ICCA/ ICGA Journal Database

WWW: <http://supertech.lcs.mit.edu/~heinz/iccaj_db/>

The *ICCA/ICGA Journal* is the quarterly publication of the ICCA, the *International Computer-Chess Association* (WWW: <http://www.dcs.qmw.ac.uk/~icca/>). Beside peer-reviewed scientific articles, the journal carries literature reviews and detailed reports about interesting events related to computer chess and other computer strategy games.

This database presents the abstracts of scientific articles and detailed tables of contents for the *ICCA/ICGA Journal* from Vol. 16, No. 1 onwards. The online version also features the full editorials of all issues covered. Please enjoy – any comments welcome!

Cumulative Table of Contents

Jos W. H. M. Uiterwijk (WWW: <http://www.cs.unimaas.nl/~uiterwyk/>) actively maintains a cumulative table of contents for the *ICCA/ICGA Journal* from Vol. 6, No. 3 onwards. It lists all scientific articles, notes, and reviews published in the journal since August 1983, sorted alphabetically by first author names. The table is available online in different document formats (HTML, PDF, PostScript, WordPerfect 7.0) from the following URLs.

http://www.cs.unimaas.nl/~uiterwyk/ICCA_J.toc.htm

http://www.dcs.qmw.ac.uk/~icca/ICCAJtoc.htm

Creation History of the ICCA/ ICGA Journal Database

The staff of the *ICCA/ICGA Journal* kindly provided the raw ASCII texts for the database without which the whole project would have been impossible. Heiner Marxen (WWW: <http://www.drb.insel.de/~heiner/>) wrote several shell scripts to convert the raw ASCII texts into a LaTeX- and LaTeX2HTML-suitable format designed by Ernst A. Heinz (WWW: <http://supertech.lcs.mit.edu/~heinz/>). Both Heiner and Ernst also performed much hand-editing to tune the final appearance of the database presentation.

ICCA Journal: Volume 16 (1993)

ICCA Journal, Volume 16: Number 1 (March 1993)

(Data still missing ...)

ICCA Journal, Volume 16: Number 2 (June 1993)

Three Positions

Mikhail M. Botvinnik

[16(2):71–75] (Text still missing ...)

The Guard Heuristic: Legal Move Ordering with Forward Game-Tree Pruning

Y.-F. Ke and T.-M. Parng

[16(2):76–85] (Text still missing ...)

Thompson: Quintets with Variations

The Editors

[16(2):86–90] (Text still missing ...)

On Telescoping Linear Evaluation Functions

Ingo Althöfer

[16(2):91–93] (Text still missing ...)

ICCA Journal, Volume 16: Number 3 (September 1993)

DISTANCE: Toward the Unification of Chess Knowledge

Robert Levinson and Richard Snyder

[16(3):123–136] This article suggests a new approach to computer chess. A graph-theoretic representation of chess knowledge, uniformly based on a single mathematical abstraction, DISTANCE, is described. Most of the traditional forms of chess knowledge, it is shown, can be formalized in this new representation. In addition to comparing this approach to others, the article gives some experimental results and suggests how the new representation may be unified with existing approaches.

Null Move and Deep Search: Selective Search Heuristics for Obtuse Chess Programs

Christian Donninger

[16(3):137–143] This article describes in detail a selective search heuristic which uses a null-move approach recursively. A variety of empirical data, ranging from tournament results against strong human players to special test positions, are presented. These results do not falsify the hypothesis that the heuristic should be considered as a serious candidate for controlling the search process in a chess program.

Most modern microcomputer chess programs use a mixed search strategy, consisting of a brute-force part to avoid shallow tactical blunders and a selective part designed to increase the efficiency of the search at greater search depths. The second part of our heuristic amounts to extending the search in forced positions, especially near the horizon.

The Bratko-Kopec Test Recalibrated

Shawn Benn and Danny Kopec

[16(3):144–146] (Text still missing ...)

A Test Suite for Chess Programs

Kevin J. Lang and Warren D. Smith

[16(3):152–161] We describe a suite of about 5500 test positions for testing chess-playing programs. The positions are mostly unoriginal and were optically scanned from chess books. The suite is in a number of files, each file being thematic by difficulty, tactical or positional content and others. A procedure for employing the tests is suggested and the authors describe how feedback by testees will allow them to expel the remaining errors. The suite aims at being diverse and of such difficulty that no player, human or machine, will score as much as 80 percent.

ICCA Journal, Volume 16: Number 4 (December 1993)

(Data still missing ...)

ICCA Journal: Volume 17 (1994)

ICCA Journal, Volume 17: Number 1 (March 1994)

Random Evaluations in Chess

Don F. Beal and Michael C. Smith

[17(1):3–9] This paper reports experiments using random numbers as "evaluations" in chess. Although this results in random play with a *depth*−1 search, it is found that strength of play rises rapidly when the depth of lookahead is increased. This counter-intuitive result is discussed and its implications for game-playing are given.

Potential Applications of Opponent-Model Search. Part 2: Risks and Strategies

Hiroyuki Iida, Jos W. H. M. Uiterwijk, H. Jaap van den Herik, and I. Samuel Herschberg

[17(1):10–14] (Text still missing ...)

Learning Patterns for Playing Strategies

Eduardo Morales

[17(1):15–26] We describe a first-order inductive framework, PAL, capable of learning chess patterns from a combination of two sources, viz. general-purpose knowledge about chess and simple example descriptions supplied to it. It is believed to be the first time that useful chess patterns have been so learned. In order to establish that patterns so learned are applicable in play, a simple playing strategy for the King and Rook against King (KRK) endgame has constructed with patterns learned by PAL. A sketch of PAL's limitations and of first-order inductive frameworks' limitations in general is given; in chess-like domains, restrictions are essential for limiting the number of clauses required for induction and for guiding the search for them. Conclusions are given about the state of pattern learning achieved and achievable with presently available means.

Estimating Asymmetry and Selectivity in Chess Programs

Javier Ros Padilla

[17(1):27–30] The present note represents an attempt to extract information from chess programs to the source code of which one has no access. The information studied is on asymmetry and selectivity, following a suggestion by Ingo Althöfer (1992). For asymmetry, some computer games with a fixed depth of search, N, were observed. The principal variation is recorded for every move and, after this, the game is analyzed at depth $N-1$. For every position in these games, three moves are compared: (1) the second ply of the principal variation of the preceding move; (2) the move considered best at depth $N-1$; (3) the move played at depth N.

A la Recherche du Temps Perdu: "That was easy"

Christian Donninger

[17(1):31–35] (Text still missing ...)

ICCA Journal, Volume 17: Number 2 (June 1994)

Self-Annotating Elementary Endgames

Rainer Seidel

[17(2):51–62] A study on elementary endgames of the lone-king type is based on a theory that satisfies formal requirements as well as being intuitively meaningful. Two endgame domains, KRK and a significant component of KBBK, have been fully worked out and constitute operational programs. Owing to their adequate basis in theory, the programs are self-annotating to the extent that even a tutorial comment can be produced for any starting position in the domain and for any subsequent enemy response. The hierarchy of its chess knowledge is sketched syntactically and semantically; a generalized procedure for move generation is exhibited, as is the principal schematic of the KBBK program. It is suggested that the knowledge so gained may be extensible to the analyses of higher-order endgames.

