A Test Suite for Chess Programs

Kevin J. Lang Warren D. Smith *
wds@research.NJ.NEC.COM

September 1993

Abstract — **We describe a suite of ≈ 5500 test positions for testing chess playing programs. They are available as pub/wds/ChessTest.tar.Z by anonymous ftp to external.NJ.NEC.COM. Almost all of these positions are unoriginal and were obtained by scanning in diagrams from chessbooks with an optical scanner. Gnuchess 4.0, at one minute per move on a 50 MHz MIPS R4000, scores 16-71% on our testfiles. We describe the software we wrote to accomplish the scanning task. If you take the test, please send us (Email: wds@research.NJ.NEC.COM) the results, the testee's name and estimated USCF rating, and notify us of any corrections in the test set.**

1 Introduction

The computer chess community has long had a need for a comprehensive test suite of chess positions for the purposes of debugging and evaluating chess playing programs. Such a suite can also be useful (if it is comprehensive enough) for finding out what areas of play one's program is weak in, so that one may then improve the program in those areas. A third use of test suites was apparently invented by the authors (D. Dailey and L. Kauffman) of the program "Socrates II" which recently won the 23rd ACM computer chess championship. This is to take a set of test positions which you know your program is capable of solving (a subset of our set might be used) and to then postulate that the strength, in rating points, of your program is a monotonic function of the runtime it requires to solve the problems in this set. One then tunes the program's selective search heuristics to reduce this runtime.

1.1 Previous test suites

There have been some previous efforts to devise chess test suites. The 300 positions from Reinfeld's "Win at Chess" [33] have been used by several authors (for example, see [3]), and are good for testing sharp tactical play (which computers excel at), but little else. CHIPTEST with singular extensions [3] scored 297 out of a possible 299 on this set, leading to claims that the best computers have "outgrown" it.

I. Bratko and D. Kopec [8] devised the 24-position "Bratko-Kopec test," some of whose positions also test "positional" play. But 24 positions is too few.

Berliner, Kopec and Northam [7] started a large effort to produce a $\approx$ 500-position test set, and as part of their effort produced an ambitious "taxonomy," or subject classification scheme, classifying all chess themes into roughly 82 categories. Unfortunately (according to a private communication from Berliner), this effort seems to have produced only about 20 actual chess positions before its authors quit working on it. The reason for this failure appears to have been the very high standards of quality and accuracy that they set for themselves.

We have included the above three tests in our test suite.

Fidelity Electronics was once reported to have a 4000-position test suite. Heuristic Software has another test suite. These have not been released to public.

Several authors have used the entire set of positions from grandmaster games as a test suite, where the "right move" is assumed to be whatever was actually played. But this approach has obvious weaknesses, namely that (presumably) it is incapable of testing players who are stronger than the grandmasters who played those games, and also, there are plenty of positions in grandmaster games in which there is another "correct" move.

Finally, Nielsen [29] describes a test set he has devised. The description was rather vague (for example, he doesn't say how many positions are in it, nor where they came from). Anyway, Nielsen assigned points to each problem in some unknown manner in an attempt to make your total point score on the (timed) test be a close approximation to your (Swedish) chess rating. Nielsen also gives you up to 40 extra points, *regardless* of your performance on the test questions, just for having a large opening book, and up to 10% extra score for knowing about transpositions. According to data he supplies comparing scores on his test for 46 computers with their Swedish ratings, the standard

*Both authors are at NEC Research Institute, 4 Independence Way, Princeton, NJ 08540.

deviation in his prediction is about 80 points, but in the outlier case of Deep Thought, Nielsen's prediction was 379 points too low, leading to the suspicion that his point system only works well (as a Swedish rating predictor) for testees whose ratings lie in a narrow range. We have never seen Nielsen's set, but it certainly sounds interesting, and the fact that he has administered it to a large number of computers makes it plausible that the quality of the test questions is high.

1.2 What one wants in a chess test suite

What one really wants in a test suite for a computer program is not only a chess position, but also what the best move is for that position. Positions with a unique best move are naturally preferred for this purpose, along with an explanation of *why* it is the best move, so that cause of the program's failure to find it (or the reason that the test question is bogus!) may be determined. The position and the best move need to be available in electronic form, but the explanation need only be available in human-readable form and indeed need only consist of a page reference in a widely available chess book. [1]

Another idea is to have positions in which there is some plausible-looking but incorrect move, and then one asks whether the testee understands that it should *not* make that move. We have included some of these positions in our test suite.

How large should the test be? We believe it should be big, comprehensive, and diverse. Our test set currently contains 5551 positions from about 25 sources. Taking this test would require 90 hours, even for a testee who is expected to solve questions at the (fast) rate of one question per minute. The point is that by including so many questions, one becomes free to focus on specific subject areas or difficulty levels by devising appropriate subsets of the questions.

