

## INTRODUCTION TO THE SPECIAL ISSUE ON GAMES: STRUCTURE AND LEARNING

BARNEY PELL

*Caelum Research Corporation  
NASA Ames Research Center  
AI Research Branch, Mail Stop: 269-2  
Moffett Field, CA 94035-1000  
pell@ptolemy.arc.nasa.gov*

SUSAN L. EPSTEIN

*Department of Computer Science  
Hunter College and The Graduate School of The City University of New York  
695 Park Avenue, New York, NY 10021  
epstein@roz.hunter.cuny.edu*

ROBERT LEVINSON

*Department of Computer Science  
University of California, Santa Cruz, CA 95060.  
Email: levinson@cse.ucsc.edu. Phone: 408-459-2087*

### 1. INTRODUCTION

The universality of strategy games suggests that they reflect some basic insights into the nature of human intelligence. Turing cited rudimentary chess reasoning as a hallmark of AI, and some of the earliest significant research was on chess and checkers. Today computer game playing continues to be a central concern because it addresses the fundamental issues in AI: knowledge representation, search, planning, and learning. Moreover, the multi-agent nature of games makes game playing an excellent forum for addressing core issues such as contingency planning and reasoning about actions and plans of other agents, while competition between programs and with humans provides useful benchmarks and encourages progress.

This special issue highlights recent innovative work by a broad spectrum of researchers and practitioners. There are clearly at least three very different approaches to computer game playing: a high-performance determination to play better than any human, a cognitively-oriented exploration of learning and behavior, and a mathematical theory of heuristics and game-playing. The competition and cooperation among these approaches drives exciting and significant work whose results extend to many other problems with large search spaces.

The traditional AI approach to game playing relies upon fast, deep search to look ahead from the current game state to all possible ways to complete the contest. For difficult games there are so many alternatives that only a fragment of the future possibilities can be considered. Therefore move selection must also rely upon a human-designed evaluation function to estimate the worth of game states prior to the end of a contest. This technique is classically supplemented by an extensive catalog of expert openings (an opening book) and pre-computed solutions to simple endgame positions (an endgame database). Because this approach tends to rely on raw computer power to compensate for a lack of knowledge or selectivity in search, these methods are often referred to as the “brute force” approach.

By many standards, this brute-force approach has been remarkably successful. Carefully engineered, deep-searching computers now dominate all but a few humans in a number of challenging games, including chess, checkers, and Othello. The surprising success of the engineering approach on these games has prompted researchers in other fields to seek similar search-intensive solutions to their problems, including theorem-proving and natural language processing (see Marsland's discussion in (Levinson *et al.* 1991)).

The brute-force approach to games has its limitations, however. First, where this approach is applicable, considerable engineering effort is required to achieve success. This effort manifests itself in highly-efficient, special-purpose representations (sometimes with game-specific hardware (Ebeling 1986)), fine-tuned evaluation functions with hand-crafted features and weights, and careful optimization of numerous search engine parameters (including time management decisions). Because people perform all this game-specific optimization, the resulting solutions are labor-intensive and difficult to generalize.

Second, the brute-force approach has achieved success only on a small subset of the games humans actually play, a subset that has received most of the attention in AI research. All of these games are two-player, finite, deterministic games of perfect information with reasonably low average branching factors. Even on a game with all those properties but a higher branching factor, such as Go, the brute-force approach has failed to produce a program that can compete favorably against a human novice. Brute force becomes even more problematic when applied to games with chance (like backgammon (Tesauro 1994)), incomplete information (like card games and battleship), or multiple players (like risk and diplomacy (Hall & Loeb 1992)).

Finally, the level of engineering characteristic of the brute-force approach is, to many, scientifically unsatisfying. Psychological studies (de Groot 1965; Reitman 1976) and common sense suggest that humans achieve expert performance at games without considering millions of game positions. Moreover, humans are able to generalize their knowledge and experience, teach others via lessons and books, and transfer their knowledge to widely different games and life situations. While we have achieved world-champion-level computer player of some popular games, we have made far less progress at understanding the general principles that enable people to become expert at and efficiently solve a wide variety of problems.

The preceding discussion illustrates that, even in the wake of major successes, the field of computer game-playing is re-evaluating itself. This introspection was manifested in a successful IJCAI-91 panel on the role of computer chess in AI research (Levinson *et al.* 1991). More recently, a growing community of AI researchers is interested in addressing the limitations of the traditional approach from both scientific and engineering perspectives. This growth led to a successful AAAI fall symposium on Learning and Planning in games (Epstein & Levinson 1994). This workshop was the seed for this special issue.

## 2. CONTENTS

This issue contains a cross-section of research on computer game-playing. Each article addresses one or more limitations of the traditional approach: overcoming engineering difficulties, machine learning, moving to new classes of games, and generality in game-playing.

