More Games of No Chance
MSRI Publications
Volume **42**, 2002

A Memory Efficient Retrograde Algorithm and Its Application To Chinese Chess Endgames

REN WU AND DONALD F. BEAL

Abstract. We present an improved, memory efficient retrograde algorithm we developed during our research on solving Chinese chess endgames. This domain-independent retrograde algorithm, along with a carefully designed domain-specific indexing function, has enabled us to solve many interesting Chinese chess endgame on standard consumer class hardware.

We also report some of the most interesting results here. Some of these are real surprises for human Chinese chess experts. For example, the aegp-aaee[1] ending is a theoretical win, not as previously believed, a draw. Human analysis for this endgame over many years by top players has been proved to be wrong.

1. Introduction

Endgame databases have several benefits. First, the knowledge they provide about the game is perfect knowledge. Second, the databases, because of their complete knowledge about certain domains, are a useful background for Artificial Intelligence research, especially in machine learning. Third, the databases often provide knowledge beyond that achieved by humans, (and increasingly, beyond that achievable by humans). Many endgame databases in many games have been constructed since Ströhlein's pioneering work (Ströhlein, 1970). In Chess, Thompson (1986) generated almost all 5-men chess endgame databases and made them widely available in CD format. Databases construction also enable Gasser (1996) to solve the game of Nine Men's Morris. Perhaps the most impressive endgame database construction so far is Schaeffer's work (Schaeffer et al., 1994) on Checkers. His program created all 8 men endgame databases and comprised more than 440 billion (4.4×10^{11}) positions. The databases played a very important role in Chinook's success.

[1] One side with King, one Assistant, one Elephant, one Gunner, and one Pawn against another side with King, two Assistants, and two Elephants.

We started our work on constructing Chinese chess endgame databases back in 1992. We have constructed many Chinese chess endgame databases since. We wanted our program to solve as many endgames as possible using only moderate hardware we had access to. So a major effort was made to improve the retrograde algorithm as well as examine ways to reduce the size of the database.

Armed with careful analysis to reduce the databases' size and our fast, memory efficient retrograde algorithm, we were able to solve one class of very interesting Chinese chess endgames. In this class of endgames, one side has no attacking pieces left but have various defending pieces, while the other side has various attacking pieces. In this paper, we describe our new retrograde algorithm, the database indexing we employed to reduce the size of the database, as well as the results we found, include the result for the aegp-aaee endgame. The database reveals a surprise for human players.

2. Fast, Memory Efficient Retrograde Algorithm

The retrograde algorithm used to construct the databases demand large computing resources. Thompson's 1986 algorithm (Thompson, 1986) needs two bits of RAM for every board configuration, or random access to disc files, and it only gives one side's result. Stiller's algorithms (Stiller, 1995) need a highly-parallel supercomputer. Others also need similarly large amounts of memory or need a very long time to build a moderate database. In chess for example, Edwards reported that a 5-man endgame database took 89 days on a 486 PC (Edwards, 1996). Six-men chess databases have to deal with approximately 64 billion positions, although symmetries reduce this, and careful indexing to eliminate illegal positions can reduce this to 6 billion for some positions (Nalimov, 2000). Database construction is hardware-limited. This will remain so indefinitely. Although hardware advances continually bring computations that were previously out of reach into the realm of practicality, each additional piece multiplies the size of the task (of creating a complete set of databases) by around three orders of magnitude. Hence efficient algorithms, as well as the indexing methods that minimize hardware resources will always be desired. We present our fast and memory efficient algorithm here first, and discuss the indexing methods in the next section.

2.1. Previous Work. The first widely-known description of the basic retrograde algorithm was by Ströhlein (1970). It was independently re-invented by many people, including Clarke (1977) and Thompson (1986, 1996). Herik and Herschberg (1985) give a tutorial introduction to the retrograde concept. Thompson's first paper (1986) gives a brief but clear outline of his algorithm. His second paper (Thompson, 1996) gives details of an improved algorithm he used to solve 6-piece chess endgames.

These algorithms all assume one side is stronger than another and only calculate the distance to conversion (DTC), which is the number of moves the stronger side needs to either mate the weaker side or transfer into a known win position in a subgame. There are two problems here:

- The algorithm assumes that if the stronger side does not win a position, it must be a draw. This is only true for some endgames. Other endgames need a separate database to be constructed for the other side. And even with two separate databases, one has to be very careful when using this database system. Some endgame types now need one database to be probed, others need two, and there are additional technical inconveniences and hazards, in using the databases after construction.
- Sometimes the distance to mate (DTM) may be preferred to the distance to conversion.

