A well known example in Chess is *the immortal game* played in London in 1851 by Adolf Anderssen and Lionel Kieseritzky in which white sacrificed bishop (on move 11 - see Fig. 1(a)) and subsequently two rooks and a queen (starting on move 18 - see Fig. 1(b)) in order to checkmate on move 23 - (Fig. 1(c)). Certainly, the last three sacrifices were tactical ones, i.e. their consequences could have been precisely calculated by Anderssen, but the introductory sacrifice (bishop on move 11) is an example of an intuitive type of move based on players experience and his "feeling" of the board position.



(a) After 11. Rg1 ... (b) After 17. ... Qxb2 (c) 23. Be7++. White won

Fig. 1. Anderssen vs Kieseritzky, *Immortal game*, London, 1851



(a) After 16. Nb6 ...

(b) After 16. ... axb6, 17. Rxd7 Bxd7

Fig. 2. Karpov vs Kasparov, New York, 1990

Another interesting example of intuitive sacrifice occurred in the game played between two great archenemies: Anatoly Karpov and Garry Kasparov in the New York match in 1990. In the middle-game position Kasparov sacrificed queen for a rook and knight on moves 16 − 17 (see Fig. 2) and this sacrifice was clearly positional with no immediate tactical or material threats. The game continued up to 53th move, when players agreed for a draw.

Theoretically, human-type intuition in machine playing may possibly emerge as a "side effect" of using a close to optimal evaluation function (on condition that such a function could be practically specified and implemented). Examples of "intuition" of such origin have been observed in the famous Kasparov vs Deep Blue re-match, in which some of the machine's moves were described by grandmasters commentating on the match as *phenomenal* and *extremely human.*

One of very few published attempts focusing on formalization of intuitive concepts in Chess was recently described by Arbiser [3]. The author proposes the way of formalizing such concepts as capture, attack, threat, sacrifice, etc. as well as the notion of *style of opponent's play*, i.e. aggressive, defensive, conservative, tactical or positional. The underlying idea is based on generalization of the null-move heuristic in such a way that instead of hypothetical opponent's moving twice in a row, the opponent is allowed to virtually change one of his or our pieces or add/delete a piece and then make a move. For example the notion of aggressive play will be implemented by exchanging one of the opponent's or our pieces into a strong opponent's piece before deciding a move. Such an exchange would most probably cause immediate threats to us thus forcing the choice of an appropriate response. In short, the following scheme is proposed: modify the board in an adequate manner before calling a regular search algorithm and ensure that the chosen move would be valid and sound in the original board position i.e. the one without initial, fictitious modification. Although the description of the method raises several questions concerning its time complexity as well as the omitted implementation details, overall the algorithm seems to be a step in the right direction.

Understanding and furthermore implementation of the mechanism of intuition in artificial players is one of the main challenges for CI in games. Several issues described in the remainder of this chapter, e.g. geometrical trajectories, positional generalization, feature abstraction may partly compliment to the implementation of intuition, but the efficient and general approach to this wonderful human ability is yet to be specified. I would argue that unless programs (machines) capable of making intuitive moves (in the above described sense) in Chess and other mind board games are created, we should be very cautious about announcing the end of the human era in these games.

In 1931 Albert Einstein wrote [28]: "*The intuitive mind is a sacred gift and the rational mind is a faithful servant. We have created a society that honors the servant and has forgotten the gift*". This aphorism, originally related to religion, can also be referred to other human activities including the domain of machine game playing development.

3.4 Abstraction and Generalization

As discussed in the previous section, one of the facets of human game playing is the ability to abstract particularly relevant game features from a given board position. This skill allows experienced players almost immediate estimation of positional and tactical strengths and weaknesses on both sides as well as to point out future possibilities and potentially promising moves. For example in Chess these crucial features include *pawn structure*, *cooperation of figures* (e.g. two rooks on the 2nd (resp. 7th) line or multiple attack on point F2 (F7 resp.)), *mobility*, *tempo* and many more.

In practically all popular mind board games vital positional and tactical features are context-sensitive. Due to the presence of other pieces on the board their appropriate classification is not a straightforward task for machine players and requires both abstraction and generalization capabilities.

Another generalization task is an attempt to reason on the quality of a move based on shallow search. On one hand it is hard not to agree with Schaeffer, Hlynka and Jussila who stated in [88]: "*There is no free lunch; you can't use shallow search results to approximate deep results*" and therefore advised: "*the weights [of an evaluation function] must be trained using depths of search expected to be seen in practice*".

On the other hand the above claims are not necessarily valid when estimating *the relative* strength of the moves without focusing of their true numerical evaluation. A crude relative estimation of possible moves is crucial for the efficacy of several search algorithms (e.g. the alpha-beta based ones). This issue is further discussed in the next section.

