

Ahmet Shapiro-Erciyas
 Chess AI, MICS
 April/14/2012

The focus of this paper will be geared towards artificial intelligence and chess, and it is for the computer science undergrad that has a familiarity with chess, a novice understanding of computer chess and of general AI gaming algorithms. This paper will serve the purpose of providing the reader with a general view of the essential algorithms for chess engines, their detrimental weaknesses and how today's chess players should try to take advantage of these weaknesses. The research is based on series of tests on a popular chess engine and is geared towards explaining what a chess programs and a human's strengths and weaknesses are. Certain positions from famous human vs. machine matches are analyzed in order to identify a cause and effect relationship between the unique position and chess algorithm. The main algorithms that will be analyzed in this paper will include the Minimax Algorithm, Quiescence Search and Alpha-Beta Pruning. I present these algorithms in relationship to the well-known phenomenon known as the horizon effect and its impact on gaming in general and how alpha beta pruning alleviates some of this effect.

With that in mind let's investigate the essence of the horizon effect. In artificial intelligence, the horizon effect refers to the phenomena in games where the number of possible positions that can be reached becomes so immense, so dramatic that computers can feasibly only search a limited portion of the game tree (typically only a few moves ahead). One of the hopes of this paper is to bring forth to the reader that the general algorithms that are used in solving chess games are not as mysteriously complicated as one may think, but rather they are quite intuitive as most algorithms are. When thinking about them in an abstract perspective, the fundamental workings of these algorithms resembles the journey of a mountain ranger through a series of mountains. As the ranger plans his travel through the mountains, he must figure out exactly where the next valley is where he can cool down for a rest. During a chess game, the chess computer does just that, it looks for as many moves ahead, enough so that at the end of the combination, it will be able to *stop calculating*(rest) in that particular position due to the lack of dynamicity in the position. By dynamicity, I merely mean *the number of plausible looking moves that are available*. In technical terms these positions are also referred to as "quiet" positions. Calculating these resting positions is one of the ways to deal with the horizon effect and is related to the quiescence algorithm. In order to gain a better idea of the above let's build on our knowledge by investigating the Minimax and Alpha-Beta Algorithms.

For a game like chess, the minimax algorithm is such a strategy, which uses the fact that the two players are working towards opposite goals to make predictions about which future states will be reached as the game progresses, and then proceeds accordingly to optimize its chance of victory. The theory behind minimax is that the computers opponent will be trying to minimize whatever value the human is trying to maximize (thus, "minimax"). Hence, the computer should make the move which leaves its opponent capable of doing the least damage on the chess board. In the ideal case where the computer has infinite time and infinite storage capacity, the computer will investigate every game outcome possible from the game's current state. For zero-sum game (win-lose-draw game) like chess, there are only three possible values; 1, -1, and 0 respectively. Then, starting from the bottom of the game tree, the computer evaluates which possible outcome is best for the computer's opponent. The algorithm then assumes that, if that game stage is reached, its opponent will make the move which leads to the outcome best for the opponent (and worse for the computer). Thus, the algorithm can *predict* what its opponent will do, and have a tangible idea of what the final game state will be if that second-to-last position is actually reached. Then, the computer can treat that second-to-last position as a terminal node of that value, even though it is not actually a terminal value. This process can then be repeated at higher levels. Eventually, each option that the computer currently has available can be assigned a value, as if it were a terminal state such that the algorithm simply picks the highest value and takes that action. A relatively straight-forward pseudo code implementation is shown below:

// returns an action

MINIMAX-DECISION (*state*)

1. **return** *the maximum of* MIN-VALUE(**RESULT**(*state*,*a*))

```
//returns a utility value
```

```
MAX-VALUE(state)
```

1. **if** TERMINAL-TEST(*state*)
2. **return** UTILITY(*state*)
3. **set** $v = -\infty$
4. **for each** *a* in ACTIONS(*state*)
5. **set** $v = -\text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$
6. **return** *v*

```
// returns a utility value
```

```
MIN-VALUE(state)
```

1. **if** TERMINAL-TEST(*state*)
2. **return** UTILITY(*state*)
3. **set** $v = \infty$
4. **for each** *a* in ACTIONS(*state*)
5. **set** $v = \text{MIN}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$
6. **return** *v*

The above algorithm will return the action that corresponds to the best possible move (with the best utility). Note again that, this is under the assumption that the opponent player plays to maximize his utility (win-loss evaluator). The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way down to the leaves, to determine the backed-up value of a state.

