An Examination of Brute Force Intelligence

Hans J. Berliner
Carnegie-Mellon University
Pittsburgh, Pa. 15213

Abstract

We here examine the current state of brute-force chess programs. We are interested in their strong points and how these are achieved, and in their weak points and what can be done about them. We compare some excellent play by Belle, the current World Computer Chess Champion, with what expert humans would do in the same situation. These comparisons show that such programs already are capable of displaying what is usually called imagination and understanding. Finally, we examine the prospects for a brute-force program in championship level play.

Introduction

In 1976, CHESS 4.5, the Northwestern University Chess Program, played in an all Class "B" tournament in California. It's rating at the time was about 1700, middle Class "B". To everyone's surprise, the program romped away with the first prize with a 5-0 score, decimating its opponents on the way. The reason for this later became apparent; it was running on a machine that was 10 times faster than the one it usually ran on. This made it possible to search on average a little more that one ply deeper. This resulted in at least a 200 point (one grade) improvement in its performance.

From 1973 to 1975, Slate & Atkin at Northwestern University [5] had been perfecting the techniques required to make a depth-first alpha-beta search into the powerful brute-force searching tool that it is today. Two devices and their interaction were responsible for this advance. Firstly, there was iterative deepening. A depth-first search must have a maximum depth specified in order for it to be able to halt. In iterative deepening, the search starts with a maximum depth of 2, and then iterates to depth 3, 4, etc., as long there is time. At first sight this appears to be very wasteful of computing time. However, the introduction of the second device, a hash table, changes this. The hash table retains two important pieces of information about any node (within the limits of the table size) visited in the search:

1. It remembers the most effective move tried at that node. When such a move is stored from a depth N search, it can now be tried first whenever the node is reached on a depth N+i search. If such a move is successful in terminating search at this node without examining any other alternatives, then not only has the search been brought to a quick conclusion, but the cost of doing a move generation at that node is also avoided. This happens about 70% of the time.
2. Upon quitting a node, its value is written into the hash table. This may be an exact value for this depth of search, an upper bound on that value, or a lower bound. It may only be a bound because often it is only neccessary to determine that the node is at least this good, and the exact value need not be determined. If, in this or future iterations, the node is again encountered with the same depth of search remaining, then a great deal is already known about its value and this may suffice to terminate the search at this node, or reduce the remaining effort by further constraining the value that the node may take. Because the tree is really a graph, two nodes in such a tree may coincide. For instance, the moves A, B, C in one branch, if produced in order C, B, A in another, may result in identical nodes.

The combination of these two techniques made possible searches that were faster than what is theoretically possible with alpha-beta. This is due to the fact that knowing the best move from a previous iteration produces an ordering of nodes that comes close enough to being optimum to make alpha-beta work near its maximum possible effectiveness. Further, the detection of identical nodes produces another exponential saving, that is especially significant in deeper searches.

These advances in search theory together with faster hardware made brute-force searching into a very powerful tool. Clearly, the deeper one could look, the better the program would play. However, it was not clear whether advances in speed of the magnitude required to really make a big difference were possible. At about this time, Greenblatt at MIT and Thompson at Bell Labs started building special purpose chess machines, while others began to look for the fastest machine they might gain access to. Thus the race for speed was on. The undisputed leader at the moment is Belle, the creation of Ken Thompson and Joe Condon of Bell Telephone Labs [6]. It searches about 30 million nodes in the time allowed for one tournament move (about 150 secs.). It does this with special purpose chess hardware for move generation and some evaluation, and a high degree of parallelism. With its ability to look ahead at least 8 ply, plus all captures from any leaf node, we have learned that a great deal that was considered "perceptual" can actually be discovered by brute force. In fact, the program is so good at calculating variations that I doubt that anyone in the world could equal it in complicated positions where accuracy of calculation is required. It executes an evaluation function at leaf nodes that is similar to that of the Northwestern program [5], which has knowledge at about the level of a class "C" player's understanding. However, such knowledge applied 8 ply down the tree appears to be sufficient to generate at

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-78-C-1551.