Many great chess authors have been working for years to create, or find in tournament games, positions which are of particular pedagogical interest, or of particular interest for testing purposes. They've also produced their own taxonomies of chess themes ([7] were not the first to think of this!) and some sources have attempted to provide complete coverage of their taxonomy. Therefore, we do not need to create test positions ourselves (a very difficult task to do without errors), nor do we need to make taxonomies, nor do we need to strive for complete coverage of a taxonomy. This has already been done for us by all the chess writers and analysts of the world, with better accuracy and insight, and spending more thinking time on the task, than the present authors (who are poor chess players) will ever have. This is not to say that positions in chess books are error-free – they are not. We just think that the task of producing a test set which *is* error free, but which is also difficult enough to present a challenge for even the best chessplayers in the world and at the same time large enough to cover most important aspects of chess play, is too difficult a task.

In practice, therefore, one must be satisfied with the level of difficulty, accuracy and subject coverage that is available in chess books, *plus* a certain amount of extra accuracy can be obtained as follows: as more and more computer programs take the test, reporting their results back to a central location (that is: us! By electronic mail!), it should gradually become clear which test questions have errors.

All we really need to do, then, is to enter chessbooks into the computer.

1.3 What we did

For this purpose, one of us (KJL) wrote a program which uses an optical scanner to read in chess positions from diagrams in chess books. We are not willing/able to give away this program, since it depends on having certain hardware and software unique to our lab, and also since it needs to be retrained on each new chessbook, which at present is a relatively quick, but not fully automated, process. We will, however, tell something about how the program works, see §3.

The other of us (WDS) acquired and scanned in the books and thus created the test suite of chess positions described in §2. This test suite is available by anonymous FTP over the internet as file "`ChessTest.tar.Z`" in directory `pub/wds` at site `external.NJ.NEC.COM`. It contains several thousand chess positions. Notify `wds@research.NJ.NEC.COM` of any corrections.

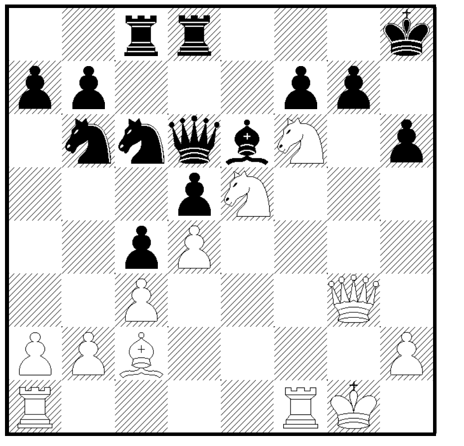
2 The test suite

2.1 Format

We have several ASCII files of chess positions. Each position is in a "Forsyth" format. In this format, each position occupies a *single line* of an ASCII file. A typical entry might be

[1]A different kind of test would require the subject to evaluate a position rather than find the best move. Because one could imagine a fine chessplayer that could not answer such questions, we do not believe that they belong in *tests* of computer chess players, although they might be useful as training data.

`2rr3k/pp3pp1/1nnqbN1p/3pN3/2pP4/2P3Q1/PPB4P/R4RK1/w Qg3g6 Reinfeld.300.1`



which indicates the following position:

White to move Qg3g6. #1/300 test positions from Reinfeld's *Win at Chess.*

Each line has three fields, separated by a "space" character. The first field is the position in "Forsyth" notation (ending with "`/w`" when white is to move, and with `/b` when black is to move). The second field is the recommended move in long-form double algebraic notation.[2] The third field (which may include spaces) comprises the rest of the line, and should indicate either directly or by citation, *why* this is the correct move, plus it can include any comments, analysis, or extra information at all.

Our files also contain lines that are not of this form; the testee may assume that every line in the file which contains exactly 8 "`/`" characters is a test position, while all other lines are comments.

As for castling and en passant, if the king and rook are on their original squares, then one can assume that they have never previously moved. Also, one can assume that there is no en passant opportunity. While these assumptions are nearly always correct, we provide for the exceptional case when they are not as follows. An "`s`" represents a black rook which, despite the fact that it is on its original square, and despite the fact that the black king may be on its original square, has forfeited its castling privileges. A "`o`" represents a black pawn which is capturable next move by en passant. These mnemonics are easily remembered since `o` is adjacent to `p` and `s` is adjacent to `r` alphabetically. (Similarly `S`,`O` are *white*.)

If several moves are listed in the second field (separated by commas, not spaces), then any one of them should be counted as a correct response *unless* it has the punctuation "`?`" in which case it is an incorrect response. (If *every* answer-move listed is "`?`" then *any* move other than these, should be scored as 'correct.') If several moves are listed, but one of them has an "`!`" appended, then although all the listed moves are game-theoretically optimal, the move with the `!` is the quickest and clearest.

2.2 Accuracy

We performed the following "sanity checks" on our data.

