The Efficiency of the α-β Pruning Implementation in Geniss Axon XP Chess Program

Vladan Vučković [1]

Abstract -**This paper is concerned with the efficiency of the Alfa-Beta pruning technique implemented in author's chess application. The original technique was postulated in Shannon basic works [1],[2] and it is one of the main algorithms for pruning the game (chess) decision tree. The author has implemented the technique in *Geniss Axon XP* and prove that technique works on many test examples. This work contains some empirical results pointing the gross factor of the tree computing optimization according to the theory prediction.**

Keywords - **The theory of games, computer chess, decision trees.**

I. Introduction

The theory of computer chess is complex connecting many sub-domains like theory of games, decision trees, theory of programming etc. The nature of the computer chess problem could be explained very simply: namely, decision tree which is the base of machine chess-playing algorithm grows exponentially with factors depending of position, hash tables, number of pieces on the board … The quality of computer play strongly depends on depth of the decision tree so the effect of the exponential explosion limits the computer chess strength. This problem is much more noticeable when the full-width searching procedure is used. When the pruning techniques, like Alfa-Beta pruning is implemented the tree growth factor drops by significant factor (Section V) but still remain exponential. Hence, it follows that the pruning mechanisms are able only to decrease but not to eliminate the exponential explosion. This paper has intention to make a step in the right direction trying to prune the decision tree as much as possible. The main pruning method is Alfa-Beta and it is implemented in author's *Geniss Axon XP* application with some technical improvements. The results of tests prove that the implementation of the pruning technique is able to cut the large parts of the tree improving the computer playing strength notably.

II. Chess Tree Searching

Chess game is started with 32 pieces (12 different categories) on the 64 square board (Fig. 1). The game are played by two opponents and it consists of the alternate moving from both sides. The main goals of the game is to mate opponent's king, to promote pion to queen, or to achieve enough material advance when opponent must resign.

[1] Vladan Vučković is from the Faculty of Electronic Engineering, University of Ni{, 18000 Ni{, Yugoslavia, Email: vld@elfak.ni.ac.yu

Game also could be drawn when both sides are agreed or when some special situations, like stalemate, appeared on the board. The main problem is to find the best move in every position that might appear on the board. The process of finding the best move implicate searching through the tree of positions **[6],[12],[16],[17]**. At the root of the tree the search for the best successor position for the player to move is performed, at the next level the best successor position from the standpoint of the opponent is searched, and so on.

Decision trees (DTs) are data structures used for handling the machine tree representation of the chess game. The structures of that type are similar to a various of decision diagrams (DD) structures used especially for representation and manipulation of discrete logic functions **[4]**. Decision trees (DT) as a general data structure are very adjustable for problem solving methods in computer treatment of logic games. Each position in chess variant forms one node in chess tree. Because of many possibly exits from the node (20 in start position), multiple decision trees structures (MDTs) have to be used for handling the chess tree.

If all possible moves from each node of the chess tree is processed, this method of tree search is called *full-width* or *Shannon type-A* searching. If one could reduce number of successors from each DT node, tree will grow with less power of exponentially and greater depths could be reached. This approach is called *Shannon type - B* search strategy or selective search **[1],[3],[7],[8],[9],[11],[13],[15]**. The main danger connected with this strategy is that if one prunes some good move from the search process gross error in the move decision process could be obtained.

At each node, the postulate is that side on the move will choose the best continuation calculated among all successors. The white side will choose the continuation with maximum evaluated value and the black side will choose the minimum of the evaluation function. The process is called *minimizing* of the chess tree. If the opponent choose the unsatisfactory successor, the side on the move will have the possibility to increase its evaluation function, so the MiniMax principle is the most rational one to the emulate of the chess thinking process.