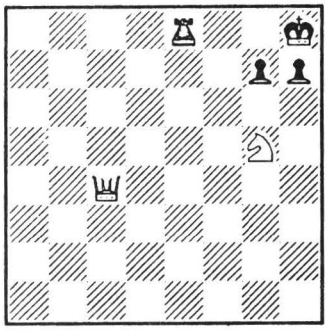

Figure 1
White to Play

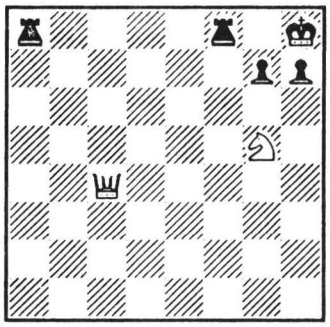

Figure 2
White to Play

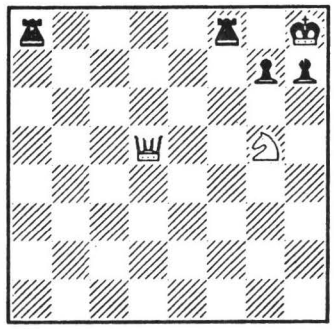

Figure 3
White to Play

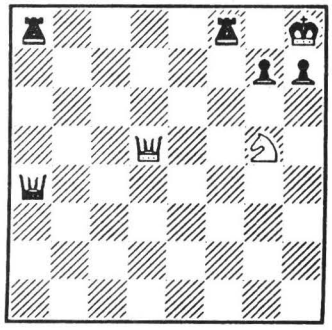

Figure 4
White to Play

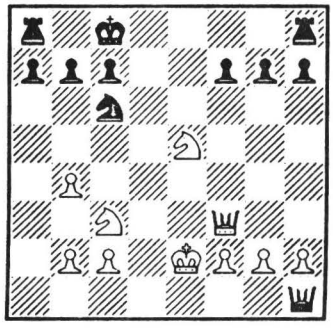

Figure 5
White to Play

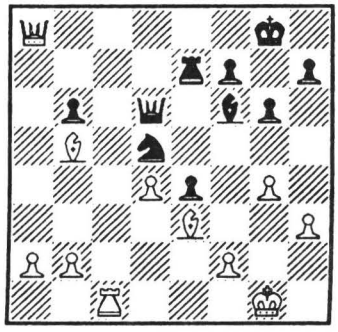

Figure 6
Black to Play

least expert level concepts a high percentage of the time. In fact, the latest incarnation of Belle has achieved a performance rating of 2328 (high master) against humans in serious competition. Thus despite its lack of "conceptualization", Belle is able to perform among the top 300 player in the U. S. We now examine this phenomenon, and the remaining shortcomings of the brute-force approach.[1]

Pattern Knowledge in Chess

Figure 1 shows a basic chess situation known as "Philador's Legacy". White to play can mate in 4 moves by 1. N-B7ch, K-N1, 2. N-R6ch!, K-R1 (K-B1, Q-B7mate), 3. Q-N8ch!, RxQ, 4. N-B7mate. This is all very well known, and I doubt there are many class "B" chess players that do not know this particular pattern; i.e. it is fundamental. In this case it leads to mate; yet under slightly differing conditions the mate can be avoided at the cost of some material. For instance, in Figure 2 White only wins rook for knight (the *exchange*), while in Figure 3 he again mates, while in Figure 4, he again wins the exchange. It would be possible to conceptualize this configuration by indicating that the queen must be on squares K6, Q5, QB4, QN3, or QR2, the knight on KN5, K5, Q6, or Q8, the black king on KR1 and that there must be a black pawn on his KR2 and KN2. However, this is only part of such a description. Actually, the KR2 and KN2 squares must either be controlled by white men (but not the knight or queen) or be occupied by black men that cannot capture on KN1 or KB2. Further, the square KN1, if defended, must be defended by a piece that would not control KB2 when it captured on KN1; i.e. it cannot be a queen. So there are a number of non-trivial tests to determine whether such a situation as Philador's Legacy pertains; yet, this is among the first 1000 or so tactical patterns that a chess player learns. Certainly, top players have many tens of thousands of such patterns stored [4]. Further, the same pattern could occur around a different focus (the location of the losing king) as in Figure 5 where the whole configuration is shifted to the other side of the board and is one file more distant from the edge. This is the most general pattern of this class and requires several additional specifications. Further, the exact defense status of the square on which the knight checks and of the back rank make the difference between mating and only winning the exchange; the latter being paltry if this were a position for which material had already been sacrificed.

