THE HEURISTIC SEARCH AND THE GAME OF CHESS
A STUDY OF QUIESCENCE, SACRIFICES, AND PLAN ORIENTED PLAY

Larry R. Harris
Department of Mathematics
Dartmouth College
Hanover, New Hampshire 03755

Abstract

This paper describes the results of applying the formal heurisitic search algorithm to the game of chess, and the impact of this work on the theory of heuristic search. It is not that the application of the heuristic search can by itself solve the problems at the heart of the computer chess, but that representing these problems within the formalism of the heuristic search will further their common solution. A separate search heuristic is proposed that does offer a common solution to the problems of quiescence, sacrifices, and plan oriented play.

Introduction

The purpose of this paper is to describe the results of applying the formal heuristic search algorithm to the game of chess, and discuss the impact of this work on the theory of heuristic search. We hope to demonstrate that a symbiotic relationship exists between the two areas in that the problems encountered when playing computer chess can be better solved within the formalism of the heuristic search and that the theory of heuristic search can be furthered by gaining insight from this very complex application.

Computer chess has b-en dominated by programs using the alpha-beta minimax search (Greenblatt) and more recently by programs using an exhaustive search (Northwestern, Kiassa, Tech II). The trend of successful programs has thus been to a more brute force approach rather than developing more formal solutions to the difficult problems that arise. Approaches using more sophisticated representation and utilization of chess knowledge such as Zobrist and Berliner have been unable to perform competitively with programs using the alpha-beta technique. The actual assessment of new approaches is hampered by the fact that a chess program hangs by its weakest link, and poor play may not be the fault of the new approach. However, approaches that attempt to use more sophisticated heuristics in preference to a general tree search are more susceptible to occasional serious errors in play. It is by no means clear that in chess the tradeoff of search effort for accuracy in evaluation can be consistently made without significant loss of precision. Thus the appeal of the heuristic search, an attempt to combine a sufficiently powerful search mechanism with a knowledgable heuristic.

The heuristic search approach has not proven itself competitively superior to the alpha-beta technique either winning, drawing, and losing games with Columbia, Northwestern, and Tech II respectively. This substantiates the current feeling in the AI community that search strategy itself plays a lower order role than other modules of the system.

However, certain aspects of this approach are worthy of study since they may prove to provide a better mechanism for dealing with the problems at the heart of the computer chess dilemma; namely, quiescence checking, sacrifices, and plan oriented play. We will specify the means by which the heuristic search can express these problems, and propose a common solution, whereas the current minimax paradigm seems roadblocked.

Definitions

The heuristic search makes use of the following functions defined over the set of legal board configurations referred to as nodes,

g(n)	- the number of moves from the root to the node n.
h(n)	= the minimum number of moves from node n to a goal.
f(n)	= h(n)+g(n) The minimum number of moves from the root to a goal via node n.

Since some of these functions are not known during the actual search we must estimate them. The estimating function is denoted by a " ". Thus, the estimating function of h(n) is denoted fi(n) and is typically referred to as the "heuristic" or the board evaluation function. A goal is a winning position.

At each step of the heuristic search the most promising line of play, the one with the best f value, is expanded. In this way the heuristic h orders the search.

The heuristic search for a game tree proceeds as follows.

1) Initialize the search tree to the current board referred to as the root.
2) Select a node for expansion by following the {-pointers to a tip node p. Halt if p is a terminal node.
3) Expand p, linking all sons into the search tree.
4) calculate fi and g for each son, use these to calculate f.
5) Back up the values of the "best" son, setting the f-pointer for each node along the path to p.
6) go to (2).
   The algorithm terminates in step (2) or when time or space constraints are exhausted, in which case the move is made to the 1st level son with the best fi value.

"Best" is defined as either the maximum or minimum value depending on its level in the search tree. In step (5) the information obtained at the tip nodes is backed up into the search tree where it can be compared to other lines of play.