A challenging test of generalization skills applicable to machines is solving game problems defined on arbitrarily large game boards. Intelligent approach to such problems requires efficient generalization from shallow search results. John McCarthy in 1998 in his comments to intelligent Chess problem solving, referring to the famous *Reti problem* (Fig. 3), stated: "*Note that Reti's idea can*

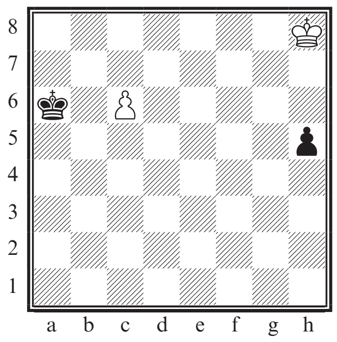


Fig. 3. Reti ending. White to begin and draw.

be implemented on a 100×100 *board, and humans will still solve the problem, but present programs will not ... AI will not advance to human level if AI researchers remain satisfied with brute force as a substitute for intelligence ... Would anyone seriously argue that it is impossible for a computer to solve the Reti problem by other than brute force?*" [66]. An interesting approach to this type of generalization is expressed by the Linguistic Geometry (LG) which focuses on evaluation of *trajectories of possible solutions* rather than on exact exploration of the game tree [98]. Instead of traditional search-based approach, LG proposes methods for construction of problem solving strategies. These strategies can be represented as trajectories on the board (e.g. in the endgame problems) and to some extent allow formalization of expert knowledge and intuition (see [98] for details).

Another challenge related to abstraction and generalization is the quest for learning methods capable of generalizing knowledge across game boards of different sizes. One particularly interesting question is: how to apply the outcomes of learning on small boards to the learning process performed on larger boards? One possible approach is to use *incremental training* methods implemented in neural networks according to the following procedure. Learning starts off in the environment (game board) smaller than the target one. During the training process the environment is gradually increased - up to desired size - and after each change of size the limited number of training examples is modified/added in order to capture new features that arose in this larger, more complicated environment. The claim is that after some training time, the system should be able to recognize features, which are *invariant to the size (degree of complication) of the environment.* In such cases these features will be shared among several instances of the environment and during the training process used to make generalizations about the learning task.

Consider, for example, a game which is played on a board of size n. In the proposed approach the training process begins on a game board of smaller size $k, k < n$ and after the agent learns how to play or solve problems on this board, the board size is increased to $k + t$, where t depends on particular game ($t \in \{1, 2\}$ for the majority of popular games). Then the agent is re-trained *in a limited manner* based on the new set of problems presented on the increased board. The re-training procedure is significantly shorter than the regular training performed on the board of size k. Once again the board size is increased and the agent is re-trained, etc. The whole procedure is stopped after the re-training on board of size n is completed.

The underlying idea is that after the preliminary phase, learning should become relatively easier, and solutions for problems defined on larger boards would be developed by the system based on already defined solutions for problems stated on smaller-size boards. Hence, *subsequent learning would mostly involve efficient use of previously acquired knowledge.*

The above described learning scheme is problem independent, but can be applied only to a certain type of games such as Checkers, Othello or Go, which can be easily defined on boards of different sizes. Such training scheme was used in Othello on 6×6, 8×8 and 10×10 boards (the board of size 10×10 being the target one) [59, 60]. The MLP neural network was fully trained on examples from 6×6 board and subsequently re-trained, in a limited manner, on 8×8 and 10×10 boards' examples. The training goal was to point out the best move in a given position. The results of the above-described incremental training procedure were compared with the full backpropagation training carried out exclusively on 10×10 board examples. The amount of time required for incremental training was considerably lower than in the opposite case. Also numerical results showed a slight improvement over a one-shot training procedure: after incremental training the network responded with the best move (selected according to applied heuristic) in 40% of the cases, compared to 34% achieved in the full backpropagation training on 10×10 board [60].

3.5 Pre-Ordering of Moves

In practical applications, efficient moves pre-ordering should rely on shallow search or no search at all, otherwise, the remaining time devoted to deeper, selective search may be insufficient. Efficacious pre-ordering of moves is again a "very human" skill. Human Chess players, for example, can estimate roughly 2 positions per second - compared to 200 billion ones checked in a second by Deep Blue - and therefore must be extremely effective in preliminary selection of moves.

There exist a few popular, search-free strategies of moves pre-ordering basing on historical goodness of the move in previous games played. These include the *history heuristics*, *transposition tables* or the *killer move* heuristics. In most cases these methods are highly effective since the assumption that a move which often appeared to be efficient in the past is more likely to be suitable for the current game position than other "less popular" moves is generally correct (see e.g. [83] for further discussion and experimental results in Checkers).

An interesting approach to moves pre-ordering in Chess was presented by Greer [43]. The method relies on pattern-oriented classification of moves based on heuristically defined *influence* of a particular move on certain board regions. At first, each square is assigned a label that represents heuristical belief in which of the two players controls this square. Combining this information for all squares leads to the chessmap which represents the regions of the board that are in favor for each side as well as the neutral areas, where none of the players has a visible advantage[9]. Additionally, for each square (or more generally each sector composed of some number of squares) the so-called valueboard is defined based on the relative strength of the control that one side has over that square (sector). The influence of a move on a given square (or sector) is defined as the sign of a difference between the valueboard after that move would have been made and before (i.e. in a current position). This allows to detect the squares that would be strengthened by that move as well as the ones that would be weakened.

In order to learn the influence relationship for Chess positions a neural network was trained based on $10,000$ positions extracted from master and grandmaster games. The input layer represented influence labels of 64 squares and the kings' locations. The desired output values in the 64 element output layer (one neuron per square) were the influence labels after a move had been made in the actual game. After training, the outputs of the network were used to order board squares according to their predicted influence values. Consequently moves that influenced the highest ranked sector(s) of the board were considered as the most promising ones.