Thus, we have our prototypical quandary: to generate patterns to drive a tree search is very complicated, and to attempt to detect the presence of any of thousands of such patterns is probably impossible in any real time environment that is not highly parallelized (we conjecture that humans detect such patterns at the visual level where parallel, array type processing is known to exist). However, a brute force search would have to look ahead 7 ply in order to find the win. Actually, the best programs now do not count responses to check as a ply of depth, so it only requires a 4 ply search. However, since such opportunites must usually be cultivated by previous play, it would seem that a brute-force program to be able to detect such a possibility would have to be searching at least 7 ply deep. Belle certainly fulfills this requirement. The 4 ply search required to find the Philador mate in a full blown position would take Belle less than 1 second. Wilkens [7] developed a highly sophisticated and efficient pattern mechanism for his program PARADISE, yet it would not be able to go through a very minimal set of patterns in anything approaching such time, not to speak of actually completing the search to verify

[1] The experiences with respect to Belle cited herein were derived as part of an informal cooperative arrangement with the authors of Belle, during which I suggested certain tests, the results of which were presumably useful to all parties concerned

that everything works. Clearly, Belle has a distinct advantage in this comparision, having specialized hardware. Possibly, with special pattern detecting hardware such operations could be considerably speeded up. However, then we would have to have some guarantee of near completeness; i.e. the brute force search will find anything within its depth, whereas the patterns, though possibly able to detect deeper ideas, may be far from complete. In fact, we have now seen about 20 games from the new Belle and have found 3 or 4 ideas in these games that we have never seen before anywhere. Thus, there is good reason to believe that such a searching program will outperform an idea driven program on shallow ideas (and we consider the above example shallow).

Examples of Brute Force "Imagination" and "Understanding"

That humans are not the only ones that can have imagination in chess first became very clear to me as the result of the game shown in Figure 6. This was the position after White's 34th move in a game DUCHESS - KAISSA from the 2nd World Computer Chess Championship, Toronto, 1977. KAISSA, the defending champion, here played R-K1, losing a rook. There was a great deal of amazement that this fine program should make such a terrible blunder. Among the lamenters were former World Chess Champion (and very likely the greatest player of all time) Mikhail Botwinnik, who was with the Soviet delegation accompanying KAISSA, and several very strong Canadian players along with master Levy who was commenting on the games. All assumed that 34.-- R-K1 was the result of a program bug, as had been seen in programs over and over again. It was not until the next day, after the programmers had had a chance to go over their debug listings that the reason for the move R-K1 became known. To everyone's surprise, KAISSA had made this move to avoid getting mated! After the obvious 34. -- K-N2, would come 35. Q-B8ch!! (the move everyone overlooked), KxQ, 36. B-R6ch, ANY, 37. R-B8ch and mate in two more moves. Clearly, both programs saw what was coming, and a collection of human masters did not. In a way the masters were right; R-K1 is a blunder, as losing a rook in this position is no different from being mated. However, the important point remained; not a single player saw what was coming. It is possible that because the idea was 10 ply deep, they did not anticipate that the programs were up to such things. However, the correct view of the situation was "Black is lost" not "Black made a blunder", and no human spectator saw it correctly.

In the above example, once a strong player is shown the move 35. Q-B8ch, he says "obviously" and is apologetic for not having seen it himself right away. However, the remaining examples are much more sophisticated and require quite a bit of study to understand them as they do not fall into obvious patterns. Consider the position in Figure 7 (from a game Belle-Fry, Virginia Open, 1981) and the game continuation. Here, White would like to take the bishop at QB8. However, he cannot now or after RxRch, because with QxR Black renews the threat of mate on the back rank, thus forcing an ending in which he has some chances of survival. When I first saw the game, I knew "something might be up" in the diagram position, but could not find it. Since then, I have shown this and the following positions to a number of experts and masters and none have been able to produce the right move or even the right idea in the 5 or so minutes they had to try. Yet, when playing 30. B-B2, Belle saw it could drive the opponent's king up the board, and one move later it realized that it could win queen for rook and bishop by continuing on the course it chose. 30. B-B2 counters the mate threat and thus reactivates the attack on the black bishop, besides threatening B-Q4 ch which

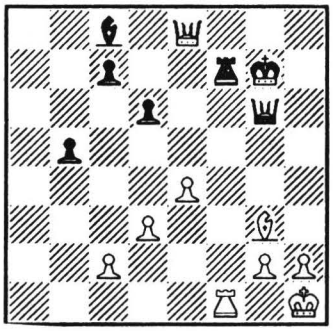

Figure 7
White to Play

The game continued:

30.	B-B2!	B-K3!
31.	B-Q4ch	K-R3
32.	B-B6!!	B-R7
33.	Q-R8ch	Q-R2
34.	Q-Q8	Q-N3
35.	R-B4	Q-R4
36.	Q-R8ch	R-R2
37.	B-N7ch	K-N4
38.	R-B5ch	K-N5
39.	P-R3ch	QxPch
40.	PxQch	RxPch
41.	QxRch	KxQ
42.	RxP	Resigns

would win the rook; thus Black's reply is forced. Further, 32. B-B6 is also sensational. It threatens the bishop, and the white bishop cannot be taken because of 33. Q-R8ch, K-N4, 34. P-R4ch winning the rook. Finally, at move 33, R-B3 wins easily because the threat of R-R3 cannot be met effectively (this is what Belle saw at move 31). However, Belle realized that Black could then put up a modicum of resistance by playing 33.-- QxB, 34. RxQch, RxR, and instead played 33. Q-R8ch, Q-R2 (forced) 34. Q-Q8!. Now catastropy is unavoidable; however, Belle had to calculate among other lines, one that it took me 1/2 hour of moving the pieces around to find: 34.-- R-Q2, 35. B-N5ch, K-R4!, 36. P-N4ch!, KxP, 37. R-B4ch, K-R6, 38. QxRch, QxQ, 39. R-R4. This had to be seen at move 33 by White. In a sense this is guilding the lily, because a sure win is there and no master in his right mind would do such calculation unneccessarily. However, the depth of this analysis shows the potential of the machine in such situations.

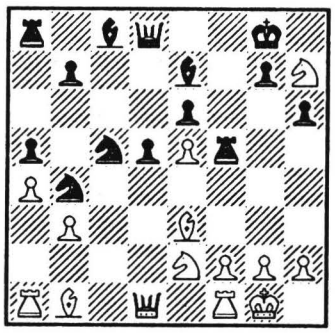

Figure 8
White to Play

The game continued:

20.	P-B4	N-K5
21.	N-KB6ch	BxN
	(if PxN, P-N4)	
22.	PxB	QxP
23.	N-Q4	N-B6
24.	Q-B3	NxB
25.	QRxN	P-K4
26.	NxR	P-K5
27.	NxPch and White won.	

Another example is shown in Figure 8 from a game Belle - Mess, New Jersey Masters Open, 1981. Here it looks as if White will lose his knight at KR7 (it went there on the last move when Belle clearly saw what was coming). This opinion is reinforced by examining the variation 20. P-KN4, RxKP when the knight remains trapped. However, Belle calmly plays 20. P-KB4, relying on 20.-- KxN?, 21. P-KN4. However, after 20.-- N-K5 it again looks bad because 21. P-KN4 is answered by R-B2 when the knight is lost. However, Belle played 21. N-B6ch and, as the game continuation shows, ended up winning the exchange; most unexpected (to a human) and disconcerting.

The final example of this type, Figure 9, comes from a 30 moves in 30 minutes game Valvo -Belle. The former has a rating above 2400, which is Senior Master. Here Belle is Black and calmly plays 16.-- BxP!!. White still did not see what was coming although it was only one move away and played 17. RxB? (it's protected isn't

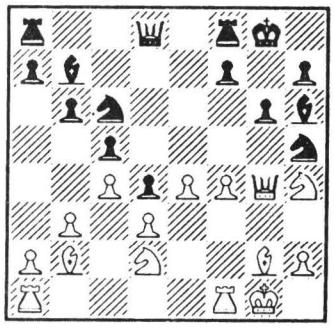

Figure 9
Black to Play

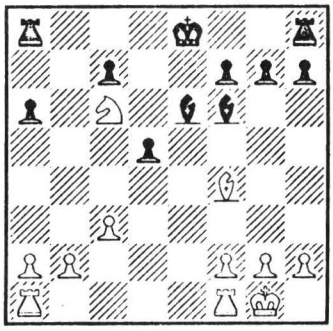

Figure 10
White to Play

The game proceeded:

18.	KR-K1	0-0	31.	R-R5	K-N1
19.	B-K5	BxB	32.	R/3-K5	P-QB3
20.	N-K7 ch	K-R1	33.	R/K-N5 ch	K-R1
21.	RxB	KR-Q1	34.	R-R6	R-KN1
22.	QR-K1	R-Q2	35.	R/5-R5	R-KN3
23.	N-B6	R-Q3	36.	RxP ch	K-N1
24.	N-Q4	R-K1	37.	R-R8 ch	K-N2
25.	P-QN4	P-N3	38.	R/5-R7 ch	K-B3
26.	P-KB4	K-N1	39.	R-R8	K-K4
27.	P-B5	PxP	40.	RxRP	K-K5
28.	NxP	R-B3	41.	RxQBP	RxR
29.	R/1-K3	K-B1	42.	NxR and White won.	
30.	N-Q4	R-Q3			

it). Now came 17.-- N-K4, whereupon he gave up because he must lose back a whole rook now and more later.