Most theoretical work dealing with the heuristic search deals with the effect of restrictions on the heuristic and their resulting impact on the search. For example, if fi(n)<h(n) for all nodes n then the search is guaranteed to find the minimal cost goal. By applying restrictions that allow for error in the heurstic we can determine theoretically how the heuristic search behaves under conditions of error (Harris '). When dealing with chess we are forced to use *an* error prone heuristic or else no search would be necessary. Thus, it is critical that the search technique remain stable, even when misled by the heuristic. It is shown that the accuracy of the search degrades at worst only linearly with the error in the heuristic.

The Search Heuristic 3

We begin our discussion of the problems of quiescence, sacrifices, and plan oriented play by introducing another forward estimating function called 3. As we will argue later, the notion of a separate search heuristic, one that helps guide the search independent of the fi evaluator, has application in problem domains other than chess. We will argue the need to order the search on the basis of information other than that represented by g or fi. The order of expansions will now be determined by f=g+h+3.

For example, in tactical positions the fi estimator is of little use in searching for the best line of play since the notions fi typically measures, such as material advantage and pawn structure, become temporarily unimportant relative to an effective tactical threat. The 3 estimator tries to forecast the tactical line and bias the search in this direction. Note that we are not saying that 3 can accurately estimate the results of the tactics -- if this could be done we would simply put the result directly in fi. We assume only that d will be oversensitive and signal when tactics <u>might</u> prevail. In these cases we leave it to the search to investigate and verify the 3 prediction. We hope to simply bias the search in the proper direction.

The need to separate fi & 3 is clear from our use of h to select actual moves to be made on the board. In this way 3 guides the search, but does not directly affece move decisions, unless the d prediction is manifested by an improved h after a successful search of the tactical line. Thus we associate moves with high fi as being "good moves to make", and moves with high 3 as being "good moves to investigate further."

<u>Quiescence Checking</u>

In order to accurately predict tactical lines 3 must be sensitive to much more than overt captures and checks, it must consider pins, forks, discovered attacks, back rank attacks, pawn promotion, and other phenomena likely to bring about an abrupt change in piece advantage. In this sense the 3 measure is a quiescence check <u>par excellence</u>. When 3 is near zero then the h esciiuace can be considered accurate since there are no pending tactical threats. In this way the "horizon effect" (Berliner) can be avoided by stopping the search when all tip nodes of the search tree are quiet, in which case the search tree is said to have terminated.

Thus we are using the same estimator that finds tactical lines to define quiescence in a much more sophisticated manner than others (Greenblatt) have suggested. Typically only overt captures and checks are included in a quiescence check, 3 includes these even to the degree of signaling high values 1 ply after the check or capture to force continued evaluation of a forced line. But 3 also includes the tactical ploys that often precede sudden fluctuations in fi.

We must consider the problem that with this loose definition of quiescence, the tree may <u>never</u> terminate. In this regard we note that 3 features are not additive; that is, they do not cancel out. If both players have roughly equally promising threats, the 3 function musu not sum to zero, as this board is anything but a quiet situation that can be accurately rated. By computing separate d values for each of the two players we can avoid this problem and can also define tree termination in a way that is more likely to occur. By separating each player's threats we can dynamically rate a board as stable when the son with the best fi value has one sided threats that would only improve the rating. Using the same

THE HEURISTIC SEARCH AND THE GAME OF CHESS
A STUDY OF QUIESCENCE, SACRIFICES, AND PLAN ORIENTED PLAY

Larry R. Harris
Department of Mathematics
Dartmouth College
Hanover, New Hampshire 03755

Abstract

This paper describes the results of applying the formal heurisitic search algorithm to the game of chess, and the impact of this work on the theory of heuristic search. It is not that the application of the heuristic search can by itself solve the problems at the heart of the computer chess, but that representing these problems within the formalism of the heuristic search will further their common solution. A separate search heuristic is proposed that does offer a common solution to the problems of quiescence, sacrifices, and plan oriented play.

Introduction

The purpose of this paper is to describe the results of applying the formal heuristic search algorithm to the game of chess, and discuss the impact of this work on the theory of heuristic search. We hope to demonstrate that a symbiotic relationship exists between the two areas in that the problems encountered when playing computer chess can be better solved within the formalism of the heuristic search and that the theory of heuristic search can be furthered by gaining insight from this very complex application.