III. Alfa-Beta Pruning Technique

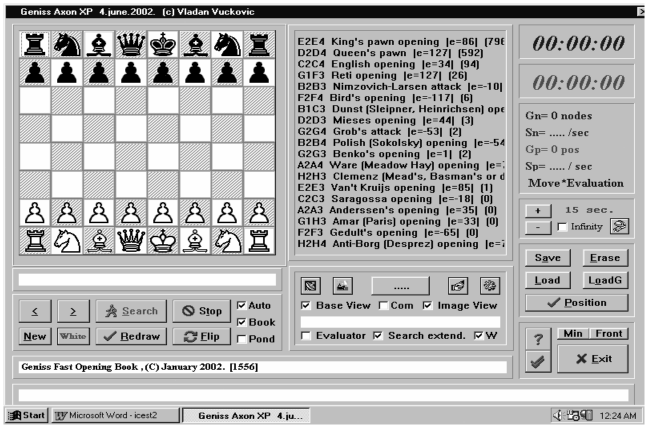
The methods described in previous chapter leading to the situation where number of positions that has to be searched by this algorithms is W^D, where W is the width of the tree (average number of moves possible in every position) and D is the depth of the tree. This is extremely inefficient searching method and some test results postered in Chapter V illustrate this facts. Alpha-Beta search is the significant improvement for reducing the number of positions that has to be searched and thus making greater depths possible in the same amount of time **[3]**,**[5]**,**[10]**,**[14]**. The idea is that in large parts of the tree it is not interesting to find the exact value of a position, but just if it is better or worse than what it has found before. The main hypothesis, proved by Shannon, is that this pruning technique applied on arbitrarily chess tree gives the exactly same key move and evaluation as full-width search with significantly chess tree reducing. The Alpha-Beta search procedure gets two arguments, which indicate the bounds between the exact values for a position is situated. When the evaluation of the specific node exceeded those bounds the searching process is aborted. The version of Alpha-Beta shown in the following is also known as *fail-soft* Alpha-Beta. It can return values outside the range alpha...beta, which can be used as upper or lower bounds if a researching has to be done. The next listing given in pseudo-C illustrates the nucleus of the Alfa-Beta algorithm:

```
int AlphaBeta (pos, depth, alpha, beta)
{
   if (depth == 0) return Evaluate(pos);
   best = -INFINITY;
   succ = Successors(pos);
   while (not Empty(succ) && best < beta)
   {
      pos = RemoveOne(succ);
      if (best > alpha) alpha = best;
      value = -AlphaBeta(pos, depth-1, -beta, -alpha);
      if (value > best) best = value;
   }
   return best;
}
```

The largest gain of the algorithm is reached when at each level of the tree the best successor position is searched first, because this position will either be part of the principal variation or it will cause a cutoff to be as early as possible. Under optimal circumstances Alpha-Beta still has to search $W^{(D+1)/2} + W^{D/2} - 1$ positions. This is much less than classic MiniMax, but still exponential. It allows reaching about twice the depth in the same amount of time. More positions will have to be searched if move ordering is not perfect **[3]**. The advanced implementation of the algorithm (implemented in *Geniss Axon XP*) could save further calculation time.

IV. Geniss Axon XP application

The *Geniss Axon XP* is the latest version of the authors Geniss applications designed to play classical chess. The predecessors are *Geniss E.C.P.* **[18]** and *Geniss Axon 2002*. The author has also developed the *Geniss Mate Solver* **[19]** for the problem chess. The main window of the application is postered in the next figure:



```
E2E4  King's pawn opening  |e=86|  (79
D2D4  Queen's pawn  |e=127|  (592)
C2C4  English opening  |e=34|  (94)
G1F3  Reti opening  |e=127|  (26)
B2B3  Nimzovich-Larsen attack  |e=-10|
F2F4  Bird's opening  |e=-117|  (6)
B1C3  Dunst (Sleipner, Heinrichsen) ope
D2D3  Mieses opening  |e=44|  (3)
G2G4  Grob's attack  |e=-53|  (2)
B2B4  Polish (Sokolsky) opening  |e=-54
G2G3  Benko's opening  |e=1|  (2)
A2A4  Ware (Meadow Hay) opening  |e=
H2H3  Clemenz (Mead's, Basman's or d
E2E3  Van't Kruijs opening  |e=85|  (1)
C2C3  Saragossa opening  |e=-18|  (0)
A2A3  Anderssen's opening  |e=35|  (0)
G1H3  Amar (Paris) opening  |e=33|  (0)
F2F3  Gedult's opening  |e=-65|  (0)
H2H4  Anti-Borg (Desprez) opening  |e=
```