To address these problems, variations on the basic algorithm have been used. Wu and Beal (1993) designed an algorithm that retains full information for both sides when generating databases. Edwards (1996) also designed an algorithm which can generate chess endgame database containing both side's distance to mate/loss. However, these algorithms usually need access the main database during the construction, and so require much more memory than Thompson's algorithm. To use these algorithms, one either has to run it on very high-end machine, or wait while it takes very long time to run. For example it took Edwards' program more than 89 days to build a 5 piece chess endgame database on a 486 PC (Edwards, 1996). In the most recent work on chess endgame KQQKQQ Nalimov had to use a 500 MHz server machine with 2GB of memory to build this 1.2 billion entries database (Nalimov 1999, 2000).

2.2. Fast, Memory Efficient Retrograde Algorithm We devised an algorithm which only needs one bit per position like Thompson's algorithm, but can still generate full information for both sides, like other algorithms. Furthermore, it can compute either distance to mate (DTM) or distance to conversion (DTC), controlled by a boolean variable.

We present our new retrograde algorithm on the next page in a pseudo C format:

The algorithm generates a pair of databases, one for each side. And the result gives the exact distances for win/lose/draw. In other words, it generates full information about the endgame.

The algorithm itself can be understood in terms of relatively simple operations on bitmaps, in the style first used by Thompson (1986). The summary below gives the memory and access requirement for this algorithm.

- DoInitialize() uses sequential access to database W, B and bitmap S. It also uses random access for bitmap R.
- Load() use sequential access to one database and one bitmap.

```
DATABASE W, B;  // full databases (i.e. depth to win/loss for each position),
                // W for White-to-move positions, B for Black-to-move,
                // sequential access only
SBITS    S;     // sequential access only bitmap
RBITS    R;     // random access bitmap

void TopLevel
{
  DoInitialize();
  n = 0;                              // depth to mate or conversion
  while (!DoneWhite() && !DoneBlack())
  {
    if (!DoneWhite())                 // last pass added new positions
    {
      S = Load(W, WIN_IN_N(n));         // S = WTM win_in_n
      R = Predecessor(S);               // R = BTM predecessors of S
      S = Load(B, UNKNOWN);             // S = BTM unknown
      S = S & R;                        // S = BTM maylose_in_n
      R = Load(W, WIN_<=_N(n));         // R = WTM win_in_n or less
      S = ProveSuccessor(S, R);         // S = BTM lose_in_n
      B = Add(S, LOSE_IN_N(n));         // B += S
      if (dtm)                          // distance_to_mate?
        S = Load(B, LOSE_IN_N(n));      // S = BTM lose_in_n
      R = Predecessor(S);               // R = WTM maybe win_in_n+1
      S = Load(W, UNKNOWN);             // S = WTM unknown
      S = S & R;                        // S = WTM win_in_n+1
      W = Add(S, WIN_IN_N(n+1));        // W += S
    }
    if (!DoneBlack())                 // done for BTM?
    {
      S = Load(B, WIN_IN_N(n));         // S = BTM win_in_n
      R = Predecessor(S);               // R = WTM predecessors of S
      S = Load(W, UNKNOWN);             // S = WTM unknown
      S = S & R;                        // S = WTM maylose_in_n
      R = Load(B, WIN_<=_N(n));         // R = BTM win_in_n or less
      S = ProveSuccessor(S, R);         // S = WTM lose_in_n
      W = Add(S, LOSE_IN_N(n));         // W += S
      if (dtm)                          // distance_to_mate?
        S = Load(W, LOSE_IN_N(n));      // S = WTM lose_in_n
      R = Predecessor(S);               // R = BTM maybe win_in_n+1
      S = Load(B, UNKNOWN);             // S = BTM unknown
      S = S & R;                        // S = BTM win_in_n+1
      B = Add(S, WIN_IN_N(n+1));        // B += S
    }
    n = n + 1;
  }
}
```

- Predecessor() use sequential access for bitmap S and random access for bitmap R.
- ProveSuccessor() use sequential access for bitmap S and random access for bitmap R.
- Add() uses sequential access for one database and bitmap S.

Thus the peak memory requirement is only random access for one bitmap and sequential access for one bitmap and two databases. Because the sequential file access is many times faster than random access, we keep the random access bitmaps to a minimum, and require just one bitmap in memory. Apart from the chess/Chinese chess specific routines, the rest is just simple load and boolean operations over the files.