1. We checked that all pawns were on the 2-7 ranks (not 1,8).
2. We checked that exactly one king of each color existed.
3. No more than 8 pawns of each color exist, etc.
4. Positions with 3 knights, 3 rooks, 2 queens, or 2 bishops of the same color on the same color squares were flagged as "suspicious" and human-verified.
5. The "right move" is always a legal move.
6. The side not-on-move is not in check.
7. In some cases redundant side-to-move information was provided in the book, which was used to help eliminate errors in deciding whose move it was.[3]

[2] As examples of our notation, some of the other legal moves in the sample position would be written `Ne5xNc6` and `Ne5xPf7+`; a promotion with capture of a knight and giving check might be indicated by `Pe7xNd8=Q+`; a castling move by `Ke1g1C`. En-passant captures similarly have a convenient `E` suffix. One may readily convert our highly-redundant move notation to plain double algebraic (`e5c6`, `e5f7`, `e7d8Q`, `e1g1`) by deleting all characters except for a-h, 1-8 or a character occuring immediately after a '`=`'.

[3] Counting promoted men plus Ps, and checking en passantable P validity, has also been performed for many of our data sets.

Also, the scanning software which created the data in the first place was supervised by a human who was required to verify/correct each piece classification.

These steps sufficed to eliminate most of the egregious clerical errors, but we have no doubt that some remain. It turns out that Murray Campbell and the "Deep Thought" team had previously typed in the equivalent of our `hardmid` test set themselves. Comparing the two sets, Campbell found 7 scanning errors in the Lang-Smith version (mostly missing white pawns on dark squares) and 1 error in his version. This was the hardest book for our scanning program to digest, so almost certainly, 1% is an upper bound on our overall scanning error rate. Campbell also noted 6 positions in this set in which the book's solution was incorrect, but he was able to repair them all by adding or deleting a piece, so these errors are explainable as printing errors; this, added to the 3 other printing errors we had spotted ourselves in this book, yields a 1% misprint rate. Even in the event all such clerical errors were eliminated, this is hardly any guarantee that the chess problems and their answers are in fact valid. Generally, we are not able, and have not tried to, provide such a guarantee.

However, the sets (Reinfeld, Bratko-Kopec) which have been administered to many computers previously are naturally expected to be highly reliable. This testing process has, over the years, uncovered 10 cooks[4] (most of which were not serious) and 8 wrong answers, that we know about, in the 300 position win-at-chess set, and 1 cook in the Bratko-Kopec 24 position set. This leads to an estimated error rate due to wrong analysis of 3%, and due to cooks, of about 3%, and the cook rate is presumably even higher in certain of our files whose sources were not going to any great effort to eliminate cooks. We also ran Gnuchess on the `larsen2`, `larsen3`, `bellin` and `finecomb` sets (total 197 problems) thinking 225 minutes per problem, and then carefully examined the results, finding 2 mis-scanned positions and 14 possible cooks. Analysing these cook candidates with the aid of Deep Thought and/or members of the Princeton chess club, we found 3 cases where gnu225 improved on the book's solution, 5 cases where gnu was completely wrong, 3 where gnu found a nonserious cook, 1 case where gnu missed the combination but probably would still have won, 1 case where gnu's suggested move might have been a cook, although the analysis gnu gave to justify it was false, and in the 1 remaining case, we are not sure. Details will be found in the testfiles. Gnu's most impressive find was (Diagram 2). On the other hand, the most impressive example of gnu's bugginess was (Diagram 3), in which gnu had claimed that after a 17 ply (plus quiescence) search, its move `1.g6g7` led a a forced checkmate (win). In fact, it leads, as Larsen showed in his book, to a draw. The second move `2.g7g8=Q` in Gnuchess's "principal variation" is in fact a further mistake leading to a *loss*, and its `4.d4d5` comes in reply to `3...d5d4` – illegal. (Also, on 29 of these 197 problems, gnu225 hung...)

See also §3.4 where an estimate that 1% of the answers were mistyped, is given.

Bottom line: our guess is that 5-6% of our questions are buggy, $<1\%$ coming from mis-scanned positions, 1% from mistyped moves, and 1% from misprints in the source book, and 3% from wrong analysis in the source book. We don't count nonserious cooks as "bugs."

So long as our tests are hard enough that no testee is capable of scoring 80%, these error rates are not going to matter too much, since the vast majority of the wrong answers are going to be due to the testee's incompetence and not to the tester's. By the time chess computers start getting good enough to break 80%, the error rate in our set will hopefully have shrunk.

2.3 Tactics

To create a tactics test suite, we scanned in the 1001 positions from Reinfeld's book [32]. The answers at the back of the book were typed in by hand (their first move only). The 300 positions from Reinfeld's other book [33], along with their answers as checked by Hans Berliner and HITECH, were obtained from Berliner. Reinfeld's positions were mostly culled from master games. We also scanned the 100 positions from chapter 1 ("Find the combination") of [25], with the first moves of their answers, along with the ≈ 100 tactics positions from [16]. For some purposes these combinations may not be difficult enough. Hence we also scanned the 879 positions in the second (hard) half of the "encyclopedia of chess middlegames" [1] and 50 positions from [12] which (Fine seems to think) are among the most difficult combinations ever played over the board.

So in total we've got ≈ 2530 positions to test "tactics," that is, positions in which, for the most part, there is a unique best move which forces the win of a usually decisive amount of material, or checkmate, or forces a draw in an otherwise lost position, and all these things happen fairly quickly (the deepest solutions are about 27 ply long, the shallowest, about 5 ply).