Proof-Number Search and Transpositions

Martin Schijf, L. Victor Allis and Jos W. H. M. Uiterwijk

[17(2):63–74] Proof-number search (pn-search) has been successfully applied to games such as connect-four, qubic and go-moku. Although pn-search is a game-tree search algorithm, minor modifications allow its application to the acyclic-graph representation of these games. To apply pn-search to cyclic-graph representations, e.g., of chess, several obstacles must be overcome, which have been the subject of our investigations. This article describes the application of pn-search to trees, acyclic graphs and cyclic graphs. Experiments on chess, using four different pn-search variants, indicate the suitability of pn-search for solving checkmate problems. The results show that pn-search adapted for cyclic-graph representations outperforms its tree and acyclic-graph counterparts.

Weak Zugzwang: Statistics on some Chess Endgames

Ingo Althöfer and Bernhard Walter

[17(2):75–77] Recently, the concept of Zugzwang has gained some attention in computer-chess research (Roycroft, 1990; Jansen, 1992a, 1992b). In this note, we present some statistics on weak Zugzwang in the four chess endgames KQK, KRK, KBBK and KBNK. The traditional definition of Zugzwang is based on the won-drawn-lost value of a position on the board. Zugzwang occurs when that value for the side-to-move is less favourable than it would have been if his opponent were to move. There are exactly three different categories (Roycroft, 1990).

Traditional Zugzwang requires a drastic change in value, such as a winning position being a draw for player A when A is forced to move first. Less drastically, we may define weak Zugzwang if the distance-to-mate for player A is larger when player A is forced to move first. Note that this definition makes sense even if the game is always a win for player A. We give a simple example from the endgame KRK, namely Kb6, Rc6 for White and Kb8 for Black. If Black is to move, he will lose in one move (0. ... Ka8 1. Rc8 mate). If White is to move, it will take two moves to win (for instance 1. Rc7 Ka8 2. Rc8 mate). For every position P on the board which is a win for White independently of who plays first we define a number $k(P)$: "k(P) := (distance-to-mate if Black is to move) minus (distance-to-mate if White is to move)" Then P is a position with weak Zugzwang if $k(P)$ is negative. In our example, we have $k(P) = -1$.

More, and More Perfect Prose

John Nunn

[17(2):78–80] (Text still missing ...)

ICCA Journal, Volume 17: Number 3 (September 1994)

Experiments in Distributing and Coordinating Knowledge

C. Paolo Ciancarini

[17(3):115–131] We have built a distributed chess program running on a network of workstations. The program consists of independent advisor processes each using different chess knowledge to evaluate the position; each advisor suggests a move to a coordinating process, which uses a selection policy to choose the move to be played. The program uses the knowledge embedded in a known sequential program, except that it is organised differently in different advisors. Experiments are reported with several different knowledge-distribution and move-selection policies. An encouraging result is that some programs built are significantly stronger than their sequential prototypes when using as few as seven workstations.

Data Compression in Encoding Chess Positions

Bernhard Balkenhol

[17(3):132–140] We sketch an algorithm encoding realistic chess positions by the one-bit answers to a sequence of yes-no questions, where realistic is roughly synonymous with "lightly" to occur in a master game. Test runs suggest that a sequence of 79 bits is required on the average for such a position to be encoded, of which process one example is given.

Fuzzy Numbers as a Tool in Chess Programs

Andreas Junghanns

[17(3):141–148] Fuzzy numbers are proposed as a tool to allow for and possibly diminish the uncertainty in searches. After exhibiting the major operations on such (triples of) numbers, the consequences of their use are discussed, specifically in forward-pruning and optimizing searches. Their application has been shown in a locally developed program, CHESSITZ, and a handful of examples shows up their advantages over some more conventional, non-fuzzy directed searches: fuzzy numbers prove good indicators of risks incurred.

ICCA Journal, Volume 17: Number 4 (December 1994)

Replacement Schemes for Transposition Tables

Dennis M. Breuker, Jos W.H.M. Uiterwijk, and H. Jaap van den Herik)

[17(4):183–193] Almost every chess program makes use of a transposition table, typically implemented as a large hash table. Even though this table is usually made as large as possible, subject to memory constraints, collisions occur. Then a choice has to be made which position to retain or to replace in the table, using some replacement scheme. This article compares the performance of seven replacement schemes, as a function of transposition-table size, on some chess middle-game positions. A two-level table, using the number of nodes in the subtree searched as the deciding criterion, performed best and is provisionally recommended.

Distributed Searches: A Basis for Comparison

C. Paolo Ciancarini

[17(4):194–206] This paper attempts to explore the impact of greatly enhanced computing power on the alpha-beta algorithm, notorious for its high processing requirements. On a limited set of test positions, it explores, under some limitations, the question of how best to distribute the power of multiple processes. The main matters probed in tendency are the vital questions of when to split the search among processes and, once splitting has been decided upon, to which process to split.

Stress is laid on the apparent limitation of speed-up, for which a severe law of diminishing returns soon sets in under any reasonable conditions, even when the best feasible splitting strategy is utilized. It is shown, again in tendency, that fertile further exploration essentially requires architectures with intrinsically low communication and context-switching overheads. A uniform hardware/software platform NETWORK C-LINDA, is presented as essentially free of distortion under carefully stated conditions.

Solution Trees as a Basis for Game-Tree Search

Arie de Bruin, Wim Pijls and Aske Plaat

[17(4):207–218] A game-tree algorithm is an algorithm computing the minimax value of the root of a game tree. Two well-known game-tree-search algorithms are alpha-beta and SSS*. We show that there exists a relation between these algorithms, which are commonly regarded as quite different. Many algorithms use the notion of establishing proofs that the game value lies above or below some bound. We show that this is equivalent to the construction of a solution tree. We discuss the role of solution trees and critical trees in the following algorithms: alpha-beta, PVS, SSS-2 and Proof-Number Search. A general procedure for the construction of a solution tree, based on alpha-beta and Null-Window Search, is given.

ICCA Journal: Volume 18 (1995)

ICCA Journal, Volume 18: Number 1 (March 1995)

Editorial:

The StarTech Massively-Parallel Chess Program

Bradley C. Kuszmaul

[18(1):3–19] The StarTech massively-parallel chess program, running on a 512-processor Connection Machine CM-5 supercomputer, tied for third place at the 1993 ACM International Computer-Chess Championship. Testing the program informally, its rating was over 2400 USCF points. StarTech employs the Jamboree search algorithm, a natural extension of Pearl's Scout search algorithm, to find parallelism in game-tree searches. StarTech's work-stealing scheduler distributes the work specified by the search algorithm across the processors of the CM-5. The program uses one global transposition table shared among the processors.