The algorithm is a generic one, and can be used to construct databases for other games. For example, to construct a 5-men pawn-less chess endgame database, 15MB RAM is sufficient to avoid random disc access. The algorithm will enable such databases to be built on a modest desktop PC in the matter of hours.

3. Reducing the Size of the Database

The Chinese chess board has 90 squares, so the simplest formula for the size of a sufficiently-large database is simply 90^n, where n is the number of the pieces in that endgame. For almost all interesting endgames, this is too large for today's technology to handle. However, the size of database can be greatly reduced by careful analysis of geometric and combinational symmetries, and game-specific details. In the following sections, we detail the methods we used to reduce the size of our database. Our methods may be compared with those of Nalimov and Heinz (2000) for western chess.

3.1. Limiting the Pieces' Placement to Legal Squares. In Chinese chess, certain type of pieces can only move inside certain parts of the board. For example, the king can only be inside the palace, the assistants can only move in the five squares inside the palace, the elephants only have seven squares, and the pawn can only move forward before it crosses the river. Table 1 gives each kind of piece and its possible squares.

So a direct way to calculate the size of a more compact database is to enumerate all possible squares for every pieces in that endgame. This brings the size down considerably, but again for most interesting endgames, it is still too big for the hardware we have access to. Fortunately, there are further ways to reduce the size, as described in the next few sections.

3.2. Vertical Symmetry. The Chinese chess board has vertical symmetry and this gives us a reduction factor of almost 2. It is almost rather than exact, because it is possible for all pieces to be in the centre file, and such positions are their own mirror position, leading to no reduction. Moreover, there is a

Name	Notation	Squares
King	k	9
Assistant	a	5
Elephant	e	7
Horse	h	90
Chariot	c	90
Gunner	g	90
Pawn	p	55

Table 1.

significant processing cost to obtain the full reduction, because the program must process all pieces in turn to reflect all piece locations, and this operation has to be re-done every time a move is made. The retrograde analysis program is both space intensive and compute intensive, so this is unwelcome. If we simplify the processing to only consider the first piece in the symmetry reduction, we achieve about half the possible reduction, as shown in Table 2.

3.3. Multiple Piece Symmetry. If in an endgame, there is more than one piece of the same type, we can exchange these pieces' places without altering the position. In other words, the pieces of the same type together have a single contribution to the database. The size of the contribution can be determined by the combinatorial arithmetic as follows.

If an endgame has n of the same type of piece and this type of piece can only move in m Squares, and then the size of this contribution is:

So for the endgames with one side having two of the same type of pieces, this reduces its size by more than half, and the saving is even greater if one side has more than two of the same type of piece.

Name	Notation	Total	VSR	Saving
King	k	9	6	33.33%
Assistant	a	5	3	40.00%
Elephant	e	7	4	42.86%
Horse	h	90	50	44.44%
Chariot	c	90	50	44.44%
Gunner	g	90	50	44.44%
Pawn	p	55	31	43.64%

Table 2.

Name	Size	VSR	Saving
kaa	70	38	45.71%
ka	40	21	47.50%
kae	275	138	49.82%
ke	62	32	48.39%
kee	183	96	47.54%
ee	21	12	42.86%
hh	4005	2045	48.94%
cc	4005	2045	48.94%
gg	4005	2045	48.94%
rp2	1485	765	48.48%
bp2	1485	765	48.48%
rp3	26235	13135	49.93%
bp3	26235	13135	49.93%

Table 3.

3.4. Piece Grouping. In Chinese chess, both king and assistants can only move inside the palace, and this allows us another possible way to reduce the database's size. We can consider a few different type of pieces together, or piece grouping. These pieces will be considered together and form a single contribution. This offers increased reduction over applying space-enumeration and symmetry to the pieces separately, because the group enumeration can eliminate impossible positions in which pieces occupy the same square. This saving is significant for Chinese chess.

Taking a king and two assistants (kaa) as an example, we can enumerate all possible patterns for these three pieces in the palace, and assign a unique id to each pattern. It turns out that kaa only has 70 different patterns, which compares to $9 \times 10 = 90$ if we do it separately.

Moreover, we can combine the piece group enumerations with vertical symmetry. In the case of kaa, 32 out of possible 70 patterns are just mirrors of others. In other words, there are only 38 different patterns if we take the vertical symmetry into account. The saving here is $32/70 = 0.457$, which is better than the savings we get from the best single-piece vertical symmetry reduction.