Computer chess has b'en dominated by programs using the alpha-beta minimax search (Greenblatt) and more recently by programs using an exhaustive search (Northwestern, Kiassa, Tech II). The trend of successful programs has thus been to a more brute force approach rather than developing more formal solutions to the difficult problems that arise. Approaches using more sophisticated representation and utilization of chess knowledge such as Zobrist and Berliner have been unable to perform competitively with programs using the alpha-beta technique. The actual assessment of new approaches is hampered by the fact that a chess program hangs by its weakest link, and poor play may not be the fault of the new approach. However, approaches that attempt to use more sophisticated heuristics in preference to a general tree search are more susceptible to occasional serious errors in play. It is by no means clear that in chess the tradeoff of search effort for accuracy in evaluation can be consiSwently made without significant loss of precision. Thus the appeal of the heuristic search, an attempt to combine a sufficiently powerful search mechanism with a knowledgable heuristic.

The heuristic search approach has not proven itself competitively superior to the alpha-beta technique either winning, drawing, and losing games with Columbia, Northwestern, and Tech II respectively. This substantiates the current feeling in the AI community that search strategy itself plays a lower order role than other modules of the system.

However, certain aspects of this approach are worthy of study since they may prove to provide a better mechanism for dealing with the problems at the heart of the computer chess dilemma; namely, quiescence checking, sacrifices, and plan oriented play. We will specify the means by which the heuristic search can express these problems, and propose a common solution, whereas the current minimax paradigm seems roadblocked.

Definitions

The heuristic search makes use of the following functions defined over the set of legal board configurations referred to as nodes.

g(n)	= the number of moves from the root to the node n.
h(n)	- the minimum number of moves from node n to a goal.
f(n)	= h(n)+g(n) The minimum number of moves from the root to a goal via node n.

Since some of these functions are not known during the actual search we must estimate them. The estimating function is denoted by a " ". Thus, the estimating function of h(n) is denoted fi(n) and is typically referred to as the "heuristic" or the board evaluation function. A goal is a winning position.

At each step of the heuristic search the most promising line of play, the one with the best f value, is expanded. In this way the heuristic fi orders the search.

The heuristic search for a game tree proceeds as follows.

1) Initialize the search tree to the current board referred to as the root.
2) Select a node for expansion by following the f-pointers to a tip node p. Halt if p is a terminal node.
3) Expand p, linking all sons into the search tree.
4) calculate fi and g for each son, use these to calculate f.
5) Back up the values of the "best" son, setting the ?-pointer for each node along the path to p.
6) go to (2).
   The algorithm terminates in step (2) or when time or space constraints are exhausted, in which case the move is made to the 1st level son with the best fi value.

"Best" is defined as either the maximum or minimum value depending on its level in the search tree. In step (5) the information obtained at the tip nodes is backed up into the search tree where it can be compared to other lines of play.

Most theoretical work dealing with the heuristic search deals with the effect of restrictions on the heuristic and their resulting impact on the search. For example, if fi(n)<h(n) for all nodes n then the search is guaranteed to find the minimal cost goal. By applying restrictions that allow for error in the heurstic we can determine theoretically how the heuristic search behaves under conditions of error (Harris '). When dealing with chess we are forced to use an error prone heuristic or else no search would be necessary. Thus, it is critical that the search technique remain stable, even when misled by the heuristic. It is shown that the accuracy of the search degrades at worst only linearly with the error in the heuristic.

The Search Heuristic 3

We begin our discussion of the problems of quiescence, sacrifices, and plan oriented play by introducing another forward estimating function called 3. As we will argue later, the notion of a separate search heuristic, one that helps guide the search independent of the fi evaluator, has application in problem domains other than chess. We will argue the need to order the search on the basis of information other than that represented by g or fi. The order of expansions will now be determined by f=g+fi+d.

For example, in tactical positions the fi estimator is of little use in searching for the best line of play since the notions fi typically measures, such as material advantage and pawn structure, become temporarily unimportant relative to an effective tactical threat. The 3 estimator tries to forecast the tactical line and bias the search in this direction. Note that we are not saying that 3 can accurately estimate the results of the tactics -- if this could be done we would simply put the result directly in fi. We assume only that 3 will be oversensitive and signal when tactics <u>might</u> prevail. In these cases we leave it to the search to investigate and verify the 3 prediction. We hope to simply bias the search in the proper direction.