## 2.1. Exploiting Structure in Search

Because the traditional engineering approach is both search-intensive and labor-intensive, one good approach is to add structure to make search more tractable. The papers by Allis *et al.* and Gasser show how additional structure of games can be exploited to dramatically augment the power of brute-force search. In both cases, the result is the construction of a program which solves a game entirely. By *solving* a game, we mean that the program determines the game-theoretic value of the initial position (von Neumann & Morgenstern 1944) and will achieve this value even against an opponent that plays perfectly.

Although both papers produce programs that solve a specific game, they use different methods because the nature of the games are different. Allis *et al.* consider the game of go-moku, which is characterized by a high branching factor (on the order of 200 moves per position). It turns out, however, that most moves in a given position can be rejected because they enable the opponent to force a win using a sequence of threats. To exploit this structure, Allis *et al.* developed a technique called *threat-space search*, which builds a search tree consisting only of moves related to threats and responses. The results of this search are used within a heuristic search method, called *proof-number search*, which helps the program to prioritize its search in the cases where no forcing lines can be found. Using a combination of these search techniques, an implemented program was able to solve the game while exploring only a tiny fraction of its search space.

The paper by Gasser considers Nine-Men's Morris, a game in which almost every position is relevant to the value of the game. Without the ability to prune the way Allis *et al.* did for go-moku, it was necessary to exploit different structure to tame the search space. Gasser performed an elaborate combinatorial and structural analysis of the game to avoid generating redundant symmetrical positions or positions that could never arise in the course of a game. He then generated a set of databases corresponding to middlegame and endgame positions, using a traditional technique called retrograde analysis. Since this technique could not enumerate positions early in the game, Gasser used a deep (18-ply) brute-force search from the opening position to reach into the endgame database and complete the solution of the game.

## 2.2. Machine Learning

Even with a brute-force approach, humans have to make many difficult decisions and fine-grained optimizations to achieve high performance in game-playing programs. Tesauro (1994) has already demonstrated that top-level backgammon play can be achieved through reinforcement learning. The next three papers, by Fawcett, Morales, and Markovitch and Sella, all present methods by which programs acquire some of their own knowledge and performance optimization directly from examples or experience playing the game.

Fawcett's paper addresses the construction and selection of features for use in an evaluation function. This tends to be a laborious task in the construction of all game-playing programs, but the performance of a program tends to be highly dependent on a good solution to this problem. The paper describes a theory of *knowledge-based feature discovery* and its implementation in a program called Zenith. The program starts with a set of basic features defined in the rules of a game and produces new features by transformations of existing features. These new features are then monitored during performance, and only the most effective ones are used in

future iterations. The program has been applied to Othello and a telecommunications domain with good results; the theory may be applicable to a wide variety of problems.

While Fawcett's Zenith program learns features for use in an evaluation-function during tree-search, the paper by Morales describes a system which learns patterns for use in a pattern-controlled search. \*\*\*His system, called PAL, operates directly on the game rules and general-purpose knowledge. PAL uses Inductive Logic Programming (ILP), a fast-growing research area that combines inductive and deductive machine learning. The paper provides an interesting example of how games pose significant challenges for techniques normally applied only to single-agent search problems. Morales overcomes these challenges with the addition to the learning method of new heuristics and constraints. In his application to chess, Morales shows how PAL learns strategies for some chess endgames with minimal human assistance.

The paper by Markovitch and Sella applies learning techniques to acquire yet another form of pattern knowledge, one concerned with resource allocation and time management. This is an area which has been neglected in most of the previous work on learning for game-playing. Their approach generates features from example board positions with local templates, and then uses these features and the example positions to form a classification network. When playing a game, the program uses this learned network to select crucial positions worthy of extra search (at a cost). This approach is shown to give increased performance on experiments in the game of Checkers.

### 2.3. New Classes of Games

The next two papers both focus on games of imperfect information, which have received comparatively little attention in computer game-playing. The lack of knowledge about the opponent's possible moves gives such games a much higher branching factor, and therefore reduces the aptness of game-tree search.

The paper by Smith and Nau describes a planning approach to card play in contract bridge. Their approach first represents information about the game in a task network extended to represent multi-agency and uncertainty. Their program then uses this network to search game trees in a space of tactical and strategic schemes, rather than individual moves, thus reducing the large branching factor. Their approach is thus similar to Allis *et al.*'s threat-space search technique and to Morales' extension of an AI technique to the multi-agent domain, with a resultant increase in generality.

While Smith and Nau address a particular game of imperfect information from a heuristic perspective (obtaining good play using human knowledge), the paper by Blair *et al.* addresses the entire class of games and does so from a game-theoretic perspective. It surveys imperfect information games and contrasts them with games of perfect information and games of chance. The paper then presents a number of results concerning the complexity of solving games of incomplete information, just as the papers by Allis *et al.* and Gasser solved games of perfect information. Following this discussion, the paper introduces a technique for solving imperfect information games which contain a particular structure, and discusses the technique's applicability as a heuristic for games that lack this structure.