There are more situations in which a few pieces can be considered together and give a single contribution. Table 3 lists more of the possible piece groupings that can be utilised.

Note the kae contribution in Table 3. kae has 275 possible patterns , and 138 if we consider the vertical symmetry reduction. The saving here is 0.498, which is slightly greater than the kaa contribution, and very close to the full saving of 0.5 that represents the theoretical symmetry limit. So in practice, we should always

endgame	database size	endgame	database size
h-aaee	646,380	egp-aaee	232,848,000
c-aaee	646,380	aagp-aaee	276,507,000
g-aaee	646,380	eegp-aaee	698,544,000
hp-aaee	35,550,900	aegp-aaee	1,004,157,000
cp-aaee	35,550,900	agg-aaee	123,634,350
gp-aaee	35,550,900	egg-aaee	188,395,200
hh-aaee	27,055,350	aagg-aaee	223,719,300
cc-aaee	27,055,350	eegg-aaee	565,185,600
gg-aaee	27,055,350	aegg-aaee	812,454,300
hc-aaee	58,174,200	pp-aaee	10,120,950
hg-aaee	58,174,200	hpp-aaee	910,885,500
cg-aaee	58,174,200	gpp-aaee	910,885,500
agp-aaee	152,806,500	cpp-aaee	910,885,500

Table 4.

use the group with maximum savings to incorporate the symmetry reduction, rather than apply separate symmetry reductions.

4. Results from the Database

We concentrate on the most interesting class of endgames. In these endgames, one side has no attacking pieces left but only defensive pieces, while the other side has various attacking pieces. Players always are keen to know what kind of piece combinations are enough to win, how hard or how easy it is to win, and how long it will take to win.

The perfect knowledge contained in the databases is invaluable for anyone who is serious about Chinese chess. Moreover, our databases show that current human understanding about this kind of endgames is far from perfect. Table 4 lists the endgames that we have solved.

To build any endgame database, the program has to build all its subgame databases first. For example, c-aaee endgame have 9 subgames, k-k, c-k, c-a, c-e, c-aa, c-ee, c-ae, c-aae, and c-aee. For the endgames we list above, if we count all subgames as well, there will be 378 in total, with about 35 billion entries in the resulting databases. Our construction program solves all these subgames automatically.

Our results are summarized on the next page, with some comments made in the next few paragrpahs. In the table, “db size” (database size) is the size of the database for this endgame. If we take out the illegal positions, and disregard the various symmetry reductions, we have a more human-like classification. We call this the “real size”. This is more useful to humans when we talk about a particular endgame. All the percentage data are based on “real size” in this

