University of Alberta

ALGORITHMS AND ASSESSMENT IN COMPUTER POKER

by

Darse Billings

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Department of Computing Science

Edmonton, Alberta
Fall 2006
Chapter 1

Introduction

1.1 Motivation and Historical Development

Games have played an important role in Artificial Intelligence (AI) research since the beginning of the computer era. Many pioneers in computer science spent time on algorithms for chess, checkers, and other games of strategy. A partial list includes such luminaries as Alan Turing, John von Neumann, Claude Shannon, Herbert Simon, Alan Newell, John McCarthy, Arthur Samuel, Donald Knuth, Donald Michie, and Ken Thompson [1].

The study of board games, card games, and other mathematical games of strategy is desirable for a number of reasons. In general, they have some or all of the following properties:

- Games have well-defined rules and simple logistics, making it relatively easy to implement a complete player, allowing more time and effort to be spent on the actual topics of scientific interest.

- Games have complex strategies, and are among the hardest problems known in computational complexity and theoretical computer science.

- Games have a clear specific goal, providing an unambiguous definition of success, and efforts can be focused on achieving that goal.

- Games allow measurable results, either by the degree of success in playing the game against other opponents, or in the solutions to related subtasks.
Apart from the establishment of game theory by John von Neumann, the strategic aspects of poker were not studied in detail by computer scientists prior to 1992 [1]. Poker features many attributes not found in previously studied games (such as checkers and chess), making it an excellent domain for the study of challenging new problems. In terms of the underlying mathematical structure and taxonomy of games, some of the most important properties include the following:

- **Poker is a game of imperfect information.** Various forms of uncertainty are a natural consequence. This property creates a necessity for using and coping with deception (specifically, bluffing and trapping),¹ and ensures a theoretical advantage for the use of randomized mixed strategies.

- **Poker has stochastic outcomes.** The element of chance (the random dealing of cards) at several stages of the game introduces uncertainty and uncontrollable outcomes. Among other things, this adds a high degree of variance to the results, and makes accurate assessment of performance difficult.

- **Hidden states in poker are partially observable.** A player can win a game uncontested when all opponents fold, in which case no private information (i.e., the cards held by any of the players) is revealed. Partial observability makes it much more difficult to learn about an opponent's strategy over the course of many games, both in theory and in practice.

- **Poker is a non-cooperative multi-player game.** A wealth of challenging problems exist in multi-player games that do not exist in two-player games. Multi-player games are inherently unstable, due in part to the possibility of coalitions (i.e., teams), but those complexities are minimized in a non-cooperative game [60, 63].

As a general class, stochastic imperfect information games with partial observability are among the hardest problems known in theoretical computer science. This

---

¹ Technical terms and jargon from poker theory appear in bold face italics throughout this dissertation, and are defined in Appendix A: Glossary of Poker Terms.
class includes many problems that are easy to express but are computationally un-decidable [20, 38].

In practice, writing a program to play a legal game of poker is trivial, but designing and implementing a competent poker player (for example, the strength of an intermediate human player) is a challenging task. Writing a program that also adapts smoothly to exploit each opponent’s particular playing style, betting patterns, biases and tendencies is a difficult learning problem.

1.2 Major Influences

Since there was no specific research on poker game-playing in the computer science literature prior to 1992, the mathematical models and scientific methodology for the research project were based on other available sources of knowledge. Three major influences were:

1. Classic books on poker strategy,
2. Fundamental principles of game theory, and
3. Traditional game-playing programs based on game-tree search.

1.2.1 Classic Books on Poker Strategy

The single most important book to date for understanding poker strategy is “The Theory of Poker” by David Sklansky [55]. Other books by Sklansky and frequent co-author Mason Malmuth also provide valuable insights [56, 57]. Additional resources, and their utility for scientific research, are discussed in Billings [1].