On the other hand Alpha-beta pruning can through as an improvement over the minimax algorithm. As the reader may have noticed the problem with minimax is that the number of game states it has to examine has an unfavorable exponential growth. While alpha-beta doesn't eliminate the exponential growth factor completely, it does cut it in half. The idea is that it is possible to not have to look at every possible game state in order to come up with the correct minimax decision. This idea is also known as pruning, or eliminating possibilities from consideration without having to examine them, where the algorithm allows us to discard large parts of the tree from consideration. When applied to a standard minimax tree, it returns the same move as minimax would, but in the process prunes away branches that cannot possibly influence the final decision. Alpha-Beta pruning gets its name from the following parameters; α , the value of the best choice found so far at any choice point along the path for MAX, and β , the value of the best choice found so far at any choice point along the path for MIN Below is the pseudo code:

```
// returns the pruned game tree
```

```
ALPHA-BETA PRUNING (node, tree)
```

1. **for a node** *n* somewhere in the tree
2. **if** ((one can move to that node) **and** (there is a better choice
3. either at the parent of the node *n*) **or** (at any point further up))
4. **then** prune *n*
5. **return** *tree*

As mentioned earlier, a chess program must also emulate the planning that the ranger does while planning a trip through the mountains (resting spots in valleys). The conventional algorithm that is used for this is called quiescence search and it is a very important part of any chess program. This is because the algorithm identifies the debt for which to calculate. It does this by identifying states that are "quiet" versus

“interesting”. States that are interesting are ones which have a lot of activity in the position; this characteristic can be defined using different kind of evaluation functions (or heuristics). Ultimately the aim of this algorithm is to make sure that a state at a certain debt is safe enough for the computer to stop evaluating at that debt. It wants to make sure that it doesn't, say, leave its queen in jeopardy. A simple implementation of the algorithm is below:

//returns the estimated value of a node if it is quiet.

QUIESCENCE SEARCH (*node, depth*)

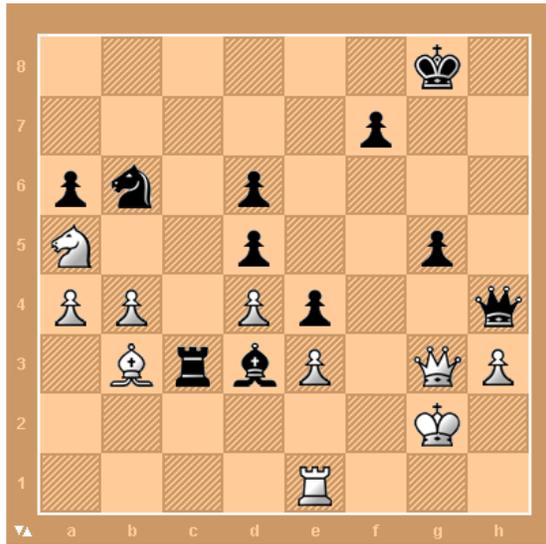
1. **if** *node(appears quiet)* **or** *node(is terminal)* **or** *depth = 0*
 2. **return** *estimated value of node*
 3. **else**
 4. ALPHA BETA SEARCH(*Children of node using recursive applications of quiescence search*)
 5. **return** *estimated value of node*
-