				RTM(%)		BTM(%)	
database	db size	real size	DTC	W	D	D	L
h-aaee	646,380	3,363,048	1	5.58	94.42	99.88	0.12
c-aaee	646,380	3,363,048	18	84.60	15.40	48.56	51.44
g-aaee	646,380	3,363,048	1	7.68	92.32	99.63	0.37
hp-aaee	35,550,900	169,290,216	31	31.06	68.94	94.47	5.53
	(highpawn)	59,694,192	31	35.47	64.53	92.33	7.67
cp-aaee	35,550,900	169,290,216	14	99.99	0.01	8.12	91.88
	(highpawn)	59,694,192	9	100.00	0.00	7.05	92.95
gp-aaee	35,550,900	169,290,216	12	14.15	85.85	99.31	0.69
	(highpawn)	59,694,192	8	12.52	87.48	99.12	0.88
hh-aaee	27,055,350	284,348,544	17	99.34	0.66	12.70	87.30
cc-aaee	27,055,350	284,348,544	3	100.00	0.00	2.26	97.74
gg-aaee	27,055,350	284,348,544	29	40.49	59.51	92.23	7.77
hc-aaee	58,174,200	284,348,544	9	99.98	0.02	7.43	92.57
hg-aaee	58,174,200	284,348,544	22	99.40	0.60	12.73	87.27
cg-aaee	58,174,200	284,348,544	7	100.00	0.00	7.30	92.70
agp-aaee	152,806,500	765,989,832	32	44.95	55.05	86.32	13.68
	(highpawn)	269,533,904	32	60.22	39.78	75.66	24.34
egp-aaee	232,848,000	1154,955,840	26	26.99	73.01	95.60	4.40
	(highpawn)	409,481,824	26	39.12	60.88	89.93	10.07
aagp-aaee	276,507,000	2730,979,776	39	45.43	54.57	86.02	13.98
	(highpawn)	958,979,200	39	60.74	39.26	75.22	24.78
eegp-aaee	698,544,000	6752,096,208	34	27.01	72.99	95.62	4.38
	(highpawn)	2407,076,064	34	39.06	60.94	90.07	9.93
aegp-aaee	1004,157,000	5211,916,080	95	70.52	29.48	50.51	49.49
	(highpawn)	1844,080,608	73	99.60	0.40	10.91	89.09
agg-aaee	123,634,350	1270,822,608	20	99.65	0.35	12.37	87.63
egg-aaee	188,395,200	1928,810,256	42	53.36	46.64	82.02	17.98
aagg-aaee	223,719,300	4474,606,752	21	99.64	0.36	12.54	87.46
eegg-aaee	565,185,600	11210,239,584	29	99.45	0.55	12.93	87.07
aegg-aaee	812,454,300	8595,899,712	20	99.64	0.36	12.53	87.47
pp-aaee	10,120,950	99,964,368	11	12.44	87.56	99.19	0.81
hpp-aaee	910,885,500	8338,089,024	34	96.72	3.28	23.93	76.07
gpp-aaee	910,885,500	8338,089,024	40	89.54	10.46	34.34	65.66
cpp-aaee	910,885,500	8338,089,024	12	100.00	0.00	5.77	94.23
aegp-aae	334,719,000	907,504,776	36	94.11	5.89	22.16	77.84
gpp-ee	113,395,950	549,454,356	39	94.03	5.97	23.92	76.08
hpp-aee	520,506,000	2,414,172,384	46	98.67	1.33	10.79	89.21
aagp-aee	152,806,500	786,450,264	49	85.37	14.63	31.46	68.54
egg-aee	107,654,400	541,173,192	51	30.59	69.41	95.66	4.34

paper. DTC is the maximum distance for the stronger side to capture a piece and transfer to a known winning subgame. W, D, and L stand for percentage of wins, draws, and losses. The L column under RTM and the W column under BTM are omitted since the values are zero in every case.

4.1. One Major Piece. From the first section of the table on page 221, we can see that when the attacking side has only one major piece, the game is usually a draw. However if the major piece is a chariot, the stronger side does have some chance to win, especially if it is the stronger side move first. In other word, the right to move, or initiative is very important here. The winning rate is 84.60% if the stronger side moves first, and drops to 51.44% if the opponent moves first.

4.2. One Major Piece Plus a Pawn. These results are given in the second section of the table on page 221. The second row for each endgame is the statistics if we assume the pawn is a high pawn. Most Chinese chess textbooks use this term to help classify endgames.

Chariot plus a pawn is too much to defend, and this comes with no surprise, because one can always use the pawn to exchange a minor piece, and a chariot against the rest of defense is a sure win.

Gunner plus a pawn is proved to be not enough to win. The gunner needs helps from minor pieces to really take effect, and in this case there are no helping pieces.

Horse plus a pawn is a very powerful combination, and it requires a very high level of skill to play well. The results here show that horse and pawn is not enough to break the best defense, even though it does have some chances.

4.3. Two Major Pieces. See third section of the table. If the stronger side have two major pieces, the advantage is usually overwhelming, except if both major pieces are gunners. Gunners need help from minor pieces, and lack of those minor pieces prevents the win.

4.4. One Gunner, One Pawn Plus Some Minor Pieces. See fourth section of the table. If we have a gunner, a pawn, what else we need to win? Having a single minor piece, either an Assistant, or an Elephant is not enough to secure a win. Two minor pieces of same type are also not enough. One has to have one minor piece of each kind to win, and if the pawn is a high pawn, the winning is guaranteed. This is a real surprise discovery, and will be discussed in detail at next section.

4.5. Two Gunners Plus Some Minor Pieces. See fifth section of the table. Two gunners plus a Assistant is already enough for win. And so two Assistants, or one Assistant and one Elephant, is also a win. But two gunners plus an Elephant is a totally different story. It most likely will end up as a draw, especially if the opponent moves first.

4.6. Two Pawns Plus One Major Piece. See penultimate section of the table. Two pawns without any major piece is not enough to win. However, one major piece and two pawns are usually too much to defend. Chariot and one pawn is already a sure win, two pawns can only make the winning sooner. Horse and two pawns are also a win, no matter how bad the pawn's positions are. Gunner and two pawns can also be regarded as a win, but the maximum DTC of 40 indicate it can be hard to play this endgame sometime.