Although written for human students of the game, the clear exposition in these texts allows a mathematically inclined reader to gain an appreciation for the underlying logical structure of the game. This insight suggests a wealth of algorithmic possibilities to be explored for knowledge-based approaches. Incorporating probabilistic knowledge into a formula-based architecture was the topic of our early research, and is discussed in Chapter 2. The serious limitations of that approach and lessons learned from the research are discussed in Chapter 6.
1.2.2 Fundamental Principles of Game Theory

The game of poker was used as a model of adversarial conflict in the development of mathematical game theory by John von Neumann in the 1920s [69]. The (general probabilistic) Minimax Theorem proves that for any two-player zero-sum game, there always exists an equilibrium strategy. Using such a strategy ensures (at least) the game-theoretic value of the game, regardless of the opponent’s strategy. Thus, playing an equilibrium strategy would guarantee not losing in the long run (assuming the players alternate positions over many games).

John Nash extended the idea of equilibrium strategies to non-zero-sum games and multi-player games, again using poker as an example [41]. A set of strategies are said to be equilibrium when no player can benefit from unilaterally changing their style or strategy [25, 24, 70]. The 1972 book “Winning Poker Systems” by Norman Zadeh attempted to apply game theoretic strategies to a variety of real poker variants, with some degree of success [71, 72, 1].

There are serious limitations to equilibrium strategies in practice, because they are static, are oblivious to the opponent’s strategy, and have implicit assumptions that generally give the opponent far too much credit. These inherent limitations have been clearly demonstrated in the simpler imperfect information games of Rock-Paper-Scissors [4, 2] and Oshi-Zumo [17].

Finding an approximation of an equilibrium strategy for real poker is discussed in Chapter 3. Further insights and limitations of applying game-theoretic methods in general are discussed in Chapter 6.

1.2.3 Traditional Game-Tree Search

Many lessons have been learned from traditional high-performance game-playing programs. Research from 1970 to 1990 focused primarily on chess, and other two-player zero-sum games with perfect information. As these programs improved, a recurring theme was an increasing emphasis on computer-oriented solutions, and

---

\[2\] In practice, poker is normally a negative constant sum game, because the host (e.g., casino) charges the players a rake or a time charge. Nevertheless, an equilibrium strategy (or approximations thereof) would be highly useful for playing the real game.
diminishing reliance on human knowledge [46, 34].

In many games with relatively simple logistics but complex strategy, computer programs have now surpassed the best human players by a vast margin. In each case, the formula for success has been the same: deep look-ahead search of the game tree using the highly efficient *alpha-beta search algorithm*, combined with a domain-specific *evaluation function* applied at the nominal leaf nodes of the search frontier [47].

- In 1990, the checkers program CHINOOK earned the right to challenge the human world champion, Marion Tinsley, in a title match. CHINOOK lost narrowly in 1992, but won the return match in 1994, becoming the first computer program to win an official world championship in a game of skill [48]. An effort is now underway to solve the game of checkers, with two of the official tournament openings now proven to be drawn [50].

- In 1997, the Othello program LOGISTELLO defeated the human world champion, Takeshi Murakami, in a six game exhibition match, winning all six games [15, 16].

- In 2000, the Lines of Action program MONA won the *de facto* world championship, defeating all of the top human players, and winning every game it ever played against human opposition [5, 3].

- In 2002, the ancient game of Awari was strongly solved, computing the exact minimax value for every reachable position in the game [68, 45]. Although the best programs already played at a level far beyond any human player, the difference between super-human play and perfect play was shown to be enormous [44].

- In 1997, the chess machine DEEP BLUE won a short exhibition match against the human world champion, Garry Kasparov, scoring two wins, one loss, and three draws [18]. This led to a widely held but premature belief that chess programs had surpassed all human players. Several years later, the programs SHREDDER, FRITZ, and JUNIOR demonstrated performances on par with the
best human players. In 2005, the program HYDRA convincingly defeated one of the strongest human players, Michael Adams, scoring five wins, zero losses, and one draw, providing a strong argument for the dominance of chess programs [23].