[✓ Base View] Com [✓ Image View]
Evaluator [✓ Search extend.] [✓ W]
[✓ Auto] [✓ Book] Pond

Geniss Fast Opening Book , (C) January 2002. [1556]

Fig 1. The main window of the *Geniss Axon XP* application

The application is used to perform serious of tests with and without Alfa-Beta pruning mechanism. The source code of the application is altered for both test situations.

V. Comparation Results

In this chapter a serious of tests are performed with intention to determinate the empirical data proving the significant search reducing by using Alfa-Beta technique. The *Geniss* application runs on Pentium I 200 MHz machine (64 Mb RAM). The test results are divided into four tables (Tables I-IV) . Tables I and II present the data generated by the full-width searcher with and without search extensions. The notion of *extension* overalls all searching beyond the search horizon - search depth. The extensions usually do some checks, captures or promotion moves. When the extensions are off, the searcher performs the classical Shannon type-A searching procedure (Table II). Tables III and IV content searching results when Alfa-Beta algorithm is set on. The tables columns have the following meanings:

- ***depth*** - search depth (in plys),
- ***positions*** - positions performed in tests,
- ***time*** - time consumption,
- ***move*** - the key move found by searching on specific depth,
- ***evaluation*** - the computed evaluation,
- ***T.G.C.*** (tree growth factor) - this factor shows the quotient between the number of positions performed on successive depths D and (D-1).

The test position is chosen by consulting the theory of chess combinations and it is extracted from the famous Steinitz-Bardeleben game played in Hastings 1895. The key position is diagrammed in Fig 2. The grandmaster Steinitz has played the power sacrifice move E1E7!! leading to the ultimate victory. The following data demonstrate how the machine handle the position:

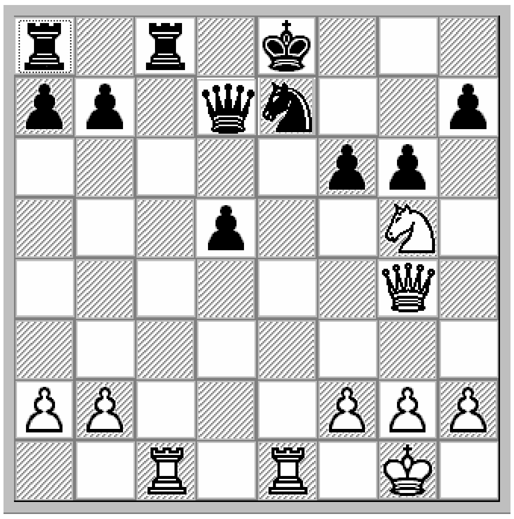


Fig 2. The test position from the Steinitz-Bardeleben game played in Hastings 1895.

TABLE I
TABLE SHOWS COMPARATION RESULTS OF THE POSITION ANALYZING WITHOUT ALFA-BETA PRUNING (WITH EXTENSIONS)

De-pth	Positions	Time	Move	Evaluation	T.G.F
1	6823	1 sec.	G4D7	+0.24	-
2	54237	6 sec.	G4D7	+0.24	7.949
3	245223	28 sec.	G4D7	+0.01	4.520
4	1154725	2:08 sec.	G5H7!	+1.22	4.708
5	3208487	6:59 sec.	G5D7!	-0.05	2.778

TABLE II
TABLE SHOWS COMPARATION RESULTS OF THE POSITION ANALYZING WITHOUT ALFA-BETA PRUNING (NO EXTENSIONS)