Two performance measures help in understanding the program's performance: the work performed, W, and the critical-path length, C. The Jamboree search algorithm seems to perform some 2 to 3 times more work than the best serial version. The critical-path length, under tournament conditions, is less than 0.1 percent of the work, yielding an average parallelism of over 1000. The StarTech scheduler achieves actual performance of approximately $T = 1.02 * W/P + 1.5 * C$ on P processors. The critical-path length and work performed can thus be used to tune performance.

A Partial Analysis of Minimaxing Game Trees with Random Leaf Values

Mark Levene and Trevor Fenner

[18(1):20–33] Random minimaxing, introduced by Beal and Smith, is the process of using a random static evaluation function for scoring the leaf nodes of a full-width game tree and then computing the best move using the standard minimax procedure. Their experiments using random minimaxing in chess showed that the strength of play increases with the depth of the lookahead. We investigate random minimaxing combinatorially in order to obtain a theoretical justification for Beal and Smith's experiments. In particular, we show that, with respect to chess, random minimaxing with the depth of lookahead equal to two is "stronger" than the same with its depth equal to unity, under the assumption that a move by the first player is better the more it restricts the second player's choice of moves (i.e., his mobility). We conjecture that these results can be generalized for depths of lookahead greater than two.

ICCA Journal, Volume 18: Number 2 (June 1995)

ProbCut: An Effective Selective Extension of the Alpha-Beta Algorithm

Michael Buro

[18(2):71–76] This article presents a new, game-independent selective extension of the alpha-beta algorithm. Based on the strong correlation between evaluations obtained from searches at different depths, it is shown how the result of a shallow search can be used to decide with a prescribed likelihood whether a deep search would yield a value outside the current search window. In its application to Othello, the technique is shown to be effective in investigating the relevant variations more deeply. It significantly increases the playing strength of an already strong brute-force Othello program.

An Integrated-Bounds-and-Values (IBV) Numeric Scale for Minimax Search

Don F. Beal

[18(2):77–81] Search procedures generally discover bounds as partial results before a final value is obtained. In particular, chess programs typically store such bounds in a hash table, along with any exact results, for every position processed. These values save computing time if the position is encountered again. For chess programs, upper bounds, lower bounds or exact values may occur. This paper shows how upper bounds, lower bounds and exact values can conveniently be represented using a single numeric scale, which enables program code to be slightly simpler, and avoids the necessity of a separate data item to distinguish bounds from exact values.

A Null-Move Technique Impervious to Zugzwang

Stefan Plenkner

[18(2):82–84] In chess programs based on the alpha-beta algorithm, various techniques exist to interrupt and modify the blind working of brute-force searches. These can be classified as search extensions (e.g., check extensions, singular extensions and others) or as forward pruning. One very popular heuristic for forward pruning consists in the idea of making a null move (Beal, 1989). The idea of a null move has been adequately set out by Goetsch and Campbell (1990) and by Donninger (1993). Null moves assume that we are given a position with COLOUR to move while COLOUR is not in check. COLOUR's opponent now makes one more move, which will attempt to let the position, as seen by COLOUR, deteriorate. If the value returned should be greater or equal to beta (as seen by COLOUR), an action is called for: beta can be raised or a beta-cutoff can be applied.

The above rests on the assumption that COLOUR has a move available with a value at least equal to that of the null move. This assumption always holds except in Zugzwang positions, i.e., those in which COLOUR only has moves worsening his position. While these Zugzwang positions are rare in an absolute sense, they are relatively frequent in endgame positions. Since one's opponent gains an advantage in null moves by COLOUR yielding a tempo and, in effect, transferring his tempo to his opponent, it follows that depth-of-search in a null-move search is reduced by some amount, say TEMPO. Should the null-move search lead to pruning, the pruning of the search tree is immense; should it fail to prune, the additional effort is limited due to TEMPO reducing the depth of search. The above argument, it is often claimed, tends to exclude null moves from endgame positions, since the risk of finding oneself in a Zugzwang position and the consequent false evaluation would be unacceptably large.

ICCA Journal, Volume 18: Number 3 (September 1995)

Controlled Conspiracy-Number Search

Ulf Lorenz, Valentin Rottmann, Rainer Feldmann, and Peter Mysliwietz

[18(3):135–147] We present a new conspiracy-number search algorithm (CNS), called Controlled Conspiracy-Number Search (CCNS). The basic steps of any CNS algorithm, the selection, the expansion, and the backup of results have been modified compared to other CNS algorithms. The selection is done by assigning demands, so called cn targets, on the nodes of a game subtree in a top-down fashion. Doing so, a set of leaves is selected in a single action. An expansion is used to check whether a leaf node can fulfil the demands imposed upon it in terms of conspiracy number. A backup subalgorithm uses heuristic information gained from the expansion step in order to prepare the game subtree for the next selection phase.

We have found our algorithm to be stronger than the alpha-beta algorithm in tactical positions. This has been shown by comparison them on a recognized set of test positions. It is able to play reasonably even in non-tactical positions, as shown at the 4th International Paderborn Computer-Chess Championship, where ULYSSES CCN, a program based on the CCNS algorithm, participated in a complete tournament with creditable results. Prospects are that, with every selection-step selecting a set of leaves for expansion, the algorithm will prove well-suited to parallelization.

Combining Knowledge and Search to Yield Infallible Endgame Programs

Wilhelm Barth

[18(3):148–159] A provably correct program for the competition of passed Pawns in the KPKP endgame has been developed, using a method described due to Barth and Barth (1992). The method combines the application of strict rules with alpha-beta search. The rules cover a large part of the positions of the endgame. For each such position they define an interval guaranteed to contain the true value. An alpha-beta search finds the values for all positions not covered and removes uncertainties left by too wide intervals in rule-valued positions. Thus, the method yields a correct evaluation for every position and, furthermore, the rules governing the position or its successors provide some tutorial insight to the user about the reasons behind the evaluations. Moreover, automatic validation assures that mistakes in the rules will be discovered and then can be eliminated in dialogue.

Only a few simple rules are essential to the KPKP algorithm, all of which are well-known to skilful chess-players. Some of them are almost trivial, yet they have to be formulated with cautious precision and to be cast into an algorithmic format. Surprising oddities come to light in the process. The behaviour of the program has been tested by solving many studies and by analyzing some game positions discussed in the literature.

An Examination of the Endgame KBNKN

Steven J. Edwards and the Editorial Board

[18(3):160–167] (Text still missing ...)

ICCA Journal, Volume 18: Number 4 (December 1995)

Tutoring Strategies in Game-Tree Search

Hiroyuki Iida, Ken-ichi Handa and Jos W. H. M. Uiterwijk

[18(4):191–204] Pursuing the idea of a good working knowledge of one's opponent model (OM search), loss-oriented search (LO search) goes beyond OM search in occasionally presenting the opponent with an intentional give-away move. When this model is extended to the tutorial level, a delicate balance must be maintained: give-away moves are allowed under the important proviso that, on a balance of probabilities, they should go unnoticed. It is acknowledged that the model is possibly too detailed to be realistic and rather naively replaces the stochastic quality (does the opponent recognize a give-away move?) by a numerical value. Yet, the paper provides valuable though preliminary results on an idealized opponent and how to tutor him.