Strength versus results: Larsen [25] claims that in his 100 positions a [human] master would find the right answer in 30 seconds, 95% of the time, if the master were forearmed with the knowledge that a combination was there. (Actually, the masters I've tried it on seem to take about 150 seconds, not 30.) Reinfeld's and Larsen's tactics

[4] For the purposes of this paper, a "cook" means an alternate solution with the same game-theoretic value. In cases where move A leads to a win of a knight for a pawn while move B leads to a quick checkmate, move A is probably a cook, according to our definition, but it is hard to take it seriously. "'Serious" cooks are those which have a good claim to being the "best move," in the minds of chessplayers acquainted with the analysis.

collections (recall, Deep Thought solved 297/299) are easier than the hard ECM positions, which in turn are easier than the Fine positions.

2.4 Positional play

Far harder to create (and to be certain of accuracy) is a set of positions testing "positional play," loosely defined as positions in which the best move forces some positional advantage which would not be recognized by a naive evaluation that only knows about material and checkmate. The issue is blurred. By searching sufficiently deep, even these positions should be soluble, even with a naive evaluator. Also, one certainly cannot ignore material and checkmate when solving these positions, and indeed a small number of our "positional" positions are purely tactical, since sometimes chess authors like to insert a tactical trick or two into their discussions of positional strategy just to keep you on your toes.

We scanned in the 495 positions from [13], of which ≈ 300 (allegedly) have a unique best move and thus are suitable for test purposes. The 48 positions from chapter 2 ("Find the plan") of [25] and the 33 positions from [6] are nearly all usable, as are the 24 positions of the "Bratko-Kopec test" (apparently mostly culled, in turn, from [17]) and the ≈ 25 "planning" and "analysis" positions from [16]. We also got 26 positions from [21], 25 from [30], 18 from [12], 10 from [7], about 50 from [20], about 110 from [23], and about 130 from miscellaneous sources.

Most of these sources provide fairly long textual explanations justifying the best move, but we only typed in the move itself, along with a reference to where in what book we got it from.

Thus we have about 800 positions testing "positional" play.

Strength vs results: Bellin and Ponzetto [6] claim that, "based on many practical tests with players of all categories," if you achieve a perfect score on their test you would have a 2760 FIDE rating – comparable to the world champion – but with a 50% score you would be about FIDE 1550, with roughly linear interpolation in between. This is with thinking 30 minutes per position and a lot of hints. The Gelfer positions are slightly easier than the Bellin positions. Larsen says his "find the plan" positions are harder than his "find the combination" positions, and advises 10 minutes of thought on each one; but they are somewhat easier than the Gelfer positions. Our "positional" collection is in turn easier than the Larsen planning positions.

2.5 Endgames

It is universally agreed that endgame play is qualitatively different from middlegame and opening play.

Endgames also can involve themes of stalemate, perpetual checks and perpetual attacks, insufficient versus sufficient mating material, recognition of known wins and draws, underpromotions, and the construction of "invasion-proof fortresses," all of which are notoriously hard for computers to understand, or else are notorious sources of program bugs, and thus are good testset material on general grounds. Also, we have included some extremely deep wins which a computer could surely only solve with the aid of "transposition tables," if then.

We scanned in essentially all suitable positions from [36], [37], [19], [4], [35], and the 63 problems from chapter 3 ("practical endgames") of [25].

We also created some easy positions designed to make sure the testee knows about all the kinds of underpromotion and certain standard wins and draws. By and large we have tried to stay away from positions with ≤ 6 men on the board, since the next generation of chess computers may be able to solve all such positions with the aid of enormous databases. Those rooting for computers will be gratified to know that about 88% of "book" endgame positions memorized by grandmasters, i.e. those in [14], are of this "potentially database solvable" type. The positions we have scanned in are for the most part not of this type, and are not the sort of things a grandmaster would intentionally memorize; he would instead seek to understand the general methods and ideas behind them.

The result is a set of about 300 endgame problems of greatly varying difficulty. Certainly the hardest ones would be difficult to solve over the board, even for a world champion. The easiest ones may be solved in seconds by an average player.

We have also scanned in the 1745 positions from [1], volume 3[5]. These expanded to 1797 positions, due to "two-in-one" positions.

2.6 Openings

Unfortunately, we are unaware of any chess book analogous to "1001 combination problems" but which instead concerns the right moves to make in chess openings. Over and over one reads in chessbooks that we must learn the principles of opening play so that we may reason for ourselves instead of just memorizing book lines – but despite this, few exercises are available to help us.

[5] We plan to scan in the other 4 volumes of these FIDE Encyclopedia of Chess Endings, if and when we can obtain them. Another book that was recommended to us, but which we have not scanned, is [26].

As a result, we had to create our own openings test. We combined 52 positions from Alekhine's greatest games [2], 27 from Karpov's games [15] and some well-known positions from openings manuals, in particular Keene and Levy's [22] and Znosko-Borovsky's [38], for a total of 110. This test should be taken with the program's opening book disabled, since the purpose is to see how well the program would play in an opening position outside its book.