The above procedure was further enhanced by giving priority to forced and capture moves. Several tests of this interesting pattern-based heuristic were carried out including the ones on the set of 24 Bratko-Kopec positions [51], for which the quality of the method - calculated as the number of searched nodes - was comparable to the result of applying the history heuristic.

Pattern-based pre-ordering of moves is in line with psychological observations of how human grandmasters make decisions about which moves to consider first. As Johannes Fürnkranz stated in his excellent review of a decade of research in AI and computer chess [35] referring to the work of deGroot [27] "*the differences in playing strengths between experts and novices are not so much due to differences in the ability to calculate long moves sequences, but to which moves they start to calculate. For this preselection of moves chess players make use of chess patterns and accompanying promising moves and plans*".

Pattern-based approaches seem to be perfectly suited for Go, which is a territory game. Since today's Go programs are far from being a threat to human players and rely only on simple pattern matching [73] application

[9] The idea of calculating the *influence* of white and black pieces in order to divide the board into sections controlled by the respective players was initially introduced by Zorbist [114] in Go.

of pattern-oriented methods of moves pre-selection in this game may be a promising research direction.

3.6 Opponent Modeling

Modeling the opponent is another fundamental issue in current AI/CI research. The problem actually extends far beyond the game playing domain and is considered as a crucial aspect of any competitive multi-agent environment (e.g. decision support systems, stock markets, trading systems, etc.). In game domain the relevance of opponent modeling strongly depends on the choice of a game. Relatively lesser impact on the quality of playing programs concerns perfect information games, such as Chess, Checkers, Go, Othello, etc. However, also in these games the problem is not negligible. The style of play (tactical vs positional, aggressive vs conservative, etc.), if properly modeled, can provide an important indication for a game playing program. For example, in a disadvantageous position a program could use a specific style of play in order to hinder the potential victory of the opponent and strive to achieve a draw. Another example is seeking the chance to win an even game by steering it to inconvenient (for the opponent) positions and thus provoking an opponent's mistake.

A similar situation is observed among human players. Nearly each of the top players in any popular board game has opponents who are "less convenient for him", i.e. achieve relatively better results against that player than is indicated by their ranking (e.g. ELO in Chess). In other words the "winning relation" is non-transitive and the ranking points provide only general, statistical information about player's strength.

Modeling the opponent is far more important in imperfect information games, especially the ones, in which deception (bluffing) is an inherent part of the rules. A simple, though instructive, example is the kids' game Rock-Paper-Scissors, also known as RoShamBo [12]. In this game, each of the players independently and simultaneously chooses among Rock, Paper and Scissors. In the simplest case of two players the winner is the one whose choice "beats" the opponent's choice under the following rules: Rock beats Scissors, Scissors beat Paper and Paper beats Rock. If both players point out the same object the turn ends with a draw. Even though the rules of the game are trivial, the game itself is more demanding that one might expect at first glance. A simple solution is of course choosing actions randomly according to uniform distribution. Such approach would statistically lead to a draw, however it does not take into account the opponent's playing policy which may possibly be inferred from his/her previous play. Moreover, except for trying to predict the opponent's next move, a skilful player (program) should avoid to be predictable itself. Hence, simple rule-based approaches are not sufficient and more sophisticated methods are to be employed.

Another example of the game in which opponent modeling is crucial for efficient playing is a dice game Perudo [57] also known as Liar's Dice. The rules of Perudo are not very complicated [58], though not as simple as those of RoShamBo. Playing the game well requires quite sophisticated analysis of opponents' past actions in order to detect possible bluffing, since the game significantly consists in bluffing and straightforward playing most probably would not lead to success against experienced opponents.

Certainly the most popular "game of deception" is Poker. Here the notion of (objectively) optimal playing is hard to define due to the huge amount of uncertainty regarding hidden opponents' cards (the *hole cards*) and the latent *community cards*[10]. Hence the optimal behavior can only be estimated with some probability and its calculation strongly depends on the opponents' actions. A simple example given in [13] concerns the frequency of bluffing by the opponent. The one who bluffs more frequently should be called more often compared to the one who bluffs relatively rarely. One possibility of modeling opponent's behavior is to construct a statistical model for the next move prediction based on sufficiently large number of games already played against that opponent. Another possibility is to train a neural network to predict the opponent's next action. Poki-X team (cf. sect. 2) employed a standard, one-hidden-layer MLP network with 19 inputs representing particular aspects of the game and three outputs corresponding to three possible opponent's decisions (*fold*, *raise* or *call*). The outcome of the network provided a probability distribution (after output normalization) of the opponent's action in a given context defined by the input features. After training on deals played by a given opponent, magnitudes of network's weights reflected the relative impact of particular input features on the predicted outcome, which allowed further exploration of the input data, e.g. finding new features and defining a relatively small number of context equivalence classes in the input space.

Additional advantage of using neural nets for the opponent modeling task is their ability to adapt to changes observed in the training patterns (representing the opponent's behavior) by adequate weights' tuning[11].

Computational Intelligence methods are very well suited to the problem of opponent modeling. Probabilistic methods allow building a generic opponent's model for a given game, which can be further optimized, e.g. with the use of genetic algorithms. An alternative approach, especially in the case when the opponent's patterns of activity vary in time, is to use neural networks, due to their capability of adapting internal parameters to the gradually changing input training data. Another promising avenue is to use TD learning and adapt internal parameters of the playing system according to achieved results.