The need to separate fi & d is clear from our use of fi to select actual moves to be made on the board. In this way 3 guides the search, but does not directly affece move decisions, unless the d prediction is manifested by an improved fi after a successful search of the tactical line. Thus we associate moves with high fi as being ''good moves to make", and moves with high 3 as being "good moves to investigate further."

Quiescence Checking

In order to accurately predict tactical lines 3 must be sensitive to much more than overt captures and checks, it must consider pins, forks, discovered attacks, back rank attacks, pawn promotion, and other phenomena likely to bring about an abrupt change in piece advantage. In this sense the 3 measure is a quiescence check <u>par excellence</u>. When d is near zero then the fi estimate can be considered accurate since there are no pending tactical threats. In this way the "horizon effect" (Berliner) can be avoided by stopping the search when all tip nodes of the search tree are quiet, in which case the search tree is said to have terminated.

Thus we are using the same estimator that finds tactical lines to define quiescence in a much more sophisticated manner than others (Greenblatt) have suggested. Typically only overt captures and checks are included in a quiescence check, 3 includes these even to the degree of signaling high values 1 ply after the check or capture to force continued evaluation of a forced line. But 3 also includes the tactical ploys that often precede sudden fluctuations in fi.

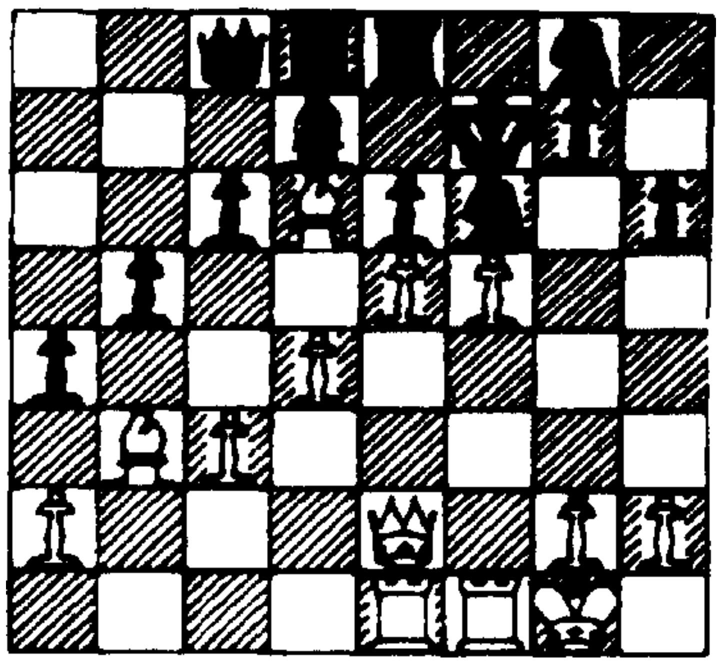
We must consider the problem that with this loose definition of quiescence, the tree may <u>never</u> terminate. In this regard we note that 3 features are not additive; that is, they do not cancel out. If both players have roughly equally promising threats, the 3 function must not sum to zero, as this board is anything but a quiet situation that can be accurately rated. By computing separate 3 values for each of the two players we can avoid this problem and *can also* define tree termination in a way that is more likely to occur. By separating each player's threats we can dynamically rate a board as stable when the son with the best fi value has one sided threats that would only improve the rating. Using the same philosophy as alpha-beta cutoffs we can backup the quiet rating when-ever one-sided attacks won't change the ordering even if they were to prove successful. In this way the root can eventually be rated as quiet even though many tip nodes of the search tree are not. This definition also makes sense from a search point of view since we would not wish to waste time searching a line of play whose outcome would not change the order of our ratings, while there remain nodes is the search tree whose ratings are unstable.