The positions from Alekhine and Karpov games were chosen because the authors seemed to think that there was a unique best move, while in the actual game something else (i.e. a mistake) was usually played. We regard this as evidence that the right move was hard to find. (Some opening positions will be found in our positional play sets too.)

The positions from opening manuals were chosen because getting the right answer, would (for a human testee, at least) demonstrate understanding of some important strategic or tactical fact about that opening.

This is quite likely a poor test. It is very unclear how hard it is, and how much variability there is in the difficulty of the individual problems. We tried to be diverse, picking positions from all sorts of openings, but the total size of our openings test is small.

2.7 How to take and grade the test: a suggestion

Supply your computer with a fixed amount of time in which it is to solve all the problems in an entire test set. It is allowed to divide up the time in any way it pleases among the problems. (Obviously, using a simplistic time-allocation strategy is going to cost you.) Record how many it got right and what was the total time used. A more detailed approach would be to record which answer it suggested for each problem, along with its prediction of the next few moves, and its evaluation. This information is helpful in debugging the test set. S.J. Edwards suggests taking the test repeatedly, each iteration using twice the thinking time of the previous iteration. He notes that his program SPECTOR (estimated rating 2050) thinking for 1 second per problem, gets 207 right out of 1001 in the `reinfeld.fin` set, and then each of nine successive time doublings adds an increment whose mean is 62.3 and whose standard deviation is 36 to its score. Edwards's empirical observation is that each time-doubling usually seems to add roughly a constant to one's test score, and his point is that the "slope" 62.3 in the log-linear graph is perhaps as interesting a number as the score 207 at one particular time allowance. (See also our experiments with "Flea" in footnote 10, in which similar behavior with roughly constant "slope," also arose.) The test files which will probably prove to be most useful are those for which you possess a copy of the source book.

3 The optical scanner and related software

3.1 Location of chessboards on the page

We scanned in the chess books at a resolution of 300 dots per inch using a Hewlett-Packard "Scan Jet Plus" scanner. Although the chess books were theoretically printed in pure black and white, the optical components in the scanner cause apparent grey levels between closely spaced black marks such as the cross hatchings on dark squares. To eliminate this phenomenon (as well as some printing problems such as ink dropouts) we used the scanner in 1-bit per pixel mode, with a black/white threshold that was carefully chosen for each book (or in some cases, for each page).

The books employed a variety of page formats, some with chess boards and text freely intermixed. To locate the chessboards, we computed the complete convolution of each page image with the image of an empty chessboard.[6] This would be prohibitively expensive using the naive convolution algorithm, but is feasible using the FFT convolution method.

Putative chessboard locations were then found by the following method:

1. Initialize a 2D array with the output of the convolution mentioned above.

2. Propose a chessboard at the location of the largest value in the array.

3. There will be many false side peaks associated with this peak because a chessboard that has been shifted by two squares still matches itself pretty well. However, no other chessboard would have been printed overlapping the first one. Thus, we can zero out all values in the array that lie within one chessboard width of the peak identified in step 2.

4. If the whole array has been zeroed out, stop. Otherwise, go back to step 2 and propose a chessboard at the location of the largest remaining value in the array.

[6] The latter must be constructed anew for each chessbook, but this is easy to do using image editing tools.

The output of this algorithm is a sequence of chessboard candidates, listed in decreasing order of quality. Using a graphical interface and a mouse it is easy to examine these candidates visually and discard the bad ones at the end of each list.

Once each chessboard had been located and its image had been extracted from the page, we generated several rotated versions of the image[7] and selected the version which most closely matched our idealized image of a chessboard.

From each chessboard image we then extracted 64 subimages corresponding to its squares. Each subimage was downsampled (using the appropriate filtering) to yield a standardized 14 by 14 pixel image of a square. These downsized images (which contained grey values since each pixel represented many pixels in the original image) were the input for the piece recognition system described in the next section.

This convolution and rotation method worked better than several simpler image segmentation methods that we had previously tried, but it was so slow it had to be performed "off line" in a big batch job.

3.2 Piece recognition

The task of the piece recognition system is to look at the image of a square and decide which of 13 different kinds of square it is: an empty square, or a square containing one of 6 kinds of men of 2 colors.

Because chess books employ slightly different fonts and are printed on different printing presses, the classifier needs to be retrained on each new book. Template matching works quite decently on light squares, but can provide terrible performance for men on dark squares where the dark squares are halftones or hashmarks. Therefore we used a trainable "neural network" to classify the images. Our stategy was to train a network on images from the first few pages of a book and then use the resulting system to process the rest of the book. Because we needed to learn an effective classification rule from a small number of training examples, we were compelled to use a network that had a correspondingly small number of free parameters.[8] Thus we employed simple hyperplanes (or "perceptrons"), rather than multi-layer neural networks that would have had more degrees of freedom, and we radically reduced the dimensionality of our input patterns by downsampling them to a 14 by 14 pixel format, which was about the smallest input format in which the pieces were still easily recognizable.