Similar successes continue to be obtained for this general class of games, usually with the same architecture of alpha-beta search combined with a good heuristic evaluation. The approach has not been successful for the game of Go, however, owing to the high branching factor and vast search space (for $19 \times 19$), and the fact that goals and subgoals are very difficult to assess with heuristic evaluation [40].

### 1.3 Extending Game Tree Representations

Many games admit some element of *random chance*. The traditional game tree representation can be extended to handle stochastic outcomes by incorporating *chance nodes*. Each branch from a chance node represents one of the finite number of random outcomes. From a search perspective, all of these branches must be considered, and combined to obtain an overall *expected value* (EV), so the size of the game tree grows multiplicatively with each subsequent chance node. The alpha-beta search algorithm is not applicable to this class of problems, but other search algorithms such as *-Minimax [28, 27] and simulation methods [67, 54] are able to contend with this form of uncertainty adequately. The property of stochasticity has not been a major impediment to world-class computer performance in practice.

The game of backgammon is a classic example of a perfect information game with an element of stochasticity (the roll of the dice). Excellent evaluation functions have been learned automatically from self-play [66, 67], resulting in several programs that are at least on par with the best human players, without requiring deep search [27].

*Multi-player games* are much more challenging, both in theory and in practice; but to date they have not received a lot of attention in AI research. Several search algorithms for multi-player game trees are known in the literature, but the potential for guaranteed safe pruning of the search space is much lower than that enjoyed by
the alpha-beta algorithm [59, 61, 64]. This fact, combined with the larger branching factor resulting from many players, means that deep search is less feasible, in general.

Moreover, multi-player games are inherently unstable, being subject to possible collusion between players.\(^3\) In practice, minor differences in search algorithms for multi-player game trees can produce radically different results. For example, two different move choices could be exactly equal in value for Player A, but could dictate whether Player B or Player C wins the game. Due to these volatile conditions, good opponent modeling (for example, knowing each player’s method of tie-breaking between equal moves) is necessary to obtain robust and reliable results [60, 62, 63, 65].

However, the major distinguishing factor between poker and other games is the property of imperfect information, the effects of which can range from obvious to subtle, from inconsequential to profound. One important consequence is that a complete strategy in poker must include a certain degree of deception, such as bluffing (betting or raising with a weak hand) and trapping (playing a strong hand as though it were weak). This fact was one of the earliest results in game theory [69]. The objective of these deception tactics is to disguise the strength of one’s hand (called information hiding), and to create uncertainty in the beliefs of the opponent, resulting in greater profitability overall.

The relative balance of these deceptive plays (and of responses to the opponent’s actions) is of critical importance. Any inappropriate imbalances necessarily imply the existence of statistical biases, patterns, or other weaknesses that are vulnerable to exploitation. Since there may be many ways of obtaining the desired balance of plays in poker, the players have some discretion in how they actually achieve that balance. For example, a particular situation might call for a 10% bluff frequency, but the player is otherwise free to decide when to bluff or not bluff. As a result, there is in general no single best move in a given poker situation.

This is in stark contrast to a perfect information game, where there is a single

\(^3\) This is true even for ostensibly non-cooperative games, like poker, since that ethic cannot be enforced, in general. John von Neumann showed that multi-player games become stable only when they devolve into a two-player game between two coalitions [69].
move, or small set of moves, that preserves the game-theoretic value of the game. In backgammon, for example, there is typically only one move in a given position that will maximize the expected value against perfect play.

Furthermore, theoretically correct play in an imperfect information game requires probabilistic mixed strategies, where different moves are chosen some fraction of the time in identical circumstances. In contrast, a deterministic pure strategy (always playing one particular move in a given situation) is sufficient to obtain the game-theoretic value in a perfect information game (although the player may choose randomly from a set of equal-valued moves).