De-pth	Positions	Time	Move	Evaluation	T.G.F.
1	1491	<< 1 sec.	G4D7	-0.56	-
2	11275	1 sec.	C1C8	+2.38	7.562
3	60530	8 sec.	G4D7	-0.09	5.368
4	259719	31 sec.	C1C8	+2.27	4.290
5	3208487	6:50 sec.	G4D7	-0.05	12.35

TABLE III
TABLE SHOWS COMPARATION RESULTS OF THE POSITION ANALYZING WITH ALFA-BETA PRUNING (WITH EXTENSIONS)

De-pth	Positions	Time	Move	Evaluation	T.G.F.
1	354	<< 1 sec.	G4D7	+0.24	-
2	1217	<< 1 sec.	G4D7	+0.24	3.437
3	2772	1 sec.	G4D7	+0.01	2.277
4	7074	1 sec.	G5H7!	+1.22	2.551
5	22915	3 sec.	G5H7!	+1.07	3.239
6	58901	6 sec.	G5H7!	+1.04	2.570
7	115213	14 sec.	G5H7!	+0.96	1.956
8	259312	31 sec.	E1E7!!	+2.97	2.250
9	569235	1:07 sec.	E1E7!!	+2.32	2.195
10	1428925	2:43 sec.	E1E7!!	+3.14	2.521

TABLE IV
TABLE SHOWS COMPARATION RESULTS OF THE POSITION ANALYZING WITH ALFA-BETA PRUNING (NO EXTENSIONS)

De-pth	Positions	Time	Move	Evaluation	T.G.F.
1	131	-	G5D7	-0.56	-
2	603	-	C1C8	+2.38	4.603
3	1518	1 sec.	G4D7	-0.09	2.517
4	3168	1 sec.	C1C8	+2.27	2.086
5	12365	2 sec.	G4D7	-0.05	3.903
6	25989	4 sec.	G5H7	+1.07	2.101
7	52843	8 sec.	G5H7	+0.96	2.033
8	99129	14 sec.	G5H7	+1.10	1.876
9	311137	44 sec.	E1E7!	+2.13	3.139
10	581238	1:26 sec.	E1E7!	+3.36	1.868
11	1123945	2:43 sec.	E1E7!	+2.42	1.933
12	2914191	6:42 sec.	E1E7!	+2.98	2.593

The analyzes of the presented data affirm the following conclusions:

- ***Table I*** - The full-width searching with extensions is very stable. The sub-dominant key move G5H7! is found at depth=4. The tree growth is extremely exponential with average T.G.F. 4.980.

- ***Table II*** - The full-width searching without extensions is unstable. The sub-dominant key move G5H7! is found at depths=1,3,7, but altered at other depths. The tree growth is extremely exponential with average T.G.F. 7.390.

- ***Table III*** - The Alfa-Beta full-width searching with extensions is very stable. The sub-dominant key move G5H7! is found at depth=4 and the dominant move E1E7!! found at depth=8 and remain stable in further. The tree growth is exponential with average T.G.F. 2.553 which is significantly lower then classical full-width searcher (Table I).

- ***Table IV*** - The Alfa-Beta full-width searching without extensions is unstable at lower depths but stabilized at depths >=6 . The sub-dominant key move G5H7! is found at depth=6 and the dominant move E1E7!! found at depth=9 and remain stable in further. The tree growth is exponential with average T.G.F. 2.600 which is similar to the previous results (Table III). One could notice that the key moves are found 2 and 1 ply deeper when no extensions are performed.