It should be noted that it is not at all difficult to describe what happened in the above examples *ex post facto*. This is done using the accepted language of functions (i.e. piece X performs function Y on square Z) which is an excellent descriptive method used by good players and evidenced in their protocols. The fact that good players have extreme difficulty with the above examples would indicate that this language of functions does not play a very strong role in the analysis that drives the search. Rather, the process of qualifying moves for searching would appear to be based on familiar patterns.

Depth 8 searching (11 or so in most endgames) can also produce remarkable positional chess even when coupled with a class "C" evaluation function. Witness the position in Figure 10 that occured between Belle and Gibson (a U. S. Chess Federation expert) as part of the Fredkin prize matches [8]. The position is rather even, and Black should keep his king in the center with satisfactory results. The only thing he really should not allow is an endgame with his bad (light squared) bishop against the white knight. It should be noted that Belle has no information allowing it to determine when one equi-valued piece is better than another. Given the above, consider how Belle maneuvers to get just this advantage whereas the expert appears to be unaware of the need to counter this strategy. Note how 25. P-QN4 is played to keep its knight secure, and how Belle keeps its grip on the dark squares while maneuvering to take over the whole board. There are probably one or two minor inaccuracies in White's play. However, even the best players don't play such positions perfectly at the board, and compared to Black's efforts at countering, White clearly dominates with effort to spare. One is led to wonder how a program with Class "C" chess knowledge can play like this. The answer is that the concept space control (which I believe is the primary factor in all the above) is applied many ply from the root. The move that maximizes space control over such a long span, also results in doing the right things with one's own pieces and making it difficult for the opponent to do so with his.

Belle was also tested on the 300 positions in *Win at Chess* [3], and turned in a surprising performance. It only got 19.5 wrong (.5 credit is given when the correct move is tendered but the supporting analysis is not all present) out of the set. According to the compiler of the volume, a master could expect to get about 30 wrong. However, the most surprising thing was that Belle discovered 9 errors in the solutions presented by the author, only 2 of which were previously known. This is certainly a convincing performance of what brute-force at depth 8 can do in chess.

Some Remaining Weaknesses of the Brute Force Approach

On the other hand, Belle does not always perform like this. It has considerable problems at times in situations that are relatively simple strategically, but require some long term plan (certain long term plans are found by merely following one's nose; others require some conceptualization and possibly reasoning). Figure 11 shows a position from the other Gibson-Belle game of the match, where Black (Belle) has established a winning position. Even good players would not be able to tell at first sight whether this position is a win, because with bishops on opposite colors it is frequently difficult or impossible to win with a 2 pawn advantage. However, even a weak player will understand that it is important to move the black king to a more active location. To this end, he will almost certainly try K-B4 followed by K-N5, and when White plays B-K6 to defend the pawn he will probably play K-B6. Now White will be in a quandary. His bishop cannot move without allowing a pawn to be lost or the king-side pawns to advance. And if White plays K-B3, then K-Q5, B-B5 when Black plays B-B8 and the king-pawn advances by means of the tactical threat P-K5ch (BxP, P-N5ch wins). The latter would be child's play for Belle were the king already at QB6; however, it saw no advantage in heading in that direction, and the game was ultimately drawn.

The above problem would be easy to remedy by simply doing a static analysis at the root of the search tree and marking those squares that would be desirable for each kind of piece. This type of devise has already been used with success by the Northwestern University chess program, although in this case it also failed to find the win.