Definitely, on-line, adaptable and close to reality modeling of the opponent is one of the fundamental challenging problems in any non-trivial game,

[10] We refer to the Texas Hold'em variant of Poker, which is now the most popular version of this game. The rules of the game can be found for example in the excellent books by David Sklansky [96, 97]

[11] Certainly such network weights' adjustment requires the use of appropriate training scheme and is possible on condition that changes in the opponent's behavior are relatively smooth.

in particular in imperfect information games, in which data hidden by the opponent can only be inferred by analysis of his/her past actions in similar game situations. Proper prediction of this hidden information increases the expected outcome by decreasing the amount of uncertainty.

It is worth to note that the problem of opponent modeling becomes much more demanding when multi-player situation is considered, in which players can form ad-hoc coalitions (formal or informal) against the current leader or in order to gain some benefits. In such a case players' decisions are highly contextual and strongly depend on short-term and long-term goals of the coalitions they belong to.

3.7 Universality of Tools

One of the grand challenges in game playing is associated with designing general-purpose methods and algorithms that abstract from particular games. CI is well located in this stream and several CI-based attempts to create game-independent methods were presented in the literature.

Game-Independent Learning

Research concerning game-independent learning aims at developing systems capable of learning any game belonging to a certain class of games. This topic was very popular in the mid 1990's when some renowned methods and systems originated.

One of the well known universal learning systems is Michael Gherrity's SAL program [36] capable of learning any two-player, perfect information, deterministic game. SAL consists of a kernel that uses TD learning combined with neural network's backprop learning in order to learn the evaluation function for a given game. The kernel is game-independent and remains unchanged for different games. The rules of making valid moves for any particular game are represented by the game-specific module. SAL learns by trial and error from the games it has played hitherto. It generates two evaluation functions, one for each playing side which allows learning non-symmetric games or imposing asymmetry in symmetric games, if necessary. The system uses only 2-ply search of the game tree. SAL's success is strongly hindered by slow learning. For example, it took the program $20,000$ games to learn to play the Tic-Tac-Toe game.

Another interesting approach to game-independent learning is represented by Susan Epstein's HOYLE system [30, 31], able to learn two-person, deterministic, perfect information games. The underlying idea of HOYLE is to use a set of game independent advisors each specializing in a narrow, specific aspect of game-playing (e.g. one advisor may focus on material advantage while another one on finding the winning moves or sequences of moves, etc.). Each of the advisors may recommend some moves and all of them can comment on these proposals from their specialized viewpoint. Finally the advisors vote using a simple arithmetic voting system. Similarly to SAL, HOYLE uses only shallow search (2-ply ahead at most).

The diversity of advisors plays a crucial role in learning a new game. Each of the advisors learns patterns from played games chosen according to its individual priorities. One advisor may be focused on patterns related to the opening moves while another, for example, on those related to strong, winning moves, etc. Besides game-specific knowledge that can be acquired by the advisors based on analysis of the played games, HOYLE is *a priori* equipped with some general knowledge about the domain of two-person, deterministic games.

The efficacy of HOYLE has been demonstrated by playing Tic-Tac-Toe and Nine-Men's Morris. The potential of Epstein's approach in more complicated games (Chess, Checkers, ...) has not been experimentally proven.

MORPH II developed by Robert Levinson [55] is another example of game-independent learning system and also a problem solver. MORPH II is a direct extension of MORPH - a Chess learning program mentioned in sect. 3.2. MORPH II uses several CI learning techniques which include neural network-like weights propagation and genetic algorithm-type pattern evolution. Additionally, MORPH II implements symbolic learning. The system is capable of autonomous abstraction of new features and patterns and development of its own learning modules. Like its predecessor, MORPH II relies on shallow search equal to only 2 plies on average. The system has successfully learnt to play Chess on a novice level. Its strength against more demanding opponents hasn't been demonstrated.

All the above-mentioned general learning systems are potentially capable of picking up any game within a certain class of games. They use CI techniques combined with AI symbolic learning and multi-agent approach. They all rely on shallow search, just 1 or 2 plies. All of them have demonstrated the ability to efficiently learn how to play very simple games or more complicated ones, but at a novice level only. Definitely this direction deserves further exploration and the issue of how to design universal, game-independent learning systems is one of the grand challenges for CI community in the area of intelligent game playing.

Multitask Learning

The idea of designing game-independent learning systems can also be realized within the multitask or incremental learning schemes. Multitask learning utilizes simultaneous learning of a few tasks and sharing the representation issues, experience and knowledge among all of them in order to make the overall learning process faster and more effective. Incremental, lifelong learning is usually implemented as a sequential learning process, i.e. tasks are being learnt one after another, but again representation of problems as well as knowledge acquired in previous learning are widely shared and involved in subsequent learning, consequently making the latter easier.

Several approaches to multitask and lifelong learning (not only within game playing domain) has been developed in the last 10 years (e.g. [108, 107, 21, 64]), but there is still a strong demand for new concepts in this area. In case of game playing one may think of using the experience gained in learning one or more games in order to alleviate the effort of learning another, similar game. This type of learning is typical for humans. For example, previous experience in playing cards is one of the fundamental factors in efficient learning of a new card game.