Quantification of Search-Extension Benefits

Don F. Beal and Michael C. Smith

[18(4):205–218] This paper considers several search-extension rules and one pruning rule that have been described in the literature. An experiment was performed to see how effective each rule was in isolation, and in various combinations. The experiment was performed on a fixed testset of positions, and results measured using node counts. The emphasis of the work was to make repeatable measurements on well-defined tasks, for future comparison with other search-extension rules. In the test domain chosen, some extension rules were strongly advantageous compared with fixed-depth search, but disadvantageous in combination with others. Notably, singular extensions were strongly beneficial if added to a fixed-depth search, but detrimental if added to a search already using check extensions, recaptures and null moves.

On Barth's "Combining Knowledge and Search to Yield Infallible Endgame Programs"

Steven J. Edwards

[18(4):219–224] [The article by Barth (1995) gave rise to a reply by Steven Edwards, desirous of setting the record straight. Below we publish his comments to which Mr. Barth was given the right to reply. His answer is also reproduced below. All Barth's (1995) diagrams have been republished with the same numbering in Edwards' Section 3 below. – Eds.]

The KPKP Endgame: An Amplification

Wilhelm Barth

[18(4):225] (See explanation provided provided for Edwards' note above.)

ICCA Journal: Volume 19 (1996)

ICCA Journal, Volume 19: Number 1 (March 1996)

The ABDADA Distributed Minimax-Search Algorithm

Jean-Christophe Weill

[19(1):3–16] This paper presents a new method to parallelize the minimax tree-search algorithm. This method is then compared to the Young-Brothers-Wait-Concept algorithm in two implementations: Othello and Chess. Results of tests on a 32-node CM5 are given.

Hybrid Heuristic Search

Andrew N. Walker

[19(1):17–23] We report on some experiments in combining a shallow tree-search with a set of heuristics, implemented as filters, specifically in the KQKR ending. It proves difficult to make progress against a computer database (assumed to have knowledge about the heuristics being used), but the combined filters compare well with strong human play.

An Examination of the Endgame KBBKN

Steven J. Edwards

[19(1):24–31] This note presents the results of the computer generation of the complete KBBKN distance-to-mate/loss database by the program SPECTOR. It is a sequel to an earlier note in this Journal (Edwards and the Editorial Board, 1995). Comments concerning prior computer-assisted investigation of KBBKN can be found in "Retrograde Analysis of Certain Endgames" by Ken Thompson (1986). Thompson's results are for WTM (White to move) and indicate a maximum of 66 moves for White to achieve a won position. Also, about 91.8 percent of KBBKN positions are won for White. The definition of 'won position' is one where White has either achieved a checkmate or has captured the black Knight while obtaining a won position in the successor configuration KBBK. Position verdicts (win/non-win) given by Thompson's KBBKN database as referenced in numerous articles have been confirmed by SPECTOR's results.

A related note by the Editors (1992) of this Journal, "Thompson: All About Five Men", provides some more results about KBBKN and other five-man endgames. A comprehensive analysis of Thompson's KBBKN distance-to-win results can be found in "Ideas on Knowledge Synthesis Stemming from the KBBKN Endgame" (Michie and Bratko, 1987). Of particular interest is a graph which presents a histogram of the distribution of the number of white wins versus the distance-to-win. There is a definite "pinch" or nodal point in the distribution that occurs in the region of 54 moves to win. The same general shape of the distance-to-conversion distribution is also present in SPECTOR's distance-to-mate and distance-to-loss distributions. For the WTM distance-to-mate distribution, this nodal point occurs at mate-in-66. For the BTM distance-to-lose distribution, it occurs at loss-in-67.

ICCA Journal, Volume 19: Number 2 (June 1996)

An Explanation Tool for Chess Endgames Based on Rules

Heinz Herbeck and W. Barth

[19(2):75–82] Moves played by a chess program are often hard to comprehend by a human opponent and the "idea" the program "had in mind" when it chose that specific move is difficult to see: chess programs usually do not provide any explanation of their moves. The rule method for endgames (Barth and Barth, 1991, 1992) uses a combination of expert knowledge with a modified version of alpha-beta-search. The expert knowledge is expressed by rules such as can be easily understood by a human chess-player; therefore they can be used to give the chess-player a blow-by-blow explanation of the program's decisions. In this paper, several methods for generating such explanations for endgame programs are described.

Beware the Bishop Pair

Mark Sturman

[19(2):83–93] (Text still missing ...)

A Parallel Algorithm for Solving Hard Tsume-Shogi Problems

Yasuichi Nakayama, Tadafumi Akazawa, and Kohei Noshita

[19(2):94–98] For the last four years, the power of Tsume-shogi solvers (i.e., programs for solving Japanese-chess problems) has advanced remarkably. Hard problems with 19 to 25 steps are now being attacked by several different types of search algorithms in the hope of solving all but a small set of these problems within a short time. In this paper, we present a parallel algorithm on network-connected distributed UNIX workstations, and show new computing results by solving 100 hard problems. The results confirm that our parallel program can solve most of the problems much more quickly than the best sequential program on a UNIX workstation.

Why Did Kasparov Blink?

Hans J. Berliner

[19(2):99–100] (Text still missing ...)

Natural Developments in Game Research

Hitoshi Matsubara, Hiroyuki Iida, and Reijer Grimbergen

[19(2):103–112] In game programming research there are four interesting and related domains: chess, xiang qi (Chinese chess), shogi (Japanese chess) and go. In this article we will compare chess with shogi, both comparing the rules and the computational aspects of both games. We will see that chess and shogi are very similar, but that there are some important differences that complicate game programming for shogi. Most important difference is the game tree complexity, which is considerably higher than the game tree complexity of chess.

We will then argue that these similarities and differences make shogi a good choice for further research in game programming. Chess will soon no longer be competitively interesting. Xiang qi has a game tree complexity similar to chess, suggesting that the same AI techniques will also be successful in this domain. Go is too risky as a next research target because little is known about the cognitive aspects of the game, which in our view hold the key to developing new techniques. Also in this article, a short history of computer shogi with the results of the latest CSA computer shogi tournament is given. In the appendix a short introduction to the rules of the game is included.

ICCA Journal, Volume 19: Number 3 (September 1996)

Machine Learning in Computer Chess: The Next Generation

Johannes Fürnkranz

[19(3):147–161] Ten years ago the ICCA Journal published an overview of machine-learning approaches to computer chess (Skiena, 1986). The author's results were rather pessimistic. In particular he concluded that "with the exception of rote learning in the opening book, few results have trickled into competitive programs" and that "there appear no research projects on the horizon offering reason for optimism." In this paper we will update Skiena's work with research that has been conducted in this area since the publication of his paper. By doing so we hope to show that at least Skiena's second conclusion is no longer valid.