Quiescence checking in the heuristic search environment is on a scale relative to the entire search tree. The nodes with high 3 will be expanded first until they become quiet relative to other sections of the tree. In fact, one could view the search process as attempting to monotonically reduce the backup 3 value of the root until it becomes close to zero; meaning that the dynamic rating of the root is accurate. If the value does not reach zero before it is time to move then the situation is too complex for accurate play, but we can at least make a reasonable move based on the information gained in reducing the 3 value of the root as much as possible.

In the alpha-beta minimax environment the queiscence check must be made independent of other factors in the tree, making relative comparison impossible. The decision must be made at the point of maximum ply whether or not to continue the expansion. Since the algorithm is recursive there is no information about other areas in the tree. As such, the decision must be made on absolute grounds, requiring a conservative approach to make the algorithm terminate within the time constraint in all cases. Thus, most quiescence checks in the alph-beta environment do not include all the factors necessary to determine whether a board is really quiet.

Critical Sacrifices

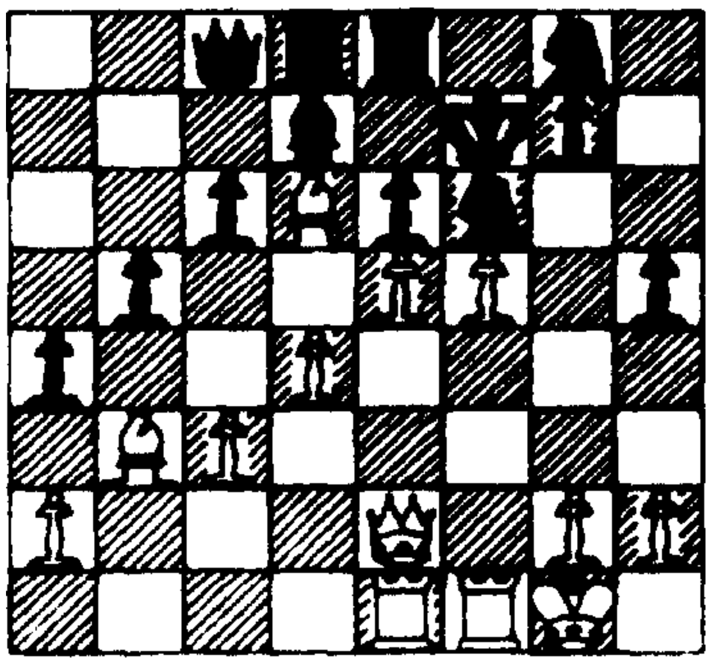
Another flaw of computer chess play is the inability to find or adequately defend from effective sacrifices. We will show how the search heuristic 3 can be used in solving this problem also. The following two boards are extreme examples of sacrificial play, but as such clearly demonstrate the problems computers face when required to search these lines of play. Board I is a famous 19-ply mating sequence described in Berliner-., that begins with a queen sacrifice. The line is 1. Q-R5ch, NxQ 2. PxPch, K-N3, 3. B-B2ch, K-N4, 4. R~B5ch, K-N3 5. R-B6ch, K-N4 6. R-N6ch, K-R5 7. R-K4ch, N-B5, 8. RxNch, K-R4, 9. P-N3, any 10. R-R4mate.


Board I

We are faced with the problem of finding such moves when they are successful, but also avoiding such moves when they are not. This latter concern is even more important since a program could play good chess without the ability to make 19-ply mates, but not if it continually enters such sacrifices when they are not successful.

The sacrificial move might be defined as a good move to investigate, but one to avoid making until its full effects are searched out. In the proposed scheme this would be a move with an incremental loss in fi representing the sacrificed piece, and *an* artificially high 3 to temporarily overcome the loss in fi. This high d will bias the search to investigate the line in spite of the material loss.