At this point the reader should have a general understanding of how the chess computer comes up with its decisions. We are ready to talk about how to higher our chances against the computer. Before we do this lets introduce a description of the strengths and weakness of computers and humans. First, compared to the chess computer, as humans we can *understand* the position; we can learn more about the game even as we are playing where as computers merely repeat what their algorithms return, and they can only calculate as far as their processor allows them. On the other hand humans can rely on their intuition. Intuition is that gut feeling that one has, and to the novice, it may sound immature to talk about intuition in a game like chess. Though as the greater players such as Kasparov and Carlson (Mozart of chess) will tell you, it is an essential part of the game. There are example games such as Kasparov vs. Topolov (1999 Hoogovens Wijk aan Zee Tournament) where Kasparov had commented about how he made decisions based partly on his instincts, about how he felt about the position. In the book “How Life Imitates Chess” Garry Kasparov talks about he describes it in a verity of contexts on of which highlights the statement “flexing your intuition leads to better decision making”. The source of this gut feeling is gained commonly by experience (and some have it naturally, hence the prodigy). The underlying idea is that great chess players attach stigma to certain kinds of states of the game; these are real feelings that they have towards how a position looks, and this evaluation is completely different than their calculation, but purely based on their intuition. This is different than simply recalling a game from memory and matching the current game state with some other state, but rather it is an unconscious articulation about what the right continuation from the current state is; it is an abstraction of ideas. For the tournament chess player the term that would be used in describing intuition on the chess board would be “strategical play”, strategizing toward long term advantages through identifying possible piece formations that allow themselves to be utilized for relatively long portions of the game. The ability to understand this kind of strategy is something that is still unique to us as human beings and serves as the boundary which separates human-style-chess from computers since this characteristic relies on and can only be formed through intuition and experience, not through any brute force calculation.

This carries us to our next point that as humans, our performance can be greatly influenced if we are not emotionally stable. Thus for a chess match a big factor among many is intimidation, which distracts us away from our ability to use our intuition. The computer on the other hand doesn't have this problem. There is interesting proof of this being a real factor in human vs. machine chess matches. In a match between Kramnik vs. Leko (World Championship Match 2004) Kramnik and his team had prepared a queen sacrifice against Leko (with mainly using computer advice), and Leko was able to refute the move over the board! Even through Leko was playing a “human” it was nevertheless a prepared move by computers, therefore he had refuted a computer but only because he had thought he was playing a human. On a different endeavor in 2006, it was the game Kramnik vs. Deep Fritz where Kramnik lost a game in a mate-in-one (a simple checkmate) which by some was seen as an offensive characteristic for a professional world championship contender to lose so easily. The couple of examples above come to show the detrimental effect of intimidation on human players. In order to further exhaust the realness of this effect, I performed a series of tests. I went on a popular chess website called ICC (Internet Chess Club) and did the

following with a popular chess engine called Fritz 11. I had Fritz play against players of a variety of strengths, for a total of 71 games (10 –min time control). All games were against human players of the site, 36 were against humans who knew that they were playing against a chess engine and 35 were against humans who did not know they were playing against a computer. I adjusted the strength of Fritz for each ICC opponent, and recorded the result of each match up. Players were picked as randomly as possible with a variety of playing strengths and the statistics are as follows. From the 36 games there were played, the ICC player won 14, drew 7 and lost 15 whereas from the 35 games, the ICC player won 18 drew 4 and lost 12. These results implies a winning ratio of 17.5/18.5 (0.9495) for the ICC player with the knowledge of playing a computer, and 20/14 (1.42) for the ICC player who did not know that they were playing against the computer. Therefore, since the rating system (strength evaluator) of the ICC server is shared by all of the players on ICC site, and that the Fritz computer strength was adjusted for each player, these statistics implies that the human player tends to play better against the computer when they don't know that they are in fact playing a computer.