Our classification system contained 13 perceptrons, each of which computed a hyperplane that separated the images (considered as points in a 196-dimensional space) of a particular kind of square from the images of every other kind of square.[9] After each perceptron had been trained using the standard perceptron learning algorithm [27], its weight vector was rescaled to be of unit length. Then, to classify a new pattern, the system computed the dot product of the pattern with the weight vector of every class and assigned the pattern to the class for which the dot product was largest.

The error rate of this method was around 1 percent. Later experiments showed that multi-layer neural networks could have cut the error rate in half, but these networks would have required more training data, and a system that needs to be trained on a whole book is of little use in processing that book.

The most important component of our system was a graphical interface that took batches of several hundred squares, classified them using the automated method described above, and then displayed them in rows according to their putative label. Since all of the examples of each class (*e.g.* all the kings, all the knights) were displayed side by side, and since most of the examples had in fact been labeled correctly, erroneously labeled images tended to "pop out" visually and could be immediately corrected by the human operator of the program using the terminal's mouse. We believe that a high level of accuracy was obtained by our system because we didn't require the human supervisor to actively think about categories and labels, but instead relied on a hard-wired visual ability to judge the similarity of adjacent objects.

3.3 Answers and final check

The answers at the back of each book were typed in by humans. Only the first moves of extended solutions were entered. These were later converted to long form double algebraic notation (from a large variety of original move notations) using an automated parsing program. Such a parser can be devised by generating all legal moves in that position in a large number of notations, then seeing which, if any, of these (typically 1000) strings match the input string. Rare cases where no match occurred (illegal move), or where more than one occurred (ambiguous notation), were refereed by a human supervisor. During this notation-conversion process, we performed the automated checks described in §2.2.

---

[7] The rotation angles covered a range of about 2 degrees.

[8] Learning theorists have shown that the performance of a neural network can be expected to increase as the ratio of training patterns to free parameters increases [5].

[9] To make the classification task easier, we actually handled the dark squares and the light squares of a chessboard using two separately trained recognition systems.

3.4 Plagiarism

This entire effort was, of course, essentially automated plagiarism. Of course, we have cited our sources. And of course, almost all of the positions in the sources were themselves taken from grandmaster games or published studies, and were not due to the authors of that book.

We were somewhat discouraged to find, however, that we were not the first plagiarizers! Many chessbooks contain the same diagrams as will be found in other chessbooks, usually with no attribution, leading to about 423 repeated positions (corresponding to 203 unique positions) in our test suite. (We make no claims about how many of these were intentional plagiarism and how many were coincidences.) Reinfeld seems to have been the most popular book to plagiarize from – in some cases he even plagiarized himself in later books.

Examining these 203 positions, we found that in 4 of them the answers differed! Of these 4 disagreements, 2 were due to known (and not very interesting) cooks in problems from [33], but 2 of them were serious, and turned out to have been due to typing errors we made (when typing in the answers from the book) that passed our sanity tests. We have now corrected these errors, but anyway this gives you some idea of the frequency of wrong answers in our test set – 1%.

Since our format is one position per line, anybody who wishes to eliminate these repeats can easily do so by use of the UNIX tool "`sort -u`."

4 Experience with the test, so far

We have administered the test to the public-domain chess program `gnuchess-4.0 pl62`, authored by Stuart Cracraft and others, and obtainable by anonymous `FTP` from `prep.ai.mit.edu`.

Gnuchess-4.0, running on a Sun SPARC 10, has played over 7500 games on the internet chess server, achieving a rating of 2310 at "blitz" chess and 2166 at "standard" chess. We ran it on Silicon Graphics IRIS indigos with MIPS R4000 50MHz IP20 processors and over 30 megabytes of RAM. On these machines, it searches at rates of 12700 ± 5000 nodes per second. We allocated 1 minute of CPU time for each test question. We did *not* turn off gnuchess's opening book. Due to bugs, occasionally gnuchess would hang and simply never answer the question, in which case we simply killed it and marked that question as WRONG. The results follow[10].

Test File Name	# RIGHT	# WRONG	(# HUNG)	TOTAL	% RIGHT	Comment
bellin.fin	6	27	1	33	18	Hard positional
ece3.fin	803	994	27	1797	45	R,N,B endings
finecomb.fin	8	42	0	50	16	Very hard combinations
gelfer.fin	84	243	2	327	26	Medium hard positional
hardmid.fin	279	600	13	879	32	Tough combinations
jenoban.fin	108	94	3	202	53	Tactical endings
kcmdm.fin	71	74	1	145	49	Separable mixture
kotov.fin	41	74	1	115	36	Mixture
larsen1.fin	68	32	4	100	68	Find the combination
larsen2.fin	15	33	1	48	31	Find the plan
larsen3.fin	27	36	3	63	43	Practical endings
openings.fin	31	79	1	110	28	openings understanding
positional.fin	54	100	1	154	35	Positional mixture
reinfeld.fin	670	331	20	1001	67	1001 combinations
sherev.fin	15	45	0	60	25	endgame strategy
speelman.fin	63	104	3	167	38	endgames
winatchess.fin	212	88	4	300	71	300 combinations
combined	2555	2996	85	5551	46	

Gnuchess (1 minute) results.