Any search, minimax or heuristic, that uses a single value to compare boards and make moves, cannot separate the good from the bad sacrifices. The reasoning is that a 5-ply minimax search could not find the mate in a single search simply because of the depth of the sequence. But, it is entirely possible to play the mating sequence using a series of 5-ply searches. The problem arises in the first search, where we are forced to decide whether the queen sacrifice is successful or not, without being allowed to fully explore the sequence. If the heuristic rates the sacrifice as worthwhile then the move will be made without a thorough investigation. This of course, is very dangerous as Board II exemplifies. Board II is identical to Board I except that black's KRP is at R4 instead of R3. The queen sacrifice is not now successful because of this small change, since the king can eventually hide and avoid further checks. The line is: 1. QxPch, NxQ 2. PxPch, K-N3 3. any, K-R3 ending the checks. Surely it must be a sensitive heuristic to be <u>sure</u> of the outcome of a queen sacrifice in situations such as these. Since the stakes are so high, most programs will play it safe and will avoid the sacrifice. This is a result of the minimax forcing a premature decision without allowing a thorough investigation of the sequence.



Board II

The only way to avoid this two-sided dilemma is to separate the search ordering from the move ordering. Once we have allowed this distinction we could explore the queen sacrifice more deeply. If the sacrifice is successful this will be manifested by an improved h value. Until the fi value of the sacrifice is rated as the best in the search tree we will not be tempted to make the move, but this does not restrain the search from investigating it further. Once again, the notion of a separate search heuristic, to guide the search to areas of the move tree that need further investigation before they can be accurately assessed, makes for a clean solution to this very difficult problem. For as long as move decisions and search criteria are based on a single value, sacrificial lines of play like those exemplified above cannot be accurately solved.

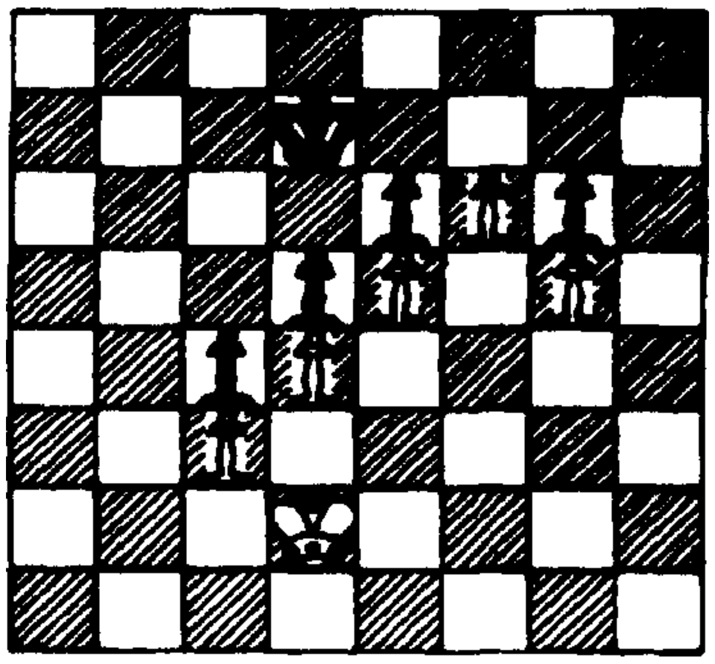
Clearly some tuning of the d heuristic is required to avoid investigating every seemingly interesting possible sacrifice. This is done by requiring substantial mobilization of attacking pieces combined with limited mobility of the attacked piece before the d rating will suggest the further consideration of a sacrifice. Both of these conditions exist in the above situation and would allow for the sacrificial line to be searched even beyond the 8th move which is not a check for white.

<u>Plan Oriented Play</u>

Computer chess programs are most often criticized because of their lack of using a plan to guide their play. The program may know to develop its pieces and castle early, thus playing the opening fairly well, only to flounder early in the midgame waiting for something to happen. Against human opponents the game changes rapidly enough so that the program must react to threats and can formulate short range attacks so the lack of a plan is not always apparent. When two computers play each other the probability of stagnation increases markedly.

Nowhere, however, is the lack of a plan more critical than in endgame play. In these cases a winning pawn advantage can be lost simply because a long range plan is needed to promote it. Let us consider some examples of such endgames and how the heuristic search environment provides a means of using long range goals. Our concern here is not the very difficult problem of suggesting what plans are worthy of consideration in a given situation, but the difficulties that occur when we try to accurately measure the effectiveness of a given plan.