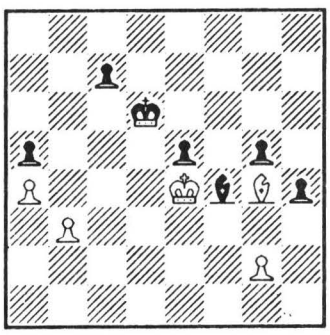

Figure 11
Black to Play

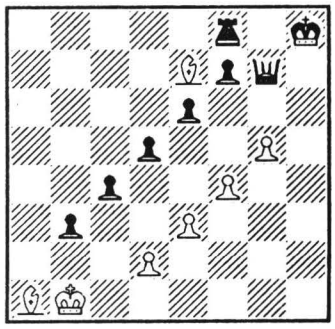

Figure 12
Black to Play

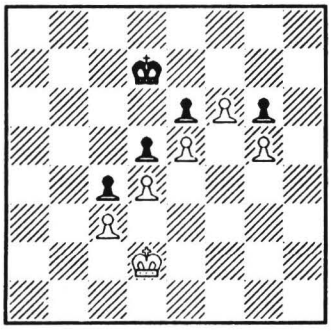

Figure 13
White to Play

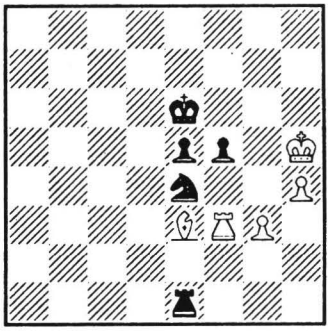

Figure 14
Black to Play

There are a number of cases where Belle, playing against masters and experts, really did not understand the position and drifted somewhat. However, as soon as the opponent made a real threat, Belle was there on the job, defending itself. It is significant, that of the 5 or so games in which this occurred, Belle did not lose a single one by being outplayed in the usual sense; i.e. by having its disadvantage gradually pushed down its throat. Besides the games Belle has lost where it was trying to win something that should have been left alone (see below), the only other games it has lost stem from some gross mis-evaluation of a position. It seems doubtful that such cases can be resolved easily by deeper searches; rather some method of detecting the problem would appear to be needed in the evaluation function.

A Re-Examination of Earlier Precepts

In 1973 I examined some problems that a program would have to overcome in order to play master level chess [1]. Since such a program now exists, it would seem appropriate to determine to what extent this has been accomplished in the manner that was said to be necessary.

A major issue related to the horizon effect and how it could cause grievous errors to be made. Belle makes no attempt to overcome the horizon effect. However, the horizon effect in an 8 ply search does not appear to cause significant problems. This is because delaying tactics require a pair of moves for each delay; one for the threat and one for the counter. Such delaying tactics are bad when they lose material or otherwise worsen the position. If material is involved, the minimum transaction unit is likely to be a pawn, and in 8 ply, 4 pawns or more would have to be sacrificed to push a threat over the horizon. This is unlikely to occur naturally. Further, the cost of the "saving maneuver" at such depth very likely is going to be greater than the original loss that is being prevented. The fact that the horizon effect is still there can be demonstrated from Figure 12, in which an 8 ply searching program playing Black will duly play 1.-- P-N7, delaying the loss of the queen. However, this example is contrived, and such situations are very unlikely to occur in an actual game. However, the horizon effect can and does occur at the end of long forced variations (see below). This only causes problems when the program relies heavily in making its move, on the branch in which the horizon effect occurs. Again, this does not occur often.

Another problem was the program's need to have a global strategy at times. We presented a pawn endgame (Figure 13) that required 13 ply of search to resolve and thus conjectured the need for a global viewpoint since such searches were then not possible, or likely to be in the foreseeable future. However, this has changed drastically. Problems such as this can now be solved easily by the best programs because the hash table detects identical positions in the tree, and terminates the search at nodes for which the value has already been computed. Thus in endgames of few moves, searching deeply is trivial for Belle and other programs too. However, as Figure 11 showed, strategy is still a real problem for programs, and one that will certainly have to be dealt with if the program is to be able to cash in its advantages in most endgames.

We also presented a position that required a 19 ply search to find the mating combination. This was intended to show the need for precise and deep calculation. No performance program can at present do this, although they are not too far away. However, the examples in Figures 7 and 8 are convincing proof that Belle can play tactics with the best humans. It may not be able to deal with long thin lines of play as well, but it deals with medium long, bushy lines with incredible effectiveness. It is not at all clear that it is more important to deal with long, thin lines of play better than with bushy lines; rather the reverse is probably true. Thus we must consider this task as more than satisfactorily met.