A Taxonomy of Parallel Game-Tree Search Algorithms

Mark G. Brockington

[19(3):162–174] In the last twenty years, a number of articles and theses have been written that contain innovative parallel game-tree search algorithms. The authors of the parallel algorithms have shown how their work is unique and interesting. In some cases, this has been shown by classifying other algorithms by listing implementation details (Bal and Van Renesse, 1986; Ciancarini, 1994). To the author's knowledge, no attempt has been made to classify the algorithms based solely on their algorithmic properties. A taxonomy would make it easy to ascertain what has and has not been accomplished in parallel game-tree search. The presentation of this type of taxonomy is the main contribution of this paper.

Replacement Schemes and Two-Level Tables

Dennis M. Breuker, Jos W. H. M. Uiterwijk, and H. Jaap van den Herik

[19(3):175–180] This note completes the comparison of the performances of seven replacement schemes. The performances are presented as functions of the transposition-table size. Some 200 chess middle-game and endgame positions have been studied. It turns out that the number of nodes of a subtree is a better estimate for potential savings than the depth of a subtree. A two-level table, using the number of nodes in the subtree searched as the deciding criterion, performs best and is recommended. Previous results based on fewer experiments are confirmed.

An Upper Bound for the Number of Reachable Positions

Shirish S. Chinchalkar

[19(3):181–182] (Text still missing ...)

ICCA Journal, Volume 19: Number 4 (December 1996)

6-Piece Endgames

Ken Thompson

[19(4):215–226] Currently I feel that it is well within modest means to solve endgames with two Kings and four pure pieces. Pawn endgames are still a little out of reach. This contribution outlines an approach for achieving endgame solutions and provides the current status of solving 6-piece endgames. Some code for the construction of endgame databases is given and some pitfalls to be avoided are described.

Multiple Probes of Transposition Tables

Don F. Beal and Michael C. Smith

[19(4):227–233] The vast majority of chess-playing programs make use of a transposition table to store positions and their associated values. Typically (and especially with very fast processors or relatively small memories) only a fraction of the values encountered can be stored. This article examines the performance of a simple multiple-probe scheme, using varying table sizes, on a number of middle-game positions. Multiple probes of the table are shown to perform significantly better than the standard single-probe implementations.

CHE: A Graphical Language for Expressing Chess Knowledge

Christian Donninger

[19(4):234–241] (Text still missing ...)

ICCA Journal: Volume 20 (1997)

ICCA Journal, Volume 20: Number 1 (March 1997)

The Dynamic Tree-Splitting Parallel Search Algorithm

Robert M. Hyatt

[20(1):3–19] This paper describes a high-performance parallel tree-search algorithm that uses Dynamic Tree Splitting (DTS) for searching alpha-beta minimax game trees. The algorithm divides the search tree among several processors on a shared-memory parallel machine. The paper deals with the following topics: (1) the DTS algorithm, (2) analyzing alpha-beta to select split points, (3) performance results of the algorithm, and (4) analysis of the results to see where further improvements might occur.

The Legitimacy of Positions in Endgame Databases

Dietmar Lippold

[20(1):20–28] Current endgame databases provide optimal sequences of moves from a given position to conversion or to mate. They do not investigate whether the given position is legitimate. This contribution considers the database as a model of the particular chess configuration under investigation and attempts to classify the positions in legitimate and illegitimate positions. Assuming a position to be legitimate if it can be reached from the starting position by legal moves only, the question is what means do we need to prove the legitimacy of a position.

For this purpose, we first define three new concepts: initially illegitimate, derivedly illegitimate and isolatedly illegitimate. Then we propose a constructive procedure which enables us to show that a position can be reached. Related to the new concepts we introduce a broad definition of illegitimacy (based on initially illegitimate only) and a narrow definition (taking into account also derivedly illegitimate). For both cases we have examined the 3- and 4-man databases of which results are given. Further research of 5- and 6-man databases is envisaged.

Searching with Uncertainty Cut-offs

Yngvi Björnsson, T. Anthony Marsland, Jonathan Schaeffer, and Andreas Junghanns

[20(1):29–37] A new domain-independent pruning method is described that guarantees returning a correct game value. Even though alpha-beta-based chess programs are already searching close to the minimal tree, there is still scope for improvement. Our idea hinges on the recognition that the game tree has two types of node: those where cut-offs occur, and those that must be fully explored. In the latter case one of the moves is best and yields the subtree value; thus for the remaining alternatives it is enough to show their inferiority. This offers an opportunity for pruning, while introducing some potential for uncertainty in the search process. There are two cases of interest. One considers the immediate alternatives to the Principal Variation itself; here a new safe cut-off is presented. The other is a proposal for an unsafe generalization, one which offers substantial search reduction but with the potential for control of the error probability. Experiments with the new pruning method show some savings in the search.

ICCA Journal, Volume 20: Number 2 (June 1997)

(Data still missing ...)

ICCA Journal, Volume 20: Number 3 (September 1997)

Learning Piece Values Using Temporal Differences

Don F. Beal and Martin C. Smith

[20(3):147–151] This paper describes experiments where we attempt to learn the relative values of chess pieces by the use of temporal difference learning applied to minimax searches. We show that we are able to learn suitable piece values, and that these values perform at least as well as piece values widely quoted in elementary chess books.

On the k-Best Mode in Computer Chess: Measuring the Similarity of Move Proposals

Ingo Althöfer

[20(3):152–165] Some commercial chess programs allow to compute not only the best move but also the k best moves for each position. Most frequently used are k = 2 and k = 3. Unfortunately, sometimes the moves are too similar in such a k-best list, or otherwise stated they do not discriminate among plans. A scheme examining the similarity and eliminating move proposals that are too similar is suggested. A simple approach dealing with this problem is presented.

How DarkThought Plays Chess

Ernst A. Heinz

[20(3):166–176] DARKTHOUGHT is a bitboard-based chess program developed at the University of Karlsruhe that has successfully participated in all world championships since 1995. On a 500 MHz DEC Alpha-21164a with 128 MB RAM, DARKTHOUGHT routinely reaches speeds of 200 K nps (nodes per second) in the middlegame while peaking at over 650 K nps in the endgame.

The article describes the design and inner structure of DARKTHOUGHT. To this end, it presents detailed accounts of the chess engine while elaborating on some of its innovations: rotated bitboards, a fully programmable leaf-node evaluation function, and versatile search parameterization.

Genius-3 Cooked Endgame Studies

Paul Wiereyn

[20(3):177–179] (Text still missing ...)

ICCA Journal, Volume 20: Number 4 (December 1997)

Can Machines Think? Deep Blue and Beyond

Daniel C. Dennett

[20(4):215–223] The question "Can Machines Think?" has always intrigued researchers and philosophers. Last year the question revived when DEEP BLUE played Kasparov, but Kasparov then held his ground. However, recently DEEP BLUE defeated the World Champion, and so the question became really acute.