Game trees can be further extended to handle imperfect information games, with the inclusion of information sets. An information set is a set of decision nodes in the game tree that cannot be distinguished from the perspective of a given player. Since the opponent’s cards are hidden in poker, this corresponds to the complete set of all possible opponent holdings in a given situation. Obviously, the same policy (such as a particular mixed strategy) must be applied identically to all of the nodes in the information set, since it is not possible to know precisely which of those states we are in.

The immediate consequence is that nodes of an imperfect information game tree are not independent, in general. Thus, a divide-and-conquer search algorithm, such as the alpha-beta minimax technique, is not applicable to this class of problems, since sub-trees cannot be handled independently.

Another characteristic that distinguishes poker from perfect information board games is that it is not enough to simply “play well”, while largely ignoring the existence of the opponent. To maximize results, it is absolutely essential to understand the opponent’s style, and the nature of their errors (such as patterns or tendencies in their play).

As a simple demonstration, consider two opponents, one of whom bluffs far too often, the other of whom seldom bluffs. Both are “weak players”, in an objective sense. To maximize our profit against the former, we call (or perhaps raise) more

---

4 A perfect information game tree can be thought of as a special case in which all decision nodes belong to their own unique information set.
often with mediocre hands. To maximize against the latter, we fold more often with marginal hands. Our strategy adjustments are diametrically opposite, depending on the nature of the opponent’s predictable weaknesses.

In perfect information games, simply playing strong moves will naturally punish weaker moves, and it is not necessary to understand why an opponent is weak. Opponent modeling has been investigated in chess and other two-player perfect information games, but has not led to significant improvements in performance [33, 31, 32, 19].

An interesting case study is the game of Scrabble. Although Scrabble is technically a game of imperfect information, that property plays a relatively minor role in strategy. Super-human performance has been attained for the two-player game without special consideration for opponent modeling. Relatively simple techniques, such as Monte Carlo simulation and selective sampling, can be used to account for the unknown information adequately [52, 54]. Moreover, the strengths of the computer player, including perfect knowledge of the dictionary and the ability to consider every legal play, are sufficient to surpass all human players in skill [53].

Figure 1.1 shows a small portion of the imperfect information game tree for any Limit poker variant, featuring decision nodes for each player during a betting round. In general, a player will choose one of three possible actions: fold (f), call
(c), or raise (r). By convention, capital letters are used to indicate the actions of the second player.

When the betting round is complete, the game is either over (one player folded, leading to a terminal node), or the game continues with the next chance event (cards being dealt). Figure 1.2 shows the game tree for a complete betting round of 2-player Limit poker (with a maximum of three raises per round).

Figure 1.3 illustrates the notion of information sets in the game of 2-player Limit Texas Hold’em. Only three of the 1,624,350 branches from the initial chance node (i.e., the dealing of the hole cards) are shown. For any given hand, a player will have 1225 indistinguishable states, since the opponent’s cards are not known. Naturally, the same decision policy will apply to all states in that information set.

Figure 1.4 shows a high-level view of the structure of Texas Hold’em. Each betting round is depicted with a triangle, and corresponding chance nodes are collected to indicate the stage of the hand. The numbers on the left indicate the branching factors at each stage, leading to more than a quintillion (1,179,000,604,565,715,751)
Figure 1.3: Information sets in an imperfect information game tree.

Figure 1.4: The overall structure of the Texas Hold’em game tree.
nodes of all types.

Defining appropriate search algorithms for this fundamentally different mathematical structure of game tree is discussed in Chapter 4. The problems encountered and the necessary modifications for future research are discussed in Chapter 6.

1.4 The University of Alberta Computer Poker Research Group

The University of Alberta Computer Poker Research Group (CPRG) is the major contributor to the academic literature on poker game-playing AI. The purpose of this section is to explain the structure of the research group and the roles of the members. Since it is a collaborative team effort, it is necessary to identify the specific contributions made by this author, distinguishing them from the work of other members, and the group as a whole. All conceptual designs, architectures, and specific component algorithms discussed in this dissertation are attributable to the author unless noted otherwise. The use of the words “our” and “we” in this document refer to the group as a whole.