Issues in efficiency of searches were examined to show that a great deal of effort would be wasted by a searching program without the proper knowledge. Again, because of the efficiency of iterative deepening and the hash table, such problem have evaporated since the programs just go through variations at an incredible rate. Attempts to incorporate knowledge mechanisms into the framework of searching have, to date, been tedious, subject to certain errors of omission, and not even guaranteed to be faster.

Finally, we indicated that programs would need more detailed evaluation of terminal patterns than they have at present. This is still valid, and is now the largest cause of Belle losing games. In Figure 14, from a game McKenna - Belle, Virginia Open, 1981, Belle has finally equalized the position after a long struggle. Now, it sees the opportunity to win a pawn and proceeds with 56.-- R-KB8??, unable to see that one of the terminal configurations in which it is ahead in material is clearly lost. Belle was searching to a depth of 10 ply when making this move. The game continued 57. RxR, NxPch, 58. K-N6, NxR, 59. P-R5!, NxB, 60. P-R6 and the pawn will queen winning the game. The line given above has consumed 7 ply so far (the response to check not being counted), and it is not possible to detect the queening of the pawn in the remaining three ply. Thus, after 60.-- N-Q4, 61. P-R7, N-K2ch, 62. K-N7, the 10 ply are up, and Belle must evaluate the position. Its evaluation function is not sensitive enough to notice that the white pawn cannot be prevented from promoting, so it judges the position as favorable for Black. It is interesting to note that if the search were being conducted one ply deeper, it would still not solve the problem correctly. We would then have

62.-- N-N3, and White may start sequences of captures in the quiescence search. However, Black will end up a pawn ahead in any case, and, unless the evaluation function judges the pawn on the 7th rank to be worth more than 2 connected passed pawns on the 4th rank, the wrong decision will still be made. Here, however, the problem would be due to the horizon effect (see above) as 62.-- N-N3 is a delaying action that sacrifices a knight to push the queening of the pawn over the horizon.

In a game with Belle, I discovered a strategy that can be used against a program that plays tactics much better than it evaluates. I intentionally let it win 2 knights for rook and pawn (a small material advantage) in order to reach a very superior endgame. The point here is that in considering a position in which it is ahead in material but behind in positional factors, it may fail to evaluate the trade-off properly. In this case, the positional advantage was worth much more than the material that it gained. The McKenna game above, could also be seen in this light, although it seems doubtful that he planned it that way.

The Prospects for Brute-Force Chess

Brute-force programs win because they have a good mix of depth of search, and knowledge applied at leaf nodes. The first brute-force programs [2] only counted material at leaf nodes. Then it became evident that even small amounts of knowledge provided programs with a sense of direction when there was no material gain to be had or defended against. The full-width search is important to ensure that all alternatives are examined, but knowledge assures that small advantages will be striven for. Programs that search one ply or so deeper than the Northwestern University program but evaluate less finely have historically lost to it. This makes sense when one considers that the concepts in the evaluation function, no matter how primitive, do produce an added projection of several ply on the leaf position being evaluated. The importance of knowledge is also substantiated by results from computer Othello. In two recent tournaments where all programs did brute-force searches, the winning program was not the one that searched deepest but the one that evaluated best. In both contests a near perfect transitive ordering could be established among the participating programs, in terms of who beat who. This would appear to indicate that whatever each knew, it was enough to beat those that knew less.

One reason why the current generation of brute-force programs play so well is Slate's principle[2] Slate's principle states: "A program that understands several goods that are worth achieving will act in such a way as to maximize its own options and restrict those of the opponent". It is not too difficult to see why this is so. Let us define as the "player", the side to move at the root, and as the "opponent", the other side. In a minimax search, it is necessary for the player to have at least one good move at every node at which it is his turn to play. Conversely, he must be able to refute every move at a node where the opponent is to play. Now, assume a program recognizes a small set of goods worth achieving, and preventing the opponent from achieving. It will be able to realize a certain value for the minimax most easily, all other things being equal, if it has the largest number of choices and the opponent has the fewest. The important point is that, when the knowledge applied at the leaf nodes is minimal, the program will only indulge in forcing behavior when some item specified by the knowledge is achievable, no matter how the opponent plays. If no known advantage is achievable, the program will vacillate. Clearly, the more knowledge a program has, the more likely it is to be able to force something it recognizes as worthwhile, and thus indulge in forcing behavior. Even, if the good in one branch turns out to be unattainable upon deeper searching, the fact that forcing moves are being made restricts the opponent's ability to achieve things on his own. This is a fine example of how a mechanism intended to provide one thing can produce additional benefits that humans find necessary to dignify with special principles. It should be noted that when something can be forced, but there is no hurry to force it, an iterative deepening program still does not vacillate. The most direct method of forcing the good is discovered at an early iteration, and the starting move of this sequence will be retained as the candidate to beat for future iterations, so it will win out over all moves that do not accomplish more.