We also tried running gnuchess at 15 minutes per problem on selected test files. Not surprisingly, it got a few more right answers:

[10] Other results: Mike McGann ran our test at 1 minute max per problem on Gnuchess 4.0 patch level 63 on a SPARC 10 (which appears not to have the bugs that plagued our installation of Gnuchess on SGIs) and the score was 2664.

"Flea" is a 3700-line C program by Rico Tudor, featuring a relatively fast and simpleminded evaluation function. It has a rating of 2036 based on 470 blitz games on the internet chess server. Running on our SGI machines (qv), Flea searches $\approx$ 7000 nodes/second. With its opening book turned off, and thinking 0.1 seconds per problem it gets 1149/5551 (20.7%) right. 0.2 seconds: 1276 (23.0%). 1.0 seconds: 1564 (28.2%). 2.0 seconds: 1791 (32.3%). 10 seconds: 2110 (38.0%) 1 minute: 2505 (45.1%). Comparison of flea at 1 minute/problem vs 2 seconds: 1537 common correct answers; 968 answers 1 minute found but 2 seconds didn't; 254 that 2 seconds found but 1 minute didn't. (Note: a random move generator would get about 4.5% right, or 250, on our test.)

Amoeba is a C program by David J. Slate with an internet rating of 2250 based on 97 blitz games. At 1 second per problem, it gets 1762 (31.7%) right. 2 seconds: 2073 (37.3%). 1 minute 2764 (49.8%).

Test File Name	# RIGHT	# WRONG	(# HUNG)	TOTAL	% RIGHT	Comment
bellin.fin	8	25	1	33	24	Hard positional
finecomb.fin	14	36	1	50	28	Very hard combinations
larsen2.fin	16	32	1	48	33	Find the plan
larsen3.fin	31	32	4	63	49	Practical endings

Gnuchess (15 minute) results.

Our chess test may be able to shed some light on the debate concerning how much more speed is necessary to make a computer program become world chess champion. Experiments by Condon and Thompson with BELLE running at different speeds [9] had shown that each doubling of machine speed would typically yield between 40 and 100 extra USCF rating points in the rating range (ratings: 1350-2520; ply deep: 4-11) studied. (Stronger players gain less rating points from a speed doubling. The 40-point claims at plies 10 and 11 are extrapolations by M. Newborn [28] based on observed move change frequencies in Belle's tree search.) But Hans Berliner has speculated that the brute force approach of simply increasing machine speed, might run into a roadblock. Specifically, he suspects that merely increasing speed will increase rating until the point where the chess program achieves "tactical sufficiency." Beyond that point, he believes that strength increases resulting from extra speed will be small, and it is best not merely to increase one's search depth, but instead to increase one's "positional knowledge."

Anyway, observe that a 15× improvement in machine speed bought gnuchess a 75% score increase on the tactics problems in `finecomb.fin` but only a 15% improvement on the positional, planning, and endgame problems in `bellin.fin`, `larsen2.fin`, and `larsen3.fin`.

5 Acknowledgements

We would like to thank Hans Berliner for giving us the Hort-Jansa and the Reinfeld *Win at Chess* sets, and Murray Campbell for checking our `hardmid` set, and both for various discussions. Steven J. Edwards (`sje@world.std.com`) gave us various useful comments and is converting our database to a "world standard" format endorsed by him. Mark Kantrowitz informs us that he will also put our test in the "CMU AI repository" – anonymous FTP to `ftp.cs.cmu.edu` in the directory `user/ai/software/games/chess/test/`.

We would also like to thank Sue Bredhoff for typing in chessmoves.

6 Last minute additions to this article

We have just scanned in ≈ 300 more positions from the two books: Dvoretsky [10] and Krabbe [24]. Dvoretsky's book, formerly in Russian and then titled *The art of analysis*, contains many difficult and very deeply analysed positions selected to be useful for training grandmasters. Krabbe's book is full of bizarre moves, underpromotions, castling, and 3-time repetition themes, so it ought to be useful for exercising bugs in your chess program.

References

[1] A. Adorjan (GM) et al. *Encyclopedia of chess endings,* FIDE, Beograd Yugoslavia, 1986 ISBN = 86-7297-005-5.

[2] A. Alekhine (Wch). *My greatest games of chess 1908-1937,* Dover reprint, 2 vols in 1, 1985.

[3] T.S. Anantharaman, M.S. Campbell, F-s Hsu. *Singular extensions: adding selectivity to brute force searching,* Artificial Intelligence 43,1 99-109. also in ICCA J 11,4 (1988) 135-143.

[4] Y. Averbakh (GM). *Chess endings, essential knowledge,* Pergamon 1987. ISBN = 0-08-032045-7.

[5] E.B. Baum and D. Haussler. *"What size net gives valid generalization?"* Neural Computation, vol.1, 1989, 151-160.