The research began in 1992 with scientific foundations, methodologies, and research philosophy [1]. This included a complete basic implementation, along with computer-oriented algorithms (rather than knowledge-based methods) for advanced hand assessment, simulation techniques, and other essential functions. The CPRG was formed in 1997 to follow up on this work. The author is the lead architect for the group, and the domain expert.\(^5\)

Dr. Jonathan Schaeffer is a co-founder, scientific advisor, and the administrative head of the CPRG. Dr. Duane Szafron is also a co-founder and scientific advisor. Dr. Robert Holte joined the group in 2001, contributing expertise in machine learning and linear programming. Dr. Michael Bowling joined the group in 2004, adding more knowledge in game theory and learning algorithms. Several M.Sc. students, summer students, and one full-time programmer/analyst have contributed to implementations and experimentation of the resulting systems.

\(^5\) The author played poker professionally from 1996 to 1999, after several years of studying and extending poker theory.
• **Denis Papp** (M.Sc. student) constructed the original LOKI system, in C++, re-implementing the author’s Monte Carlo simulation and weighted enumeration algorithms for hand assessment, along with numerous other components (discussed in Chapter 2) [42]. He incorporated the GNU poker library high-speed hand comparators as a core function [39]. He implemented all of the communication protocols to enable LOKI to participate in poker games on the IRC Online Poker Server [14].

• **Lourdes Peña** (M.Sc. student) built on top of the existing system (LOKI II) for the first implementation of selective simulation techniques and the subsequent experiments [43, 11].

• **Aaron Davidson** (M.Sc. student) re-wrote the entire codebase (re-christened POKI), in Java, using native methods where necessary to maintain high-speed performance. He performed code reviews with the author, discovering and correcting numerous errors, and made significant improvements to many components. The neural network approach for opponent modeling was entirely his own design [22, 7, 21]. Aaron developed test suites for conducting experiments, and wrote the University of Alberta online poker server, allowing extensive empirical testing. He also proposed new simulation methods to reduce the problem of compounding errors with sequential actions. Those ideas were refined and reformulated by the author as the Miximax and Miximix algorithms for imperfect information game-tree search (discussed in Chapter 4). Aaron then implemented and co-developed refinements for those systems [8].

• **Neil Burch** (programmer/analyst) implemented numerous algorithms and support routines, and performed many of the scientific experiments reported in CPRG publications. He developed a system for specifying general poker game definitions and converting them into the sequence form linear program encoding described by Koller *et al.* [36, 37]. Neil oversaw all related computations, using a commercial linear program engine (CPLEX) to produce the game-theoretic equilibrium solutions (discussed in Chapter 3) [6]. He
also wrote alternate implementations of adaptive architectures (discussed in Chapter 4), for the purposes of testing and comparison [8].

- **Terence Schauenberg** (M.Sc. student) implemented the adaptive *Miximax* algorithm; co-developed the data structures, parameters, and abstractions used in VEXBOT; and performed related experiments (discussed in Chapter 4) [8, 51]. He implemented the author’s Expected Value Assessment Tool (EVAT) and Luck Filtering Assessment Tool (LFAT), which were precursors to the Ignorant Value Assessment Tool (DIVAT) performance metric (discussed in Chapter 5) [10]. Terence has also investigated a variety of methods for learning approximations of Nash-equilibrium solutions by means of *fictitious play*.

- **Bret Hoehn** (M.Sc. student) performed an independent study of opponent modeling, under the direction of Dr. Holte. He used the tiny game of Kuhn poker to reduce the complexity of learning an opponent’s weaknesses and quickly adopting an appropriate counter-strategy [30, 29]. Serious limitations are encountered despite the large reduction in pertinent variables, demonstrating some of the fundamental impediments to rapid learning and adaptation in partially observable stochastic domains.