A board suggested by Berliner is shown in Board III. In this case the white king must simply flank the black pawn wall to support the advance of the white pawn at B6. In this case the plan itself is quite clear, but the manner in which it musu be carried out is hard for a computer to find. The problem appears difficult to a computer program because the white king must first move <u>away</u> from the desired position supporting the advance of the pawn at B6. We can expect that most chess heuristics will rate the moves K-K3 or P-B7 initially better than the required K-B2. Given this error in the heuristic how would the minimax and heuristic searches react?



Board III

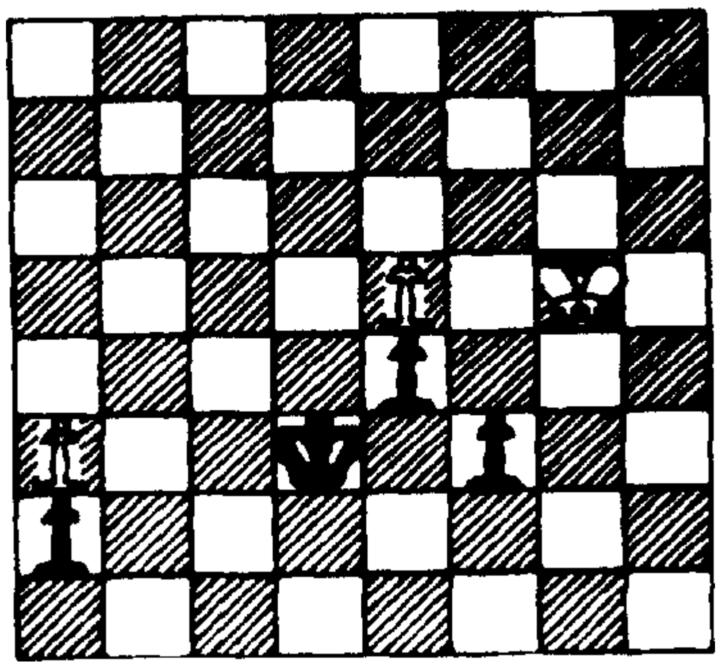
Berliner gives a detailed scenario for the minimax. The move P-B7 will soor lose the pawn so the rating drops accordingly. But the rating for the move K-K3 increases as the king progresses directly towards its goal. The rating for K-B2 will be relatively low since the distance from the king and its goal is widened. At the bottom of the 5~ply search tree the minimax will compare the relative closeness of the king to the passed pawn obtained by K-B2. The choice is quite clear and the minimax backs up a descendant of K-K3 not realizing the futility of the move.

How then can the heuristic search, operating with the *same* error prone heuristic, find the winning move? It should not be surprising that the heuristic search can solve the problem since it has long been used to solve similar -puzzles such as the 9-puzzle (Nilsson). It is just for situations such as this in which the heuristic search has been most successful. The search scenario proceeds as follows.

The moves P-B7 and K-K3 will undoubtedly be expanded first because of their high ratings relative to K-B2. In fact the move K-B2 would remain unexpanded for quite some time until all other moves demonstrate their lack of merit. Soon P-B7 will lose the pawn and be rated below K-B2. However, the descendants of K-K3 will at first improve as the king progresses. When the king reaches KB4 the king will be forced to retreat. retreat. This retreat plus the buildup of g values for these lines will soon diminish their ratings until the point that K-B2 is expanded.

In the previous example there was one obvious plan and were it not for "getting over the hump" of the pawn wall, the solution would be easily found by any search technique. A similar problem arises when there are multiple plans that could apply in a situation and the best plan requires more moves to become effective.

Consider the situation in Board IV. Here, Black could formulate 2 plans to exploit his pawn advantage. He could aggressively attack the weak white pawn at R2 by moving K-B3. Black could also try to create a passed pawn in two ways by moving either K-Q3 or K-B4. The first two moves lead to a draw, but thit is not evident for 6 ply. The 3rd plan wins, but is not clearly evident for about 15 ply. Thus a program that searches 5 ply on all 3 lines will be unable to distinguish the good plan and is likely to draw the position.