Generally speaking, in case of similar games some representation issues and high level game rules are either common or very similar. Therefore, when learning a new game there is no need to start from the very beginning. The learning system may exploit already possessed knowledge - albeit usually its appropriate tuning will be required.

4 Conclusions

Mind games provide cheap, replicable environments, perfectly suited for testing new Computational Intelligence learning methods and search algorithms. They also serve as an excellent framework for testing and validating various implementations of human-type cognitive skills, e.g. intuitive behavior, creativity, knowledge discovery, or unguided, autonomous and context-sensitive learning.

All the above skills are crucial in achieving the long-term goal of CI in mind game playing research, which is the ability to mimic human methods of learning, reasoning and decision making. Several challenging problems have to be addressed on this path. Some of them are proposed and motivated in this chapter.

One of the most interesting issues is implementation of mechanisms of autonomous knowledge discovery that would lead to creation of new game features and new playing strategies. In particular a very challenging task is autonomous choice of board features that compose efficient (close to optimal), descriptive game representation allowing adequate evaluation of board positions. At the moment the development of a world-class playing program requires that the set of features be predefined by human experts. Even though there exist a few notable examples of learning how to play certain games without human expertise, there is still a lot of work ahead.

Another fundamental issue is the ability to improve artificial player's behavior through the learning process resting solely on the experience-based knowledge acquired from previously played games. An interesting, open problem in this area is analysis of pros and cons of two main learning schemes used so-far, i.e. playing against external opponents vs self-playing. In the former case, additional open problems concern the optimal choice of the training opponents and the training schedule. Both types of learning were successfully applied to various board games especially with TD learning algorithms. Further exploration of these two directions may ultimtely lead to the development of a general purpose learning engine (system) capable of learning any board mind game.

Another formidable challenge is implementation of intuition in the game-playing systems or, more precisely, implementation of mechanisms that would efficiently pretend human-type intuitive behavior. Such achievement would straightforwardly lead to the efficacious search-free pre-selection of moves and instantaneous estimation of position strength as well as the ability to play strong positional moves relying on shallow search only. All three above mentioned skills are typical for experienced human players, but still generally non-attainable for machines.

Yet another challenging issue concerns game independent learning, in particular incremental learning methods allowing for sequential or simultaneous learning of a few games. Sequential incremental game learning may rely on appropriate tuning of already possessed knowledge and generation of new features only when necessary, i.e. when they are "sufficiently different" from already discovered ones. Simultaneous learning requires that representational and computational issues be shared on-line among various learning tasks (games) and each learning task benefits from this synergy.

Achieving the above challenging goals does not necessarily mean construction of an omnipotent, unbeatable program/machine capable to play "in God's way". On the contrary, as humans make mistakes and are not infallible, also the CI-based playing systems may possibly suffer from "human" weaknesses, though to a much lesser extent.

In summary, CI-based methods focus on the way the game playing issues are implemented and solved rather than on the quality of play, which is regarded as relevant, yet subservient, supplementary goal. This distinguishes CI-based game-learning systems from traditional AI-based approaches focused mainly on maximization of the level of play. Consequently, in the near future AI playing systems are with high probability hardly to be defeated by CI-based ones.

The above conclusion, on the other hand, does not mean that application of CI methods to mind game playing is not interesting or not advantageous. On the contrary, I would argue that the mind game playing research will more and more be tied with Computational Intelligence methods and its future will be closely related to the development of psychologically motivated learning processes attempting to follow higher-level human competencies.

Acknowledgement

The author is grateful for the support from the Warsaw University of Technology, grant no. 504G 1120 0008 000.

References

[1] I. Aleksander. Neural networks - evolutionary checkers. *Nature*, 402(6764):857, 1999.

[2] T. Anantharaman and M. Campbell. Singular extensions: Adding selectivity to brute-force searching. *Artificial Intelligence*, 43:99–109, 1990.

[3] A. Arbiser. Towards the unification of intuitive and formal game concepts with applications to computer chess. In *Proceedings of the Digital Games Research Conference 2005 (DIGRA'2005)*, Vancouver, B.C., Canada, 2005.

[4] L. Barone and L. While. An adaptive learning model for simplified poker using evolutionary algorithms. In *Proceedings of the Congress of Evolutionary Computation (GECCO-1999)*, pages 153–160, 1999.

[5] L. Barone and L. While. Adaptive learning for poker. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 566–573, 2000.

[6] J. Baxter, A. Tridgell, and L. Weaver. Experiments in parameter learning using temporal differences. *ICCA Journal*, 21(2):84–99, 1998.

[7] J. Baxter, A. Tridgell, and L. Weaver. Knightcap: A chess program that learns by combining td(λ) with game-tree search. In *Machine Learning, Proceedings of the Fifteenth International Conference (ICML '98)*, pages 28–36, Madison Wisconsin, July 1998.

[8] J. Baxter, A. Tridgell, and L. Weaver. Learning to play chess using temporal differences. *Machine Learning*, 40(3):243–263, 2000.

[9] D. Beal. A generalised quiescence search algorithm. *Artificial Intelligence*, 43:85–98, 1990.

[10] D. F. Beal and M. C. Smith. Learning piece values using temporal differences. *ICCA Journal*, 20(3):147–151, 1997.