## 2.4. Generality

The last three papers in this issue, by Levinson, Pell, and Epstein *et al.*, are directly concerned with the scientific problems of generality in game-playing. As such, each discusses an implemented system that plays more than one game. They all advocate the design of a general game-playing program whose testing on many games will support the transfer and generalization of knowledge. While the papers are unanimous in their emphasis on games as vehicles for discovering general principles for artificial intelligence, the approaches have interesting differences.

The paper by Levinson provides a blueprint for the development of a fully-independent, generalized search system. His central thesis is that all search problems have a mathematical structure which can be exploited through learning and analysis, and the ability to exploit this structure is the source of power for general problem-solving systems. The paper elaborates this thesis as a unifying view for research on game-playing and documents progress to date in filling out the blueprint. Levinson's own progress builds on the Morph chess system, which extracts graphical structures during game-playing experience, learns the value for these structures by a variety of methods, and uses this knowledge to improve its performance at chess. His current paper describes the MorphII system, which generalizes Morph into a fully domain-independent game-player based on reinforcement-learning. This is achieved by defining a compact, generic representation for search problems. MorphII manipulates problems expressed in this generic representation, and learns by generating patterns derived from problem structure and evaluating them based on experience.

The paper by Pell is very much in the spirit of Levinson's thesis, but, driven by a particular methodological problem that arises in general game-playing: how can we demonstrate that a game-playing program is general? Success in competition in any one game could be as much a product of clever, game-specific engineering as a product of general principles. Pell's approach, called *meta-game playing*, is to develop programs which play a well-defined class of games entirely without human assistance. While these programs can be developed and illustrated using well-known games, true evidence of generality comes from strong performance against other meta-game programs in the context of games produced by an automatic game generator and never before seen by the human developer. The paper describes the first program implemented within this paradigm, called METAGAMER, which plays a class of chess-like games without human assistance. The program uses general knowledge about this class of games to construct, from first principles, its own game-specific representation and evaluation function for each game it plays. Experiments on chess and checkers illustrate how METAGAMER applies general knowledge to specific games, while results on generated games assess the applicability of this knowledge across the entire class of games.

Epstein *et al.*'s paper, the last in the collection, also fits Levinson's framework of exploiting mathematical structure to perform general and powerful search. The system called Hoyle has been used to explore a wide variety of themes involving general game playing. When it begins learning to play a new game, Hoyle relies on general game knowledge. This knowledge may be costly to apply, and because it is more general, can provide only limited power. As Hoyle gains familiarity with the game through play, however, it acquires game-specific knowledge that is inexpensive and accurate. The current paper focuses on the learning and subsequent application of spatially-oriented, visually-perceived patterns. Hoyle monitors the occurrence and importance of an evolving pattern knowledge base for each game it plays, and

over time places increasing importance on useful patterns. Eventually, particular game-specific patterns can replace some general game knowledge as Hoyle's primary decision rationales. In this sense, Hoyle represents an interesting middle ground between Morph and METAGAMER: like METAGAMER, Hoyle starts off using general game knowledge, but like Morph it eventually becomes highly reliant on patterns learned through experience.

### 3. CONCLUSION

Many important game-playing issues await research. Efforts in knowledge representation should seek a more general representation for rules, develop a representation for domain-independent strategies, and support a learning program that develops game-specific knowledge as rich as that of a hand-coded program. Much work remains on how to build a system that achieves and demonstrates general game-playing skill, and yet is highly competitive. This will require the automation of management functions normally performed by humans, including features, search methods, time and resource allocation, efficiency concerns, choice of data structures, combinatorial analysis, and test set generation.

The papers in this issue illustrate how interesting, productive, and wide-ranging work in game playing can be. Further progress, elaboration and synergy among these methods and the cited issues should bring AI closer to its goals. Correspondence with the editors and authors of the papers is welcome.

### REFERENCES

- DE GROOT, A. 1965. *Thought and Choice in Chess*. The Hague: Mouton.
- EBELING, C. 1986. *All the Right Moves: A VLSI Architecture for Chess*. Ph.D. Dissertation, Computer Science Department, Carnegie-Mellon University.
- EPSTEIN, S. L., AND LEVINSON, R. 1994. AAAI93 Fall Symposium Reports - Games: Planning and Learning. *AI Magazine* 15(1):14–15.
- HALL, M. R., AND LOEB, D. E. 1992. Thoughts on programming a diplomat. In VAN DEN HERIK, H., AND ALLIS, L., *Editors.*, *Heuristic Programming in Artificial Intelligence 3: The Third Computer Olympiad*. Ellis Horwood.
- LEVINSON, R.; HSU, F.-H.; MARSLAND, T. A.; SCHAEFFER, J.; AND WILKINS, D. E. 1991. Panel: The role of chess in artificial intelligence research. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*.
- REITMAN, J. 1976. Skilled perception in go: Deducing memory structures from inter-response times. *Cognitive Psychology* 8:336–356.
- TESAURO, G. 1994. TD-Gammon, A Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation* 6(2):215–219.
- VON NEUMANN, J., AND MORGENSTERN, O. 1944. *Theory of Games and Economic Behavior*. Princeton University Press.