Additionally to the point about intimidation above (which is don't be intimidated just because you are playing a computer), one of the most intrusive ways to tackle chess engines is through exploiting their quiescence search. Recalling back to the rangers journey through the mountains, one can exploit quiescence search by finding a calculation sequence that has seemingly quiet moves but that the computer will blunder on. The idea is that if the ranger is the computer program, its equivalent to saying that the ranger miscalculates his next resting spot, and in reality the valley happens to be still ways away. A famous game between Ponomariov vs. Fritz (Bilbao 2006) exemplifies this point (see below).



event	date				
2nd Festival Internacional	2005.11.21				
site	rnd	score	eco		
Bilbao ESP	2	1-0	A45		
white	rating				
Ruslan Ponomariov	2704				
black	rating				
Fritz (Computer)	?				
1	d4	Nf6	21	Re1	h4
2	c3	d5	22	h3	Rb7
3	Bf4	Bf5	23	Na5	Rbb8
4	e3	e6	24	Ba4	a6
5	Qb3	Nbd7	25	Bb3	Nb6
6	Qxb7	Bd6	26	Qb2	Qd7
7	Bxd6	cxd6	27	a3	Rc7
8	Qa6	Rb8	28	Qa2	Rbc8
9	Qa3	Qb6	29	Nf4	Qf5
10	b4	0-0	30	a4	Bd3
11	Nd2	e5	31	g4	hxg3
12	Ngf3	Qc7	32	fxg3	g5
13	Ba6	e4	33	g4	Qh7
14	Ng1	Rb6	34	Nh5	Nxh5
15	Rc1	Nb8	35	gxh5	Qxh5
16	Be2	Rc8	36	Qh2	Qh4
17	Bd1	Bd7	37	Kg2	Rxc3
18	Ne2	Bb5	38	Rxc3	Rxc3
19	0-0	Nbd7	39	Qg3	Bc2
20	Nb3	h5	40	Qxh4	gxh4

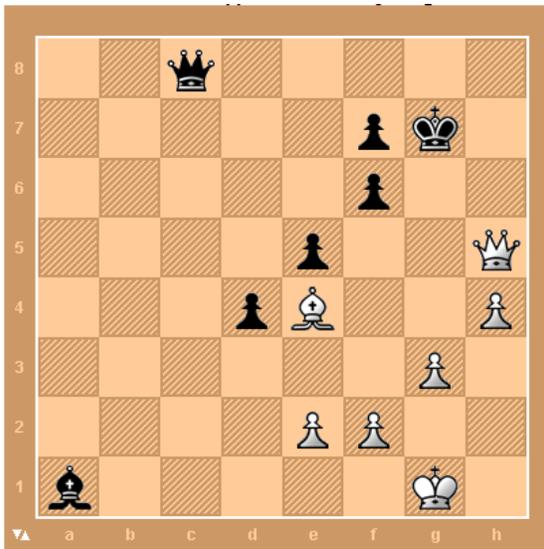
Fritz moves **39)** ... Bc2, seemingly winning a piece after Ponomariov plays **40)** Qxh4 gxh4, **41)** Rc1 Rxb3, **42)** Nxb3 Bxb3, **43)** a5 Nc5

event	2nd Festival Internacional		date	2005.11.21	
site	Bilbao ESP		rnd	score	eco
white	Ruslan Ponomarev		rating	2704	
black	Fritz (Computer)		rating	?	
41	Rc1	Rxb3	61		
42	Nxb3	Bxb3	62		
43	a5	Nc4	63		
44	b5	Ba4	64		
45	bxa6	Bc6	65		
46	a7	Kg7	66		
47	a6	Ba8	67		
48	Rb1	1-0	68		
49			69		
50			70		
51			71		
52			72		
53			73		
54			74		
55			75		
56			76		
57			77		
58			78		
59			79		
60			80		

This is the point that registers as “quiet” for quiescence search, but in fact it is not, since after the forced combination 44) b5 Ba4, 45) ba6 Bc6, 46) a7 Kg7, 47) a6 Ba8, 48) Bb1 black will have to give up both of its pieces in order to stop the two passed on the a-file after 49) Rb8, blacks king is too far away to help.