[11] H. Berliner. The B* tree search algorithm: A best-first proof procedure. *Artificial Intelligence*, 12(1):23–40, 1979.

[12] D. Billings. Thoughts on RoShamBo. *ICGA Journal*, 23(1):3–8, 2000.

[13] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134:201–240, 2002.

[14] B. Bouzy and T. Cazenave. Computer Go: an AI oriented survey. *Artificial Intelligence*, 132(1):39–103, 2001.

[15] D. Bump. GNU Go. http://www.gnu.org/software/gnugo/gnugo.html, 1999.

[16] M. Buro. Probcut: An effective selective extension of the alpha-beta algorithm. *ICCA Journal*, 18(2):71–76, 1995.

[17] M. Buro. From simple features to sophisticated evaluation functions. In H. J. van den Herik and H. Iida, editors, *Proceedings of Computers and Games Conference (CG98)*, volume 1558 of *Lecture Notes in Computer Science*, pages 126–145, Springer, Berlin, 1999.

[18] M. Buro. Toward opening book learning. *ICCA Journal*, 22(2):98–102, 1999.

[19] M. Buro. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence*, 134:85–99, 2002.

[20] M. Campbell, A. J. Hoane Jr., and F.-h. Hsu. Deep Blue. *Artificial Intelligence*, 134:57–83, 2002.

[21] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[22] K. Chellapilla and D. B. Fogel. Evolution, neural networks, games, and intelligence. *Proceedings of the IEEE*, 87(9):1471–1496, 1999.

[23] K. Chellapilla and D. B. Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391, 1999.

[24] K. Chellapilla and D. B. Fogel. Anaconda defeats Hoyle 6-0: A case study competing an evolved checkers program against commercially available software. In *Congress on Evolutionary Computation, La Jolla, CA, USA*, pages 857–863, 2000.

[25] K. Chellapilla and D. B. Fogel. Evolving a neural network to play checkers without human expertise. In N. Baba and L. C. Jain, editors, *Computational Intelligence in Games*, volume 62, pages 39–56. Springer Verlag, Berlin, 2001.

[26] P. Darwen and X. Yao. On evolving robust strategies for iterated prisoner's dilemma. volume 956 of *LNCS*, pages 276–292. Springer, 1995.

[27] A. D. de Groot. *Thought and Choice in Chess*. Mouton Publishers, The Hague, 1965.

[28] A. Einstein. *Cosmic Religion, with Other Opinions and Aphorisms*. 1931.

[29] M. Enzenberger. Evaluation in Go by a neural network using soft segmentation. In *Advances in Computer Games: Many Games, Many Challenges: Proceedings of the International Conference on Advances in Computer Games (ACG-10)*, pages 97–108, Graz, Austria, 2003.

[30] S. Epstein. Identifying the right reasons: Learning to filter decision makers. In R. Greiner and D. Subramanian, editors, *Proceedings of the AAAI 1994 Fall Symposium on Relevance*, pages 68–71, New Orleans, 1994. AAAI Press.

[31] S. L. Epstein, J. Gelfand, and J. Lesniak. Pattern-based learning and spatially-oriented concept formation in a multi-agent, decision-making expert. *Computational Intelligence*, 12(1):199–221, 1996.

[32] T. E. Fawcett and P. E. Utgoff. Automatic feature generation for problem solving systems. In D. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Conference on Machine Learning*, pages 144–153. Morgan Kaufmann, 1992.

[33] D. B. Fogel. *Blondie24: Playing at the Edge of Artificial Intelligence*. Morgan Kaufmann, 2001.

[34] D. B. Fogel, T. J. Hays, S. L. Hahn, and J. Quon. A self-learning evolutionary chess program. *Proceedings of the IEEE*, 92(12):1947–1954, 2004.

[35] J. Fürnkranz. Machine learning in computer chess: the next generation. *ICGA Journal*, 19(3):147–161, 1996.

[36] M. Gherrity. A game-learning machine. PhD Thesis, University of California, San Diego, CA, 1993.

[37] M. L. Ginsberg. GIB Library. http://www.cirl.uoregon.edu/ginsberg/gibresearch.html.

[38] M. L. Ginsberg. GIB: Steps toward an expert-level bridge-playing program. In *International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 584–589, Stockholm, SWEDEN, 1999.

[39] M. L. Ginsberg. GIB: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:303–358, 2001.

[40] H. Givens. PokerProbot. http://www.pokerprobot.com/, 2006.

[41] D. Gomboc, T. A. Marsland, and M. Buro. Evaluation function tuning via ordinal correlation. In *Advances in Computer Games: Many Games, Many Challenges: Proceedings of the International Conference on Advances in Computer Games (ACG-10)*, pages 1–18, Graz, Austria, 2003.

[42] J. Gould and R. Levinson. Experience-based adaptive search. In R. Michalski and G. Tecuci, editors, *Machine Learning: A Multi-Strategy Approach*, pages 579–604. Morgan Kaufmann, 1994.

[43] K. Greer. Computer chess move-ordering schemes using move influence. *Artificial Intelligence*, 120:235–250, 2000.

[44] E. A. Heinz. Adaptive null-move pruning. *ICCA Journal*, 22(3):123–132, 1999.