4.7. Some of the Hard Subgames. We list some of the hard subgame results in the last section of the table on page 221. Results for all subgames can be found online at http://www.msri.org/publications/books/Book40/files/wu-beal.txt.

These endgames are mostly winning for stronger side except for egg-aee. However, the distance-to-conversion is rather long and so it can be hard to play them accurately.

5. The aegp-aaee Endgame

Among the many Chinese chess endgames we solved so far, the most interesting discovery is the aegp-aaee endgame. It is interesting not only because humans had made a thorough investigation on this endgame and it is very well known amongst experts, but also because our research reveals that this endgame is a theoretical win for the stronger side (aegp), contrary to current human belief. Furthermore, the maximum distance to conversion (the maximum number of moves the stronger side need to capture the first piece) is 95, which is 35 more than the maximum move allowed in official Chinese chess rules.

5.1. Human Analyses. Gunner and Pawn is a well-known class of endgames, where the strong side has only one Gunner and one Pawn as the attack force, plus a few defending pieces, while the opponent usually have the best defence possible, that is have both elephants, and assistants. One of the very early Chinese chess endgame books, the *Shi Qin Ya Qu*, first published in 1570, has already a few positions in this class. The most famous work on this class of endgame in human history, however, has to be the excellent work by Chen Lianrong at 1930s. His book, *Pao Bin Endgames*, is dedicated to this class of endgame, and has been regarded as a milestone in modern Chinese chess endgame theory.

In his book, he showed that the aeegp-aaee endgame is theory win for the stronger side. That was a real surprise for the Chinese chess community at that time, because the common belief at that time was that even aaeegp-aaee is a draw game! This is one of the biggest contribution of his book.

However, our research has produced a stronger result by showing that even with one elephant, the stronger side has already enough to win, even though it can take as many as 95 moves to capture the first piece!

5.2. Computer Analysis. To produce the computer analysis, the program ran on a Pentium Pro 200 machine with 128 MB memory under Windows NT 4.0. It took about 92 hours to generate the database for this endgame containing both sides full information using the distance-to-conversion (DTC) metric.

Here is a further breakdown of the surprise result, that aegp-aaee is actually a mostly-winning endgame. It is a win provided that the pawn has passed the river or can pass the river safely, and is not an old-pawn2.

```
Database:     110011-220000       Database Size: 1004157000

                   RTM                        BTM
Illegal      244406400 ( 24.34%)        265487568 ( 26.44%)
Lose                 0 (  0.00%)        358203335 ( 35.67%)
Draw         290685444 ( 28.95%)        380466097 ( 37.89%)
Win          469065156 ( 46.71%)                0 (  0.00%)
Total       1004157000 (100.00%)       1004157000 (100.00%)

Maximum Distance-to-conversion:  95
```

If we take out the illegal positions, and disregard the various symmetry reductions, we have a better, or more human like, classification, as follows:

```
Database:     110011-220000       Real size:     5211916080

                   RTM                        BTM
Lose                 0 (  0.00%)       2579230888 ( 49.49%)
Draw        1536511488 ( 29.48%)       2632685192 ( 50.51%)
Win         3675404592 ( 70.52%)                0 (  0.00%)
Total       5211916080 (100.00%)       5211916080 (100.00%)

Maximum Distance-to-conversion:  95
```

If we assume the pawn is a high-pawn, we have:

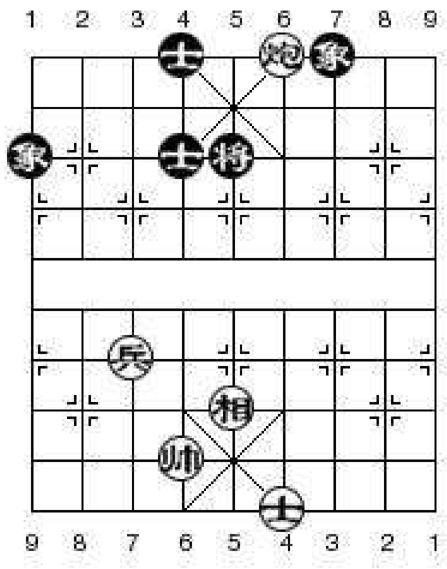
```
Database:     110011-220000        HighPawn:     1844080608

                   RTM                        BTM
Lose                 0 (  0.00%)       1642833536 ( 89.09%)
Draw           7324704 (  0.40%)        201247072 ( 10.91%)
Win         1836755904 ( 99.60%)                0 (  0.00%)
Total       1844080608 (100.00%)       1844080608 (100.00%)

Maximum Distance-to-conversion:  73
```