[6] Robert Bellin (IM) and Pietro Ponzetto. *Test your positional play,* Macmillan 1985. ISBN = 0-02-028090-4.

[7] Hans Berliner (WCorrCh), D. Kopec (IM), E. Northam. *A taxonomy of concepts for evaluating chess strength,* pages 179-191 in *Advances in computer chess 6,* Ellis Horwood 1991. ISBN = 0-13-006537-4. Also in IEEE Supercomputing '90, IEEE computer science press November 1990.

[8] D.Kopec and I. Bratko. *A comparison of human and computer performance in chess,* page 57-72 in Advances in Computer Chess III, edited by M.R.B. Clarke, Pergamon 1982.

[9] J.H. Condon and K. Thompson. *BELLE,* page 201-210 in *Chess skill in man and machine,* edited by Peter Frey, Springer 1983.

[10] Mark Dvoretsky (IM). *Secrets of Chess Training,* Batsford 1991. ISBN=0-7134-6287-6

[11] Max Euwe (WCh). *Strategy and tactics in chess,* David McKay Co., New York ≈1950.

[12] Reuben Fine (GM). *The world's great chess games,* Bonanza Books, Crown Publishers, 1951. Now available as a Dover reprint.

[13] Israel Gelfer (Coach of Israeli nat'l chess team): *Positional chess handbook,* Macmillan 1991. ISBN = 0-02-028831-X.

[14] David Hooper. *A pocket guide to chess endgames,* G. Bell, 1970. ISBN = 0-7135-1761-1.

[15] A. Karpov (Wch). *Karpov's collected games,* (David Levy, ed.) Robert Hale and Co. London 1975.

[16] D. Kopec, G. Chandler, C. Morrison, N. Davies, I.D. Mullen. *Master chess, a course in 21 lessons,* Pergamon Press 1985 ISBN = 0-08-029725-0.

[17] Hans Kmoch (IM). *Pawn power in chess,* Dover publications 1990. ISBN = 0-486-26486-6.

[18] N. Krogius (GM), A. Livsic, B. Parma, and M. Taimanov (GM). *Encyclopedia of chess middlegames and combinations.* FIDE, Beograd 1980.

[19] I.A. Horowitz (IM). *How to win in the chess endings,* Tartan books, David McKay co. inc. New York 1957.

[20] V. Hort (GM) and V. Jansa. *The best move,* RHM press (apparently now defunct) New York 1980.

[21] Raymond Keene (GM) and Andrew Whiteley. *Ray Keene's good move guide,* Oxford University Press 1982. ISBN = 0-19-217582-3.

[22] Raymond Keene and D.N.L. Levy (IM). *How to play the opening in chess,* B.T. Batsford Ltd., 2nd ed 1980.

[23] Alexander Kotov (GM). *Think like a grandmaster,* B.T. Batsford ltd, 1971. ISBN=0-7134-3160-1.

[24] Tim Krabbe. *Chess Curiosities,* George Allen and Unwin, London 1985 ISBN=0-04-794021-2

[25] Bent Larsen (GM): *Bent Larsen's good move guide,* Oxford University Press 1982. ISBN = 0-19-217593-9.

[26] A. Livshits and J. Speelman. *Test your endgame ability,* 2nd edition, Batsford 1992.

[27] M.A. Minsky and S.L. Papert. *Perceptrons,* MIT Press 1969.

[28] M. Newborn. *A hypothesis concerning the strength of chess programs,* ICCA J 8,4 (Dec 1985) 209-215; 9,4 (Dec 1986) 199.

[29] J.B. Nielsen. *A chess-computer test set,* ICCA J 14,1 (March 1991) 33-37.

[30] Aron Nimzowitsch (GM). *My system,* (New, re-edited edition) Hays Publishing Inc. 1991. ISBN = 1-880673-85-1

[31] Aron Nimzowitsch. *Chess praxis,* Dover Reprint 1962.

[32] Fred Reinfeld. *1001 winning chess sacrifices and combinations,* Melvin Powers Wilshire book company (third printing 1955) ISBN = 0-87980-111-5.

[33] Fred Reinfeld. *Win at chess,* Dover 1945 ISBN = 0-486-20438-3.

[34] Fred Reinfeld. *The complete chessplayer,* Prentice Hall 1953.

[35] M.I. Shereshevsky. *Endgame strategy,* Pergamon 1985 ISBN = 0-08-029746-3.

[36] Jon S. Speelman (GM). *Endgame preparation,* Batsford, 1981. ISBN = 0-7134-3999-8.

[37] Jon S. Speelman (GM). *Analysing the endgame,* Batsford 1981. ISBN = 0-7134-1897-4.

[38] Eugene A. Znosko-Borovsky. *How to play the chess openings,* Dover 1971, reprint of Pitman 1935. ISBN = 0-486-22795-2.

[39] Eugene A. Znosko-Borovsky. *How to play the chess endings,* Dover 1974, reprint of McKay 1940. ISBN = 0-486-21170-3.

(Higher chess titles than national master have been listed, for authors, when known.)