This article attempts to define a suitable sword-in-the-stone test, equally valid for computers and humanity. The following tests are successively discussed: the Chess Test, the Turing Test, and the Gödel Test. The Chess Test is considered to be obsolete after Kasparov's defeat. The Turing Test is illustrated by providing background on the Loebner Prize Competitions. Its restricted version is a problematic sword-in-the-stone test; its unrestricted version is certainly not a test that any machine is going to pass any time in the foreseeable future. Finally, the Gödel Test is discussed.

Evaluation Tuning for Computer Chess: Linear Discriminant Methods

Thomas S. Anantharaman

[20(4):224–242] The evaluation function of a software version of the DEEP THOUGHT chess program is tuned against a database of master-level games. The agreement between the move chosen by the program and by a human master is maximized. A set of algorithms solving the maximization problem is presented. The trade-off between computational speed and the chance of getting stuck in a local maximum is extensively investigated.

The performance of the chess program with the tuned evaluation function is experimentally measured. The program has played several matches of 500 games each with different settings against a fixed program. The results show that 98% of the performance of the evaluation function can be achieved in about a week by tuning the evaluation function from scratch against the database. If the evaluation tuning is started from the best evaluation used so far, the new evaluation function improves the playing strength by 34 rating points.

Does Deep Blue use Artificial Intelligence?

Richard E. Korf

[20(4):243–245] When DEEP BLUE played Garry Kasparov in February 1996 and May 1997, the extensive IBM Web pages devoted to the site claimed that DEEP BLUE did not use artificial intelligence (AI). I argue that this claim is (1) inaccurate, (2) representative of a wide-spread phenomenon in the field, and (3) ultimately harmful to AI.

ICCA Journal: Volume 21 (1998)

ICCA Journal, Volume 21: Number 1 (March 1998)

A Hypothesis on the Divergence of AI Research

Franz-Günther Winkler and Johannes Fürnkranz

[21(1):3–13] Artificial Intelligence has been conceived as the science of both programming computers to perform intelligent tasks and devising computational models of human reasoning. Originally, both aspects were considered to go hand in hand, but it soon became apparent that AI research was determined to split into an engineering branch and a cognitive branch corresponding to the objectives mentioned. Research in computer chess is a prominent and successful example of this development. We conjecture that the reason for the divergence is a non-linear interaction between the strict algorithmic processing requirements of computational models on the one hand and the associative human knowledge structures on the other hand.

Are there Practical Alternatives to Alpha-Beta?

Andreas Junghanns

[21(1):14–32] The success of the alpha-beta algorithm in game playing has shown its value for problem solving in artificial intelligence, especially in the domain of two-person zero-sum games with perfect information. Still, there are different algorithms for game-tree search which challenge the value of the alpha-beta algorithm. This paper describes and assesses the alternatives proposed according to how they try to overcome the limitations of alpha-beta. We conclude that for computer chess no practical alternative exists, but many promising ideas have potential to change that in the future.

The Significance of Kasparov versus Deep Blue and the Future of Computer Chess

Dennis DeCoste

[21(1):33–42] In this paper we argue that the Garry Kasparov versus DEEP BLUE matches of 1996 and 1997, despite some shortcomings, have had a significant positive impact on computer-chess and artificial-intelligence research. Such matches can be viewed as valuable lessons in how (and how not) to use man-machine competitions to obtain a better focus on applied research and on the development efforts. One of the most important goals is to provide benchmarks (both in terms of development effort and match performance) that more knowledge-intensive (i.e., less brute-force) approaches must surpass to justify their extra complexity.

We will briefly summarize some of the latest developments in computer-chess research and highlight how our work on a program called CHESTER tries to build on them to provide such justifications. Using our CHESTER work as an example, we propose a possible shift in emphasis for the computer-chess community: away from tournament performances (nowadays a short-term goal) and more toward treating the chess domain as a valuable testbed for research on emerging large-scale data-mining and machine-learning techniques. The successes of DEEP BLUE, despite legitimate questions of validity and scope, have in some sense freed (and perhaps even compelled) us to refocus on such longer-term goals.

ICCA Journal, Volume 21: Number 2 (June 1998)

Extended Futility Pruning

Ernst A. Heinz

[21(2):75–83] This article presents a new selective pruning technique for alpha-beta based game-tree search in computer chess, called extended futility pruning. It builds on ideas of both razoring and normal futility pruning at frontier nodes (depth = 1), and it cuts complete branches of the search tree at pre-frontier nodes (depth = 2) according to solely static criteria at the respective nodes. Hence, extended futility pruning performs true forward pruning.

Although extended futility pruning is theoretically unsound, extensive experiments with our master-strength chess program DarkThought show that it works markedly well in practice. Even at fixed search depths, extended futility pruning exhibits hardly any loss of tactical strength while shrinking the search trees by 10 to 20 percent on average as compared with normal futility pruning.

Moreover, extended futility pruning combines nicely with a conservatively limited variation of razoring. It reduces the search trees at fixed search depths of more than 10 plies by an additional 5 to 15 percent on average. Finally, we have observed that the presented pruning schemes scale very well with the search depth, and reap ever more benefits at higher depths.

Experiments in Parameter Learning using Temporal Differences

Jonathan Baxter, Andrew Tridgell, and Lex Weaver

[21(2):84–99] This paper deals with the problem of automatically learning evaluation-function parameters. In particular, we describe some experiments in which our chess program KnightCap learnt the parameters of its evaluation function using a combination of temporal difference learning and on-line play on FICS (Free Internet Chess Server) and ICC (Internet Chess Club). The main success reported is that KnightCap went from a blitz rating of 1650 to a blitz rating of 2150 in just three days and 308 games. We provide details of our learning algorithm, details of KnightCap, and some reasons for KnightCap's rapid improvement in playing strength.

Learning to Play Chess Selectively by Acquiring Move Patterns

Lev Finkelstein and Shaul Markovitch

[21(2):100–119] Human chess players do not perceive a position as a static entity, but as a collection of potential actions. Moreover, they seem to be able to follow promising moves without considering all the alternatives. This contribution investigates the possibility of incorporating such capabilities into chess programs. We describe a methodology and a language for representing move patterns. A move pattern is a structure consisting of a board pattern and a move that can be applied in that pattern. Move patterns are used for selecting promising branches of a search tree, hence allowing for a narrower and therefore deeper search tree. Move patterns are learned during training games and are stored in a hierarchical structure to enable fast retrieval. The paper describes algorithms for learning, storing, retrieving, and using the move patterns.

ICCA Journal, Volume 21: Number 3 (September 1998)

Games: The Next Challenge

Jaques Pitrat

[21(3):147–156] After the successes of many chess-playing programs, one may wonder whether it is necessary to keep on developing chess programs. Perhaps it would be better to realize programs for games such as Go, where the best programs are still weak. In this contribution, we suggest that it is time to shift focus on general game-playing systems and to realize them. Such systems will receive the specification of a new game. Subsequently they will analyze the specifications so that they will be able to play the game reasonably well. We will see how one can define such a specification and how a system may study it in order to extract useful knowledge. The problem is ultra difficult, but also highly challenging for AI. Obviously, as long as we call a system intelligent when it solves only one problem we must face the critique that most of the intelligence is in the brain of the author and little in the program itself.