We give a position with the maximum distance to conversion on the next page. We use Chinese notation, where each side counts the column from his right hand side, '=' means move side ways, '+' means move forward and '−' means move backward. The first letter is the type of moving piece, followed by a number indicating which column the moving piece resides in. In case there is

3a1Ge2/9/e2ak4/9/9/9/2P6/4E4/3K5/5A3/r ---- 407963625

RTM 407963625

1. G4-8 k5=6 2. A4+5 k6-1 3. A5+4 k6=5 4. G4=5 k5=6 5. G5=2 a4-5 6. G2=4 a5+6 7. E5+3 e7+9 8. G4=3 k6-1(k6=5,a4+5,a6-5,e1-3,e1+3) 9. K6=5(K6-1,G3+1,G3+2, G3-1,P7+1) k6+1(a4+5,a6-5,e1-3,e1+3) 10. K5-1(G3+1,G3+2,G3-1,P7+1) k6-1(a6-5, e1-3,e1+3) 11. G3+1(G3+2,G3-1,P7+1) k6+1(a6-5,e1-3,e1+3) 12. G3+1(G3-2,P7+1) k6-1(a6-5,e1-3,e1+3) 13. G3-2(P7+1) k6+1(a6-5) 14. K5=4(P7+1) a6-5 15. P7+1 a5+4 16. G3=4 k6=5 17. G4=6 k5-1 18. G6=3 k5+1 19. A4-5 k5=4 20. K4=5(A5+6, G3-1) a4-5 21. A5+6 a5+4 22. K5+1(G3-1) a4-5 23. G3-1 a5+4 24. K5=6 a4-5 25. E3-5 (G3=5) a5-6(e9-7,e9+7) 26. G3=5 a4+5(e9-7,e9+7) 27. K6=5 k4-1(e9-7, e9+7) 28. E5-7 k4=5(e9-7,e9+7,e1-3) 29. E7+9 k5=4(e9-7,e9+7,e1-3) 30. K5=6 (G5=9) k4+1(a5+6, e9-7,e9+7,e1-3) 31. G5=9 e1-3 32. K6-1(G9+1,P7+1) k4-1(e9-7,e9+7) 33. P7+1 k4=5 (a5+6,e9-7,e9+7,e3+5) 34. A6-5(E9+7,G9=7,G9+1,P7+1,P7=6) a5-4(e9-7,e9+7) 35. G9=7(P7+1) e3+1(e3+5) 36. P7+1(P7=6) a6+5 37. K6=5(E9+7,P7=6) k5=6 38. E9+7 (P7=6) e1-3 39. P7=6 e3+5 40. P6=5 k6+1(a5+4,e9-7,e9+7,e5-7) 41. G7+1(G7+2,G7+3, P5=4) k6-1(e9-7) 42. P5=4 k6=5(k6+1,a5+4,e9-7,e9+7,e5-7) 43. P4=3 k5=6(e9-7,e9+7) 44. A5+4(P3=2) k6=5(a5+4,e9-7,e9+7,e5-7) 45. P3=2 k5=6(a5-6,a5+4,a5+6,e9-7,e5-7, e5+7) 46. P2=1 e9-7(e9+7) 47. K5=4 e5+7 48. G7=4 k6=5 49. G4=3(G4=8,G4=9) e7-5 50. G3=2 k5=6 51. G2+8 k6+1 52. E7-5 e5+3(e5+7) 53. G2-8 k6-1 54. G2=4 k6=5 55. G4=8 a5-6 56. K4=5 a4+5(e7+5,e3-1) 57. K5=6 a5+4(e7+5) 58. G8+8(P1=2) e7+5(e3-5, e3-1) 59. P1=2 a6+5(e5-7,e5+7,e3-1) 60. P2=3 k5=6(e5+7) 61. P3=4 k6+1 62. P4=5 k6-1(e5+7) 63. K6=5(E5+3,P5=6) e5+7 64. K5=4(E5+3,P5=6) e3-1 65. E5+3(P5=6) e7-9 66. P5=6 e9+7 67. G8-8 e7-5 68. P6=5 e1-3 69. G8+8 k6+1 70. P5=4 e5+3(e3+1) 71. A4-5(E3-5) e3+5 72. A5-6(E3-5) k6-1(a5-6,e5-3) 73. K4+1 k6+1(e3-1) 74. K4+1 k6-1(e3-1) 75. K4=5(A6+5) k6=5(e3-1) 76. A6+5 e3-1 77. P4=5 e5+7 78. K5=4 a5+6 79. G8-9 e1+3 80. K4=5(G8=5) k5=6(e3-5,e3-1,e7-5) 81. A5+4(G8=5,G8=4) k6=5 (a4-5,e7-5) 82. G8=6(G8=4) e7-9 83. K5=6 a4-5 84. A4-5(P5=4) k5=6(a5-6,e3-1,e9-7, e9+7) 85. A5-4(P5=4) k6=5(k6+1,e3-5,e3-1,e9-7,e9+7) 86. P5=4 k5=6(e3-5,e9-7) 87. P4=3 k6=5(k6+1,e3-5,e3-1,e9-7,e9+7) 88. G6=5(P3+1) k5=6(e3-5) 89. P3+1 e3-5 (e9-7,e9+7) 90. K6=5 e9-7(e9+7) 91. K5=4 e5-3(e5+3,e5+7) 92. K4-1(E3-1,G5+1) e3+1(e7+9) 93. A4+5(E3-5) k6+1 94. P3+1 k6-1 95. G5+8