The technique of analyzing the root position and marking squares on which pieces would be well located if they can survive there (e. g. near the opponent's king) has served the Northwestern program well in the past in producing strategy-like behavior. It should be noted that this valuation of the placement of the pieces does not get at one important aspect of chess: the cooperation of pieces. Merely, because a pieces is well placed, this does not mean that it can cooperate with its fellow pieces well from this location. However, I have not found any examples of such lacks in the play of Belle or the Northwestern program. It is possible that such needs are being taken care of by other mechanisms, or that the level of play has not yet reached the point where such concepts are important.

Full-width searches are now beginning to find moves that are obscure to even very good players. The ability to see everything within a given search envelope is more complete than anything that a pattern driven process that projects no deeper can do. At present such searches are being performed in approximately the same time that it takes expert humans to perform at the same level of skill. Thus, within its 8 ply performance envelope, Belle is superior to human performance. However, pattern driven processes do at times capture notions that are deeper than what any brute-force program can achieve. Therefore the issue is: for any given domain, are the deep ideas capturable by patterns, and can a pattern-driven process have access to a sufficient number of such patterns so it can outperform the brute-force searcher. At present, the best human players feel that their understanding of chess will allow them to survive the onslaught of brute-force searching. The choice is between being absolutely accurate within the first 8 ply of search and having a weak understanding of what lies beyond, or making occasional errors in low level searches, but having a much better long-range understanding of the game. In chess, it is well known that it is important to play soundly. "Tactics is 90% of chess" is a common dictum. This means that calculated sequences of moves may not have any errors in them if one wishes to succeed. It is apparent from the games of Belle against good players that they make lots of mistakes in tactical calculation. However, being forewarned about Belle's strong and weak points, it may be possible for very good players to avoid tactical situations in order to attempt to assert their strategic strengths. It is not clear if such a strategy can be consistently pursued against a program that knows as much as Belle does.

It appears that the strategy of deliberately creating an opportunity for Belle to win some very slight amount of material at some long term strategic cost is the best way to try to beat it. In tactical play it appears to be of World Championship caliber, and

[2] Personal communication, not previously published

even when unable to find a useful strategy, it is usually able to defend itself against the intentions of an opponent when these become apparent. Thus the best hope is to get it to bite on something that turns out in the long run to be indigestible. However, it would take a strong and experienced player to succeed at that, as baited traps frequently catch the baiter.

With a 2300 performance capability at present, it would seem that improving the evaluating procedures (that can also be executed in parallel in hardware) should be able to raise the program's ability the 200 or so points needed to play with the best players in the World. Some such contests are already scheduled, and should provide some interesting data on just how far away machines are from wresting the World chess title from humans.

BIBLIOGRAPHY

[1] Berliner, H. J., "Some Necessary Conditions for a Master Chess Program", *Proceedings 3rd International Joint Conference on Artificial Intelligence*, pp. 77-85, August 1973.

[2] Berliner, H. J., "A Chronology of Computer Chess and its Literature", *Artificial Intelligence*, Vol. 10, No. 2, April, 1978, pp.201: 214.

[3] Reinfeld, F., *Win at Chess*, Dover Books, 1958.

[4] Simon, H. A., and Gilmartin, K., "A Simulation of Memory for Chess Positions", *Cognitive Psychology*, Vol. 5, pp. 29-46, 1974.

[5] Slate, D. J., and Atkin, L. R., "CHESS 4.5 -- The Northwestern University Chess Program", in *Chess Skill in Man and Machine*, P. Frey (Ed.), Springer-Verlag, 1977.

[6] Condon, J. H. and Thompson, K., "Belle Chess Hardware", to appear in *Advances in Computer Chess - 3*, Pergammon Press, Ltd., London.

[7] Wilkins, D., "Using Patterns and Plans in Chess", *Artificial Intelligence*, Vol. 14, No. 2, September, 1980, pps. 165-203.

[8] Anonymous, "$100,000 Prize established for First Computer World Chess Champion", *SIGART Newsletter*, #73, October, 1980, p. 15.