Efficient Interior-Node Recognition

Ernst A. Heinz

[21(3):157–168] This article re-examines the implications of interior-node recognition when focussing on its efficient yet seamless integration into modern chess programs. After a thorough discussion of the fundamental principles, we reveal various problems related to the practical application of interior-node recognition. Subsequently, we present an implementation framework for recognizers that solves all known problems and has already proven its practical viability in our high-speed chess program DarkThought.

Among others we introduce the new concept of material signatures which allow for the quick and easy classification of chess positions into different categories of material distribution. By including material signatures in the internal representation of the chess-board, they can be updated incrementally during the execution of moves. This makes the computation of material signatures very cheap in practice.

Synthesis of Chess and Chess-like Endgames by Recursive Optimisation

Kevin P. Coplan

[21(3):169–182] A new algorithm is presented for synthesising correct and optimal game-playing programs directly from the specifications. The programs deal with chess-like endgames. The algorithm is based on the principle of self-optimisation allowing the program-construction process to be time-feasible. A LISP implementation of the synthesising algorithm has generated a program for playing optimally a three-piece endgame on a specialised board. The program established the general KB/RK configuration as a win for White, and constructed a maximin of 39 moves; moreover it provided an analysis of the maximin position. The experimental results are presented. Finally, the relationship to other game-playing work is discussed, and the applicability of the algorithm to more general domains is considered.

Planning a Strategy in Chess

Jan van Reek, Jos W. H. M. Uiterwijk, and H. Jaap van den Herik

[21(3):183–192] Planning plays a major role in chess. The range of planning may be short, medium, or long. Short-term planning is dominated by tactics; medium-range planning is usually identified with positional play, and long-term planning is assumed to require human interpretation for the recognition of strategic patterns. In this note we investigate the differences among the three types of planning. We have tested three programs (Crafty, M-Chess 6, and M-Chess 7) on a set of 24 positions. M-Chess 7 scores faultlessly for tactics and positional play, and 73 percent for strategy. If the hardware and software are further improved, the computer analysis will become sovereign in long-term planning too.

ICCA Journal, Volume 21: Number 4 (December 1998)

Chess Endgames and Neural Networks

Guy McC. Haworth and Meel Velliste

[21(4):211–227] The existence of endgame databases challenges us to extract higher-grade information and knowledge from their basic data content. Chess players would like simple and usable endgame theories if such holy grail exists; endgame experts would like to provide such insights and be inspired by computers to do so. Here, we investigate the use of artificial neural networks (NNs) to mine these databases and we report on a first use of NNs on KPK. The results encourage us to suggest further work on chess applications of neural networks and other data-mining techniques.

DarkThought Goes Deep

Ernst A. Heinz

[21(4):228–244] This contribution follows up on the work by Hyatt and Newborn (1997) who let the program CRAFTY "go deep". We repeated their experiment with our own chess program DARKTHOUGHT and obtained similar experimental results. Both experiments provide strong empirical evidence for the surprising observation that even at high search depths of 11 to 14 plies modern chess programs steadily discover new best moves in still 16 percent of all searches on average.

Moreover, the experiments do not reveal any conclusive trend towards fewer new best moves at search depths beyond 14 plies. Hence, the available experimental results do not really fuel the intuitive notion that such "changes of mind" taper off continuously with increasing search depths. If at all, only the behaviour of DARKTHOUGHT with a drop to 13.7 percent new best moves on average in iteration 14 hinted at decreasing "changes of mind" for search depths of 15 plies or more.

Additionally gathered data about the 14-ply searches of CRAFTY and DARKTHOUGHT allowed us to study the behaviour of both programs in greater detail. This led to the astonishing finding that regardless of the actual search depth a sizeable 30 to 50 percent of all new best moves on average represented "fresh ideas" which the programs never deemed best before. The finding adds support to Newborn's (1985) hypothesis about the playing strength of chess programs. Furthermore, the additional numbers educated us about continuing odd/even instabilities of modern chess programs. These instabilities decreased notably only at high search depths of 9 to 14 plies in positions with reduced material as they mostly occur in endgames and late middle games.

ICCA Journal: Volume 22 (1999)

ICCA Journal, Volume 22: Number 1 (March 1999)

Book Learning – A Methodology to Tune an Opening Book Automatically

Robert M. Hyatt

[22(1):3–12] The paper describes some experiments in book learning by the program CRAFTY, a computer-chess program developed by the author. CRAFTY plays on various Internet chess servers, and as a result, typically plays over 20,000 games per year. A favourite tactic of many human chess players is to find a weak book line that the program follows repeatedly. They play that opening over and ove until they find a way to win the game. Thereafter they replay this game many times. Previously the author would simply edit the book by hand when such cooks were found, to prevent the program from playing these openings time after time. Recently the author has developed an approach that automates this book editing or book tuning by letting CRAFTY determine which openings are good and which are bad, and adjusting its book selections accordingly.

A Pattern-Oriented Approach to Move Ordering: The Chessmaps Heuristic

Kieran R. C. Greer, Piyush C. Ojha, and David A. Bell

[22(1):13–21] A move-ordering method is presented that orders most moves by using a neural network in the form of the chessmaps heuristic. The neural network is trained to order sectors, or areas of the chessboard, depending on the territorial control of each side. Moves are then ordered depending on which sectors they influence. The complete move-ordering method takes account of immediate tactical threats, in the form of forced or capture moves, before the positional evaluation of the chessmaps heuristic is used.

The chessmaps heuristic has extracted some intelligent chess information from the examples it has been trained on. In particular, it has learnt a very basic strategy. When White controls the whole chessboard, the chessmaps heuristic suggests attacking sectors for White. When Black controls the whole chessboard, the chessmaps heuristic suggests defensive sectors for White. When the amount of control is equal, the chessmaps heuristic suggests attacking sectors where defensive factors play a part. So, it looks like the program knows something about when to attack or defend. The heuristic has also learnt some notions of centralisation.

Because this AI approach is relatively simple, it is suitable when supported by a substantive brute-force search. Preliminary test runs comparing this move-ordering method with other heuristic combinations involving the killer and history heuristics are encouraging, since our move-ordering method searched fewer nodes.

Endgame Databases and Efficient Index Schemes for Chess

Ernst A. Heinz

[22(1):22–32] Endgame databases have become an integral part of modern chess programs during the past few years. Since the early 1990s two different kinds of endgame databases are publicly available, namely Edwards' so-called tablebases and Thompson's collection of 5-piece databases. Although Thompson's databases enjoy much wider international fame, most current chess programs use tablebases because they integrate far better with the rest of the search. For the benefit of all those enthusiasts who intend to incorporate endgame databases into their own chess programs, this note describes the index schemes of Edwards' tablebases and Thompson's databases in detail, explains their differences, and provides a comparative evaluation of both.