event	2nd Festival Internacional		date	2005.11.21	
site	Bilbao ESP		rnd	score	eco
white	Ruslan Ponomarev		rating	2704	
black	Fritz (Computer)		rating	?	
41	Rc1	Rxb3	61		
42	Nxb3	Bxb3	62		
43	a5	Nc4	63		
44	b5	Ba4	64		
45	bxa6	Bc6	65		
46	a7	Kg7	66		
47	a6	Ba8	67		
48	Rb1	1-0	68		
49			69		
50			70		
51			71		
52			72		
53			73		
54			74		
55			75		
56			76		
57			77		
58			78		
59			79		
60			80		

Example of strategical advantage: The first match of Kasparov’s 1996 game series against Deep Blue (see below after Kasparov with the white pieces plays 33. Qh5).



event	Match		date	1996.?? ??	
site	Philadelphia (USA)		rnd	score	eco
			2	1-0	A49
white	Garry Kasparov			rating	?
black	Deep Blue (Computer)			rating	?
1	Nf3	d5	21	Qd7	Qc8
2	d4	e6	22	Qxa7	Rb8
3	g3	c5	23	Qa4	Bc3
4	Bg2	Nc6	24	Rxb8	Qxb8
5	0-0	Nf6	25	Be4	Qc7
6	c4	dx4	26	Qa6	Kg7
7	Ne5	Bd7	27	Qd3	Rb8
8	Na3	cx4	28	Bxh7	Rb2
9	Naxc4	Bc5	29	Be4	Rxa2
10	Qb3	0-0	30	h4	Qc8
11	Qxb7	Nxe5	31	Qf3	Ra1
12	Nxe5	Rb8	32	Rxa1	Bxa1
13	Qf3	Bd6	33	Qh5	Qh8
14	Nc6	Bxc6	34	Qg4+	Kf8
15	Qxc6	e5	35	Qc8+	Kg7
16	Rb1	Rb6	36	Qg4+	Kf8
17	Qa4	Qb8	37	Bd5	Ke7
18	Bg5	Be7	38	Bc6	Kf8
19	b4	Bxb4	39	Bd5	Ke7
20	Bxf6	gxf6	40	Qf3	Bc3

A long term advantage here is present for white due to white's powerful bishop in the center of the board which occupies a lot of space (relative to black's poor bishop who occupies only 3 square). An additional advantage is white's passed pawn on the h-file (versus black's doubled pawns on the f-file). As it is too late to undo these positional mistakes, soon Kasparov will be able to utilize the misplaced pieces in black's structure to force Deep Blue to make a mistake. The key point here is that the computer's lack the ability to understand the implications of these piece structures, and so as humans that is our strong point.

Conclusively, throughout this paper I have analyzed the essential algorithms used for computer chess. We have concluded that there still may exist room for human players in the competition against chess computers. The research was based on series of tests on a popular chess engine and its performance against players on a popular chess engine. There was also analysis of famous human vs. computer chess matches that were exemplified to explain the effects of essential chess algorithms. Towards the end I discussed what humans and computers strengths and weaknesses were. Through the research I have concluded that there will always be room for development in computers ability to mimic the positional understanding that humans have on the chess board. Additionally I've found that intuition is what still separates humans from computers in chess and in artificial intelligence. What I also found was that intimidation plays a considerably important role in how chess games are played, especially against computers. Further research questions may include; "Can today's computers can beat by the top humans today when humans are not in an intimidated state?", "How much of a role does intimidation have in human versus human chess?", and "What does it take to reach this potent state of confidence by a human?".

Works Sited

Russell, Stuart, and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. 2009. Print.

Kasparov, Garry, and Mig Greengard. *How Life Imitates Chess*. 1st. New York: Bloomsbury, 2007. 120-130. Print.

. N.p., n.d. Web. 2 March 2012. <www.chessgames.com>.

"Artificial Intelligence." Elsevier. 43.1 (2012): 85-98. Print.