Board IV

When searching with multiple plans in mind is is not enough to rate by simply taking the best value obtained by any plan. This would tend to bias the search towards the plan that has the most immediate payoff. This, of course, can prove to be short sighted, often ignoring the better plan For example, if we are required to rate Board IV using only one value, then we are forced to determine statically which plan is best. When this determination is in error, as will often be the case, the search would be biased away from investigating the best plan. It is actually necessary to keep separate ratings of each plan and either perform the search sequentially for each plan or perform it in parallel using the multiple heuristic values. The former *case* would seem prohibitive with regard to time. The latter case would require backing up multiple information for each plan. This would severely hinder the ability to make alpha-beta cutoffs since the cutoff would have to occur for all plans. On the other hand, the heuristic search could simply continue the expansion of both plans taking into account the g factors so that small "humps" to be crossed, and a bias in d to investigate all plans to the deptl.

necessary to accurately assess their potential.

Thus, as plan oriented play is introduced into computer chess, it is clear that part of its implementation must be in biasing the search deep enough to accurately measure the success of the plan. The search heuristic d is one means by which this can be carried out.

Impact upon the General Heuristic Search

We have *seer the* utility of the search heuristic a* in chess. The use of this heuristic came about because the most effective order of expansion differed from the order determined by the g+fi measure. That is, some nodes were expanded, not because of high g+fi ratings, but because their fi rating could not be accurately measured. In these cases the search itself must be used to help accurately rate the node.

Considerations such as this already appear in the literature of the heuristic search, although in a different terminology. In applying the heuristic search to solve the Traveling Salesman Problem, Pol uses fi with two factors that are really search heuristics. Both of these factors tend to more effectively order the sequence of expansions. Since they are based on search criteria (depth in the tree) and not based on the node itself, we could label them as part of the search heuristic d.

This relabeJing allows us to distinguish the heuristic factors relating to the node itself, fi, and those relating to the search process 3. Once this is done we can standardize the definition of f to be f = g + fi + 3 and recognize that this "biasing of ?" is an important aspect of the algorithm itself and will be necessary in many problem domains where high performance is required.

One spinoff of work on a high performance chess program has been this recognition of the need for a separate search heuristic. There are other insights to the general technique of heuristic searches that can be gained from high performance chess. We use the formalism of the heuristic search to abstract the ideas of chess programmers and, if possible, apply them in other search domains.

An example of this is Berliner's "Refutation Description", in which he reduces the search space by passing information up the tree. It is applied in chess when a certain line of play thought to be good, ends in a catastrophe. At this point details of the catastrophe, such as the attacker, attack path, attackee, and flight path are passed back up the tree. From this point only moves that avoid the catastrophe are considered.

If we describe this technique in the terminology of the heuristic search, we can try to apply it to other problem domains. Whenever the heuristic rises suddenly (gets worse) after the expansion of a node, then we have detected some sort of catastrophe; since the path was considered to be the best we had, or we wouldn't have been expanding it. If we can now isolate the reason for this sudden rise in fi, this information could be used to reduce the search space.

Hopefully, continued work on high performance problem solving systems such as chess will continue to contribute searching techniques that can be applied effectively across problem domains.

References

1. Greenblatt, R.D et al, "The Greenblatt Chess Playing Program", Prodeedings of the 1967 Fall Joint Computer Conference, pp. 801-R10
2. Zobrist, A.L., Carlson, F.R., "An Advice Taking Chess Computer", Scientific American, June 1973, v. 228, No. 6, p. 92
3. Berliner, H., "Chess as Problem Solving: The Devlopment of a Tactics Analyzer" Ph.D. Thesis, Carnegie-Mellon University
4. Harris, L.R.,"A Model for Adaptive Problem Solving Applied to Natural Language Acquisition", Technical Report TR-133, Cornell University
5. Harris, L.R. "The Heuristic Search Under Conditions of Error", Artificial Intelligence, Fall 1974, vol 5, no. 3, p. 217
6. Berliner, H.,"Some Necessary Conditions for a Master Chess Program", Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973
7. Nilsson, N. ,<u>Problem Solving Methods</u> in <u>Artificial Intelligence</u>/ New York:McGraw Hill
8. Pohl, I., "Avoidance of (Relative) Catastrophe etc.", Proceedings of the Third International Joint Conference on Artificial Intelligence, 1973