In addition we introduce new indexing methods that improve on the published state of the art and shrink the respective index ranges even further (especially for large databases and endgames with Pawns). This reduces the overall space requirements of the resulting databases substantially. We also propose several solutions for the problem of potential en-passant captures in endgame databases with at least one Pawn per side. Shortly after the initial submission of our original text for this note, Nalimov independently applied similar techniques as ours and even more advanced index schemes to his new tablebases which are now publicly available on the Internet, too. We briefly put his work into perspective as well.

ICCA Journal, Volume 22: Number 2 (June 1999)

Exhaustive and Heuristic Retrograde Analysis of the KPPKP Endgame

Christoph Wirth and Jürg Nievergelt

[22(2):67–80] Although the techniques for constructing chess endgame databases are well understood, it is a permanent challenge to extend the size of the endgames that can be computed with the resources currently available. Ken Thompson's pioneering attack on 5-piece endgames was limited to those with at most one Pawn. Using our parallel retrograde analysis program RetroEngine on a variety of hardware platforms, we have constructed all 111 databases necessary to compute the KPPKP endgame, i.e., King and two Pawns vs. King and Pawn.

Furthermore, we present a new approach called heuristic retrograde analysis that trades accuracy for space and time, i.e., it reduces the space and time needed to solve an endgame while accepting a small error rate. Experiments for the KPPKP endgame yielded reductions in the required space and time by factors of more than 50. The penalty incurred is that a (computer) player using the heuristic database plays suboptimally in less than 4 percent of the positions submitted.

The KPPKP database contains a wealth of interesting positions. We present 15 endgame studies of unconventional design and surprising difficulty.

Knowledgeable Encoding and Querying of Endgame Databases

Ernst A. Heinz

[22(2):81–97] Modern chess programs quickly become I/O-bound if they probe their external endgame databases not only at the root node but also at interior nodes of the search tree. This tendency increases at faster search speeds if the I/0 speed does not scale accordingly. Hence, the foreseeable trends in CPU and I/0 technology will not improve the probing but rather aggravate it. Instead of resorting to "quick and dirty" fixes such as stopping the accesses at a specific depth, our chess program DARKTHOUGHT probes its 3-piece and 4-piece endgame databases everywhere in the search tree at full speed without any I/0 delays. It does so by loading the entire databases into the main memory of its host system.

To this end, we introduce a new domain-dependent encoding technique that reduces the space consumption of all 3-piece and 4-piece endgame databases to roughly 15 Mbytes overall. Apriori studies of Edwards' publicly available distance-to-mate tablebases provided the necessary feedback for our so-called knowledgeable encoding. We rely on the algorithmic recognition of rare exceptional endgame positions in order to achieve a compact representation of the stored data. The knowledgeable approach enables chess programs to preload all 3-piece and 4-piece endgame databases even on cheap personal computers with low memory capacities starting at 32 MBytes of RAM.

Toward Opening Book Learning

Michael Buro

[22(2):98–102] In this contribution an opening-book framework for game-playing programs is presented. The research is motivated by the aspiration to play a sequence of games successfully, i.e., to avoid losing a game twice in the same way. We show how reasonable move alternatives can be found to deviate from previous lines of play. Variants of the algorithm are used by several of today's best Othello programs. They allow the programs to extend their opening books automatically.

Computer Go: A Research Agenda

Martin Mueller

[22(2):104–112] (Text still missing ...)

ICCA Journal, Volume 22: Number 3 (September 1999)

Adaptive Null-Move Pruning

Ernst A. Heinz

[22(3):123–132] General wisdom deems strong computer-chess programs to be "brute-force searchers" that explore game trees as exhaustively as possible within the given time limits. We review the results of the latest World Computer-Chess Championships and show how grossly wrong this notion actually is. The typical brute-force searchers lost their dominance of the field around 1990 when the null move became popular in microcomputer practice. Today, nearly all world-class chess programs apply various selective forward-pruning schemes with overwhelming success.

To this end, we extend standard null-move pruning by a variable depth reduction and introduce what we call adaptive null-move pruning. Quantitative experiments with our chess program DARK-THOUGHT show that adaptive null-move pruning adds a new member to the collection of successful forward-pruning techniques in computer chess. It preserves the tactical strength of DARKTHOUGHT while reducing its search effort by 10 to 30 percent on average in comparison with standard null-move pruning at search depths of 8 to 12 plies. Moreover, adaptive null-move pruning is easy to implement and scales nicely with progressing search depth.

Efficient Approximation of Backgammon Race Equities

Michael Buro

[22(3):133–142] This article presents efficient equity approximations for backgammon races based on statistical analyses. In conjunction with a 1-ply search the constructed evaluation functions allow a program to play short races almost perfectly with regard to checker-play as well as doubling cube handling. Moreover, the evaluation can naturally be extended to long races without losing much accuracy.

ICCA Journal, Volume 22: Number 4 (December 1999)

KQQKQQ and the Kasparov-World Game

Eugene V. Nalimov, Christoph Wirth, and Guy McC. Haworth

[22(4):195–212] The 1999 Kasparov-World game for the first time enabled anyone to join a team playing against a World Chess Champion via the web. It included a surprise in the opening, complex middle-game strategy and a deep ending. As the game headed for its mysterious finale, the World Team requested a KQQKQQ endgame table and was provided with two by the authors. This paper describes their work, compares the methods used, examines the issues raised and summarises the concepts involved for the benefit of future workers in the endgame field. It also notes the contribution of this endgame to chess itself.

Rotated Bitmaps, a New Twist on an Old Idea

Robert M. Hyatt

[22(4):213–222] This paper describes some developments related to using bitmaps (64-bit integers using one bit for each square of the chessboard). In late 1994, after the ACM computer-chess event in Cape May, New Jersey, I decided to embark on a complete replacement chess program for CRAY BLITZ. I was interested in using the bitmap approach mentioned by Slate and Atkin in CHESS 4.X to determine for myself whether this approach was suitable for chess or not. In developing this new program, the concept of rotated bitmaps was developed, and this turned out to be what was needed to make this type of data structure produce reasonable performance.

Learning Piece-Square Values Using Temporal Differences

Don F. Beal and Michael C. Smith

[22(4):223–235] This paper reports on the results obtained from using improved temporal difference learning methods to learn piece-square weight sets for use in a chess program. The learning takes place solely from self-play, starting from zero values. A comparison is made between values learnt from piece weights only, and piece weights plus positional weights. The weight sets obtained, when displayed as grey-scale diagrams matching chessboards, can be visually seen to correspond to various items of simple chess knowledge of the type found in elementary chess books, and regarded as basic information for beginning chess players. The paper also considers the effect of the squashing function used to map evaluations into probabilities-to-win.

The Kasparov-World Match

Peter Marko and Guy McC. Haworth

[22(4):236–238] (Text still missing ...)

Two Strategic Shortcomings in Chess Programs

Jan van Reek, Jos W. H. M. Uiterwijk, and H. Jaap van den Herik

[22(4):239–240] (Text still missing ...)

ICGA Journal: Volume 23 (2000)

(Data still missing ...)