more than one same type of pieces in same column, 'F' and 'B' can be used to specify "front" or "back". The last number can be the relative ranks this move take, if the move is within the same column, or it can be the target column, if the target square is in a different column.

6. Conclusion

In this paper, we have described a improved, memory efficient retrograde algorithm. We have also outlined some methods we employed to reduce the size of the Chinese Chess endgame databases. Then we give the results for some of the Chinese chess endgames. One example in detail is the aegp-aaee endgame. The discovery that the aegp-aaee is a winning endgame is very interesting. Our research has shown that human understanding about this endgame is still far from perfect.

And the maximum distance of 95 will certainly be a challenge to human capacity, if not beyond it. This result is likely to shock the Chinese chess community, and makes a significant addition to knowledge of the world's most popular game.

References

[1] Clarke, M. R. B. (1977). A Quantitative Study of King and Pawn against King. Advances in Computer Chess 1, 108-118. Edinburgh University Press, Edinburgh.

[2] Edwards, S. J. (1996). An Examination of the endgame KBBKN. ICCA Journal, Vol. 19, No. 1. 24-32.

[3] Gasser, R. (1996). Solving Nine Men's Morris. Games of No Chance (ed. R. J. Nowakowski), pp. 101-113. MSRI Publications, v29, CUP, Cambridge, England. ISBN 0-521-64652-9.

[4] Herik, H. J. and Herschberg, I. S. (1985). The Construction of an Omniscient endgame data base. ICCA Journal, Vol. 8, No.2, 66-87.

[5] Lake, R. , Schaeffer J., and Lu, P. Solving Large Retrograde Analysis Problems Using a Network of Workstations. Advances in Computer Chess 7, Maastricht, Netherlands, 1994, 135-162.

[6] Nalimov, E. V. and Heinz, E. A. (2000). Space-Efficient Indexing of Endgame Databases for Chess. Advances in Computer Chess 9. (eds. H. J. van den Herik and B. Monien)

[7] Nalimov, E. V., Wirth C. and Haworth G. Mc. C., (2000) KQQKQQ and the Kasparov-World Game, ICCA Journal, Vol 22, No. 4, pp.195-212

[8] Stiller, L. B. (1995). Exploiting Symmetry On Parallel Architectures. Ph.D. thesis. The John Hopkins University, Baltimore, Maryland.

[9] Ströhlein, T. (1970). Untersuchungen über kombinatorische Spiele. Dissertation, Fakultät für Allgemeine Wissenschaften der Technischen Hochschule München.

[10] Thompson, K. (1986). Retrograde Analysis of Certain Endgames. ICCA Journal, Vol. 9, No. 3. 131-139.

[11] Thompson, K. (1996). 6-Piece Endgames.

[12] Wu, R and Beal, D. F. (1993). Retrograde Analysis of some Chinese Chess Endgames. Technical Report. QMW 1993.

[13] Xu, Zhi (1570) Shi Qin Ya Qu.

[14] Chen, Lianrong (1930). Pao Bin Endgames.

Ren Wu
ren_wu@hp.com

Donald F. Beal
Department of Computer Science
Queen Mary & Westfield College
London E1 4NS
United Kingdom
don@dcs.qmw.ac.uk