[45] F.-h. Hsu. *Behind Deep Blue.* Princeton University Press, Princeton, NJ, 2002.

[46] R. M. Hyatt. Crafty. ftp.cis.uab.edu/pub/hyatt, 2006.

[47] R. M. Hyatt, H. L. Nelson, and A. E. Gower. Cray Blitz. In T. A. Marsland and J. Schaeffer, editors, *Computers, Chess, and Cognition*, pages 111–130. Springer Verlag, New York, 1990.

[48] IBM Corporation. Deep Blue technology. http://www.research.ibm.com/know/blue.html, 2006.

[49] L. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[50] G. Kendall and G. Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 995–1002. IEEE Press, 2001.

[51] D. Kopec and I. Bratko. The Bratko-Kopec experiment: A comparison of human and computer performance in chess. In M. R. B. Clarke, editor, *Advances on Computer Chess 3*, pages 57–72. Pergamon Press, Oxford, 1982.

[52] C. Kotnik and J. K. Kalita. The significance of temporal-difference learning in self-play training td-rummy versus evo-rummy. In T. Fawcett

and N. Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003)*, pages 369–375, Washington, DC, USA, August 2003. AAAI Press.

[53] H. Kuijf. Jack - computer bridge playing program. http://www.jackbridge.com, 2006.

[54] M. Kusiak, K. Walędzik, and J. Mańdziuk. Evolution of heuristics for give-away checkers. In W. Duch et al., editors, *Artificial Neural Networks: Formal Models and Their Applications - Proc. ICANN 2005, Part 2, Warszawa, Poland*, volume 3697 of *LNCS*, pages 981–987. Springer, 2005.

[55] R. Levinson. MORPH II: A universal agent: Progress report and proposal. Technical Report UCSC-CRL-94-22, Jack Baskin School of Engineering, Department of Computer Science, University of California, Santa Cruz, 1994.

[56] R. A. Levinson and R. Snyder. Adaptive pattern-oriented chess. In L. Birnbaum and G. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 85–89. Morgan Kaufmann, 1991.

[57] A. Macleod. Perudo as a development platform for Artificial Intelligence. In *13th Game-On International Conference (CGAIDE'04)*, pages 268–272, Reading, UK, 2004.

[58] A. Macleod. Perudo game. http://www.playperudo.com/, 2006.

[59] J. Mańdziuk. Incremental learning approach for board game playing agents. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI2000)*, volume 2, pages 705–711, Las Vegas, USA, 2000.

[60] J. Mańdziuk. Incremental training in game playing domain. In *Proceedings of the International ICSC Congress on Intelligent Systems & Applications (ISA2000)*, volume 2, pages 18–23, Wollongong, Australia, 2000.

[61] J. Mańdziuk, M. Kusiak, and K. Walędzik. Evolutionary-based heuristic generators for checkers and give-away checkers. *Expert Systems*, 2007, (*accepted*).

[62] J. Mańdziuk and K. Mossakowski. Looking inside neural networks trained to solve double-dummy bridge problems. In *5th Game-On International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE04)*, pages 182–186, Reading, UK, 2004.

[63] J. Mańdziuk and D. Osman. Temporal difference approach to playing give-away checkers. In L. Rutkowski et al., editors, *7th Int. Conf. on Art. Intell. and Soft Comp. (ICAISC 2004), Zakopane, Poland*, volume 3070 of *LNAI*, pages 909–914. Springer, 2004.

[64] J. Mańdziuk and L. Shastri. Incremental Class Learning approach and its application to handwritten digit recognition. *Information Sciences*, 141(3–4):193–217, 2002.

[65] D. McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35:287–310, 1988.

[66] J. McCarthy. Homepage of John McCarthy. http://www-formal.stanford.edu/jmc/reti.html, 1998.

[67] M. L. Minsky. Steps towards artificial intelligence. In *Proceedings of IRE*, volume 49, pages 8–30, 1961.

[68] T. M. Mitchell and S. Thrun. Explanation based learning: A comparison of symbolic and neural network approaches. In P. E. Utgoff, editor, *Proceedings of the 10th International Conference on Machine Learning*, pages 197–204, San Mateo, CA, 1993. Morgan Kaufmann.

[69] D. E. Moriarty and R. Miikkulainen. Discovering complex othello strategies through evolutionary neural systems. *Connection Science*, 7(3):195–209, 1995.

[70] K. Mossakowski and J. Mańdziuk. Artificial neural networks for solving double dummy bridge problems. *Lecture Notes in Artificial Intelligence*, 3070:915–921, 2004.

[71] K. Mossakowski and J. Mańdziuk. Neural networks and the estimation of hands' strength in contract bridge. In L. Rutkowski et al., editors, *8th International Conference on Artificial Intelligence and Soft Computing (ICAISC06)*, Lecture Notes in Artificial Intelligence, pages 1189–1198, Zakopane, POLAND, 2006.

[72] M. Müller. Computer Go as a sum of local games: An application of combinatorial game theory. PhD Thesis, ETH Zürich, Switzerland, 1995.

[73] M. Müller. Computer Go. *Artificial Intelligence*, 134:145–179, 2002.

[74] A. Newell, J. C. Shaw, and H. A. Simon. Chess-playing programs and the problem of complexity. *IBM Journal of Research and Development*, 2(4):320–335, 1958.