- **Morgan Kan** (M.Sc. student) implemented the author’s DIVAT method for direct assessment of poker decision quality, and performed numerous experiments during its development that led to deeper insights into the problem (discussed in detail in Chapter 5) [10].

The research group has expanded rapidly in recent years, with the addition of post-doctoral fellows **Finnegan Southey** and **Martin Zinkevich**; M.Sc. students **Chris Rayner**, **Nolan Bard**, and **Mike Johanson**; and research associate **Carmelo Piccione**.

The research has also branched out with several new topics (which are outside of the scope of this thesis), including development of the author’s *pdf-cutting algorithm* for creating parameterized probabilistic profiles of the poker strategy space, and new methods for rapid learning using Bayesian inference methods [58].
1.5 Summary of Contents

This thesis identifies four distinct approaches to computer poker-playing, with a corresponding program architecture designed for each technique. Each approach has proven to be highly successful, despite the inherent theoretical limitations. Each generation has superseded the previous one by addressing the most important limitations discovered during the extensive empirical testing, which includes millions of games played. The core chapters of this paper-based thesis are comprised of the academic papers that stemmed from each of these studies.

1.5.1 Knowledge-based Methods and Simulation (1997-2001)

The first two approaches, discussed in Chapter 2, are formula-based strategies and simulation. Formula-based methods are a generalization of the somewhat intuitive but overly-simplistic method of deterministic rule-based systems. Various forms of simulation are an important technique for enhancing the performance of established programs, or for playing the game directly.

The representative paper for the formula-based and simulation methodology is “The Challenge of Poker”, published in the journal Artificial Intelligence [7]. The paper subsumes most of the previous work by the CPRG [13, 12, 42, 14, 11, 43, 49, 22]. Some of the most important contributions of this work include:

- Expert systems for the (relatively uncomplicated) strategy of the first betting round (the pre-flop), based on values determined by Monte Carlo roll-out simulations.
- Exhaustive enumeration algorithms for the assessment of hand quality (hand strength and hand potential).
- Selective simulation techniques for enhancing and refining expected value estimates.
- Statistical opponent modeling, and routines for the utilization, maintenance, and updating of relevant belief states.
• Procedures and advanced modules for post-flop betting strategy, incorporating general and specific opponent modeling, and including occasional deceptive plays (bluffing and trapping).

• The only poker programs (POKI and its derivatives) that are known to play better than an average human player who plays in low-limit casino games.

In recent years, numerous hobbyists and researchers have referred to these early publications, and based their poker programs on those architectures. They have invariably discovered the advantages and the inherent limitations of knowledge-based systems for themselves.

1.5.2 Game-Theoretic Methods (2002-2003)

The third approach, discussed in Chapter 3, is based on game theory. This addresses the serious short-comings of the formula-based approach in achieving a well-balanced betting strategy, with an appropriate ratio of deceptive plays (bluffs and traps) in relation to the frequency of legitimate bets, calls, and folds.

The corresponding paper “Approximating Game-Theoretic Optimal Strategies for Full-scale Poker”, won the Distinguished Paper Award at the International Joint Conference on Artificial Intelligence in 2003 [6]. Some of the most important contributions of this work include:

• Abstraction techniques for exact and near-exact reformulation of defined poker games, yielding reductions of the problem size by about two orders of magnitude.

• Crude but powerful abstraction techniques, capable of reductions of the problem size by more than ten orders of magnitude (from \(10^{18}\) states to less than \(10^8\) states), but with no guarantees on error bounds. These severe abstractions nevertheless maintain the key properties and relationships of the game, such that exact solutions to the abstract game provide reasonable approximations for use in the full-scale game.
• Poker programs (known collectively as PSOPTI or SPARBOT) that exhibit a vast improvement in skill for two-player Limit Texas Hold’em.