VI. Conclusion

The Alfa-Beta pruning mechanism represents the fundament of almost all today's chess programs (including author's *Geniss Axon XP*). The main intention of this paper is to present some quantitative data in goal to dispose the precise contribution of the Alfa-Beta to the chess tree searching. The tables and the analyzes in Chapter V affirm that Alfa-Beta has massive tree pruning ability with factors up to 1:140 for searching with extensions (depth=5) and 1:260 for searching without extensions at the same depth. The search extensions add some stability into the searching process. The key moves are found a ply or two earlier. Also, the extensions generate a huge number of positions depending on tactical character of them. For the test performed in previous chapter, at depth=10, the extensions applied about 145% to the searched positions but relieve program to find the solution earlier (31 sec, depth=8 compared to 44 sec, depth=9 without extensions).

The general conclusion is that regular and fast search algorithm must contain Alfa-Beta pruning mechanism implemented in program nucleus supported by capture/check extensions and good move-ordering generator. The classical full-width algorithms without these mechanisms are irrelevant for the modern computer search researching.

References

[1] Claude E. Shannon "Programming a computer for playing chess", first presented at the *National IRE Convention*, March 9, 1949. New York, U.S.A

[2] Claude E. Shannon "A chess-playing machine", reprinted with permission, *Scientific American*, 1950, USA

[3] Peter W. Fray *Chess Skill in Man and Machine*, texts and monographs in computer science, Springer-Verlag, 1977,1978, New York, USA

[4] Milošević, R. S. Stanković, C. Moraga "Fast reordering strategies for BDDs", *University of Dortmund, 2000.*, Germany.

[5] *Analysis of Alpha-Beta Pruning,* http://www.npac.syr.edu/copywrite/pcw/node351.html, 2000.

[6] Allis, L.V. Ingo Althöfer: "On pathology in game tree and other recursion tree models" *ICCA Journal*, Vol. 15, No. 2, p. 80. (1992).

[7] Althöfer, I. Ralph U. Gasser: "Harnessing computational resources for efficient exhaustive search" *ICCA Journal*, Vol. 18, No. 2, pp. 85-86. (1995).

[8] Anantharaman, T.S., Campbell, M.S. and Hsu, F. "Singular extensions: adding selectivity to brute-force searching" *ICCA Journal*, Vol. 11, No. 4, pp. 135-143. (1988).

[9] Anantharaman, T.S. "Confidently selecting a search heuristic" *ICCA Journal*, Vol. 14, No. 1, pp. 3-16. (1991).

[10] Bal, H.E. and Renesse, R. van "A summary of parallel Alpha-Beta search results" *ICCA Journal*, Vol. 9, No. 3, pp. 146-149. (1986).

[11] Bettadapur, P. "Influence of ordering on capture search" *ICCA Journal*, Vol. 9, No. 4, pp. 180-188. (1986).

[12] Bruin, A. de, Pijls, W. and Plaat, A. "Solution trees as a basis for game-tree search" *ICCA Journal*, Vol. 17, No. 4, pp. 207-219. (1994).

[13] Buro, M. "ProbCut: an effective selective extension of the - algorithm" *ICCA Journal*, Vol. 18, No. 2, pp. 71-76. (1995).

[14] Junghanns, A. "Are there practical alternatives to Alpha-Beta?" *ICCA Journal*, Vol. 21, No. 1, pp. 14-32. (1998).

[15] Kaindl, H., Horacek, H. and Wagner, M. "Selective search versus brute force" *ICCA Journal*, Vol. 9, No. 3, pp. 140-145. (1986).

[16] Marsland, T.A. "A review of game-tree pruning" *ICCA Journal*, Vol. 9, No. 1, pp. 3-19. (1986).

[17] Walker, A.N. "Uniqueness in game trees" *ICCA Journal*, Vol. 7, No. 4, pp. 193-202. (1984).

[18] Vuckovic, Vladan "The Basic Elements of the Chess Program Implementation", *Proceedings of a Workshop on Computational Intelligence and Information Technologies*, pg. 17-23, June 20-21. 2001.

[19] Vuckovic, Vladan "Decision Trees and Search Strategies in Chess Problem Solving Applications", *Proceedings of a Workshop on Computational Intelligence and Information Technologies*, pg. 141-159, February 27. 2001.