[75] D. Osman and J. Mańdziuk. Comparison of tdleaf(λ) and td(λ) learning in game playing domain. In N. R. Pal et al., editors, *11th Int. Conf. on Neural Inf. Proc. (ICONIP 2004), Calcutta, India*, volume 3316 of *LNCS*, pages 549–554. Springer, 2004.

[76] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, 87(1–2):255–293, 1996.

[77] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Exploiting graph properties of game trees. In *13th National Conference on Artificial Intelligence (AAAI-96)*, volume 1, pages 234–239, Menlo Park, CA, 1996.

[78] E. A. Poe. Maelzel's chess player. *Southern Literary Messenger*, (April), 1936.

[79] J. B. Pollack, A. D. Blair, and M. Land. Coevolution of a backgammon player. In C. G. Langton and K. Shimokara, editors, *Proceedings of the Fifth Artificial Life Conference*, pages 92–98. MIT Press, 1997.

[80] A. Reinefeld. An improvement to the scout tree-search algorithm. *ICCA Journal*, 6(4):4–14, 1983.

[81] T. P. Runarsson and S. M. Lucas. Coevolution versus self-play temporal difference learning for acquiring position evaluation on small-board Go. *IEEE Transactions on Evolutionary Computation*, 9(6):628–640, 2005.

[82] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[83] J. Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE PAMI*, 11(11):1203–1212, 1989.

[84] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. New York: Springer-Verlag, 1997.

[85] J. Schaeffer. Chinook. http://www.cs.ualberta.ca/˜ chinook/, 2006.

[86] J. Schaeffer. Poki-X. http://www.cs.ualberta.ca/˜ games/poker/, 2006.

[87] J. Schaeffer, J. C. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53(2-3):273–289, 1992.

[88] J. Schaeffer, M. Hlynka, and V. Jussila. Temporal difference learning applied to a high-performance game-playing program. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 529–534, 2001.

[89] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook: The world man-machine checkers champion. *AI Magazine*, 17(1):21–29, 1996.

[90] N. N. Schraudolph, P. Dayan, and T. J. Sejnowski. Temporal difference learning of position evaluation in the game of go. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing 6*, pages 817–824. Morgan Kaufmann, San Francisco, 1994.

[91] N. N. Schraudolph, P. Dayan, and T. J. Sejnowski. Learning to evaluate go positions via temporal difference methods. In N. Baba and L. C. Jain, editors, *Computational Intelligence in Games*, volume 62, pages 77–98. Springer Verlag, Berlin, 2001.

[92] Y. G. Seo, S. B. Cho, and X. Yao. Exploiting coalition in co-evolutionary learning. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1268–1275. IEEE Press, 2000.

[93] C. E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41 (7th series)(314):256–275, 1950.

[94] B. Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134:241–275, 2002.

[95] H. Simon. Making managenet decisions: The role of intuition and emotion. In Weston Agor, editor, *Intuition in Organizations*, pages 23–39. Sage Pubs., London, 1987.

[96] D. Sklansky. *Hold'Em Poker.* Two Plus Two Publishing, Nevada, USA, 1996.

[97] D. Sklansky and M. Malmuth. *Hold'Em Poker for Advanced Players, 21st Century Edition.* Two Plus Two Publishing, Nevada, USA, 2001.

[98] B. Stilman. *Liguistic Geometry. From search to construction.* Kluwer Academic Publishers, Boston, Dordrecht, London, 2000.

[99] G. Stockman. A minimax algorithm better than alfa-beta? *Artificial Intelligence*, 12(2):179–196, 1979.

[100] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[101] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998.

[102] G. Tesauro. Neurogammon wins computer olympiad. *Neural Computation*, 1:321–323, 1989.

[103] G. Tesauro. Practical issues in Temporal Difference Learning. *Machine Learning*, 8:257–277, 1992.

[104] G. Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

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

[106] S. Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 1069–1076. The MIT Press, Cambridge, MA, 1995.

[107] S. Thrun. *Explanation-Based Neural Network Learning: A Lifelong Learning Approach.* Kluwer Academic Publishers, Boston, MA, 1996.

[108] S. Thrun and T. M. Mitchell. Learning one more thing. Technical report, Carnegie Mellon University, USA, CMU-CS-94-184, 1994.

[109] W. Tunstall-Pedoe. Genetic algorithms optimizing evaluation functions. *ICCA Journal*, 14(3):119–128, 1991.

[110] A. M. Turing. Digital computers applied to games. In B. V. Bowden, editor, *Faster than thought: a symposium on digital computing machines*, chapter 25. Pitman, London, UK, 1953.

[111] P. E. Utgoff. Feature construction for game playing. In J. Fürnkranz and M. Kubat, editors, *Machines that Learn to Play Games*, pages 131–152. Nova Science Publishers, Huntington, NY, 2001.

[112] A. van Tiggelen. Neural networks as a guide to opitimization. The chess middle game explored. *ICCA Journal*, 14(3):115–118, 1991.

[113] T. Yoshioka, S. Ishii, and M. Ito. Strategy acquisition for the game "othello" based on reinforcement learning. *IEICE Transactions on Information and Systems*, E82-D(12):1618–1626, 1999.

[114] A. Zorbist. Feature extractions and representation for pattern recognition and the game of go. PhD Thesis, University of Wisconsin, 1970.