• The first demonstration of a program that could be competitive with a world-class player.

Several other researchers have recently built on this work, including Andrew Gilpin and Tuomas Sandholm at Carnegie Mellon University [26].

1.5.3 Adaptive Imperfect Information Game-Tree Search (2004)

The fourth approach, discussed in Chapter 4, is based on imperfect information game-tree search, with built-in data structures for opponent modeling and adaptive play. This addresses the serious short-comings of the game-theoretic and formula-based approaches in rapidly adapting to the opponent’s style of play, exploiting biases and predictable patterns, and making it much more challenging to learn against the program. The Miximax and Miximix algorithms accommodate the more general class of game trees where some domain information is hidden from one or more players, and where each decision node may be associated with a randomized mixed strategy, rather than a single action.

The related paper is “Game-Tree Search with Adaptation in Stochastic Imperfect Information Games”, from the 2004 Computers and Games conference [8]. Some of the most important contributions of this work include:

• A generalized framework for stochastic imperfect information games based on generalizations of the (perfect information) Expectimax algorithm.

• Refined methods for opponent modeling, with direct applicability to expected value calculations for each available action.

• Abstraction techniques for partitioning distinct betting sequences into a manageable number of highly correlated situations.

• The experimental poker program VEXBOT, which eventually learns to defeat all known programs by a large margin, and can provide a serious threat to world-class players.
1.5.4 Assessment of Performance (2005)

Chapter 5 addresses the difficult issue of performance assessment in poker. Unfortunately, measuring the performance of a poker program simply by playing games requires many thousands of trials to produce a single data point, which is then only relevant to that one narrow set of preconditions. Moreover, performance in poker is decidedly non-transitive: “A beats B” and “B beats C” does not imply that “A beats C”, nor does it say anything about the relative magnitude of win rates against future opponents. The outcome of any particular match may be governed by a clash of styles, rather than the objective strengths of the players. Testing against a wide variety of opponents is essential, but is not guaranteed to be sufficient.

To combat these serious obstacles, the author invented the Ignorant Value Assessment Tool (DIVAT). Similar metrics (called EVAT and LFAT) were developed previously for analyzing experiments and matches, but they had serious shortcomings. DIVAT provides an objective means of accurately assessing decision quality, with a large reduction in the natural variance of outcomes. The tool is based on a hindsight expected value assessment of each decision, comparing the actual equities against a theoretically motivated baseline [9, 35]. The paper “A Tool for the Direct Assessment of Poker Decisions” has been accepted for publication in the International Computer Games Association Journal [10].

1.5.5 Conclusion

Chapter 6 concludes the thesis with a retrospective look at some of the most important lessons that have been learned over the years. A major theme that ties these publications together is the evolution of architectures for poker programs. Each approach has both theoretical and practical limitations. Some of these limitations were known before the system was built, but the full implications can only be understood after many implementations and refinements are tested. Recurring themes include the need for well-balanced betting strategies, better opponent modeling, and faster learning and adaptation.

---

6 The ‘D’ refers to the author’s first initial.
For each architecture, program development is often a cyclic process, with each iteration introducing an improved method for handling a particular aspect of the game that had become the limiting factor to performance. In some cases, the cycle was very long and arduous, with some “temporary” components not being re-visited again for years.

There has always been a healthy interplay between theory and practice. Diminishing returns from these refinements help identify fundamental limitations that necessitate a revolutionary change – a new approach and new architecture that does a much better job of addressing some critical strategic aspect of the game. Ultimately, we seek unifying methods that reduce the complexity of the system, and eliminate human intervention, allowing the program to “think for itself”.

Although much work remains to be done, poker programs have evolved from very weak players to programs that are a serious threat to world-class players. The past successes and failures suggest what types of solutions are the most viable in general, and which directions of research will be most fruitful in the future.
Bibliography


