Artificial Intelligence 120 (2000) 235–250

using move influence Kieran Greer 1

e Engineering, Faculty o ewtownabbey, N. Irelan

Received 12 June 1999; received in revised form 1 February 2000

T

move. Depending on what squares a player controls, the chessmaps heuristic tries to determine w the important areas of the chessboard are. Moves that influence these important areas are then or

first. The heuristic has been incorporated into a move-ordering algorithm that also takes ac of immediate tactical threats. Human players also rely strongly on patterns when selecting m but would also consider immediate tactical threats, so this move-ordering algorithm is an at to mimic something of the human thought process when selecting a move. This paper prese new definition for the influence of a move, which improves the performance of the heuristic. I presents a new experience-based approach to determining what areas of the chessboard are impo which may actually be preferred to the chessmaps heuristic. The results from game-tree sea suggest that the move-ordering algorithm could compete with the current best alternative of

the history heuristic with capture moves in a brute-force search.  2000 Elsevier Science B.V rights reserved.

Keywords: Chessmaps heuristic; Computer chess; Search heuristic; Pattern-oriented; Neural network

This pape

heuristic called the chessmaps heuristic, that orders moves depending on which areas o chessboard they influence. The definition for the influence of a move is given in Section

1 Email: krc.greer@ulst.ac.uk.

0004-3702/00/$ – see fron

|  |  |
| --- | --- |
| Fig. 1. Layout of a chessboard divided | Fig. 2. An alternative layout of a |

from 1 to 64 as shown.

divided into different numbers of sectors. For Fig. 1 the sector size is only 1 square l and tests showed that this was the most effective sector size. The chessmaps heuristic the output of a neural network to order these sectors into relative importance. Moves influence important sectors are then thought to be stronger than moves that influence important sectors only and so would be looked at first. Details of preliminary tests usin move-ordering algorithm for game-tree searches are reported in [6]. This paper will pre an improvement made to the chessmaps heuristic and provide further support to the val of the move-ordering algorithm through the results of more game-tree searches.

Two methods will be used to try and determine what sectors are the most relevant position. The first method is the knowledge-based chessmaps heuristic, which uses a ne network to try and learn a relation between the control of the squares and the influenc a move. The definition for the control of a square is given in Section 2.1. For this me to be sensible, the control of the squares must contain important information for any c position. The chess concept of space is represented by how much of the chessboard control. It is known that a player with a space advantage often attacks, while a player w space disadvantage often defends. So the control of the squares can be used to define a

basic strategy, which the neural network managed to learn to some degree (see Sectio Also, as the control of the squares is calculated by determining what squares the pi attack or defend, it is directly related to the movements of the pieces and the relations between the pieces. There is thus important information contained in the control o squares, but this needs to be represented to the computer program in an appropriate

and one attempt is presented in this paper. The second method is experience-based, w the sectors are ordered according to the results of the previous search of the game-This is really an alternative to the chessmaps heuristic. A novel aspect of both of t sector-ordering methods is that they order moves depending on their influence; that is piece does not need to actually move to the sector in question itself. Other move orde heuristics have been concerned with the squares that the pieces actually move to.

Pattern recognition plays an important role in a chess player’s thought process. C masters store about 50000 chunks of chess information, represented by patterns. They

retrieve this information and apply it to any new chess position. The information

between the different pieces. Work done on the psychology of chess players inclu their thought processes when selecting moves can be found in [4] and [12]. Much of goes into selecting a move occurs during the first few seconds of looking at a posi when a player scans the board trying to recognise relevant chunks of information and an overall impression. The control of the squares is a crude attempt at giving a ge first impression of the position and would contain much less information than the pa chunks that a human player stores but is a step in this direction. There have been va attempts at pattern-based approaches to learn to play chess. One recent attempt is [9] there are also other works (e.g., [3,8,13]). A recent study of machine learning atte applied to computer chess can be found in [5]. Neural networks are particularly well s to the problem of pattern recognition. Because the information being fed into the n network for this method is very general, it is possible to represent all phases of a c game, which is necessary to generate a general search heuristic. Other methods conce with extracting rules that determine if specific moves can be played would not be prac here, because of the number of rules or cases that would need to be stored.

The chess position and move must be pre

the squares is the same as has been used previously, but a new definition for the influ of a move will be presented.

2.1. Control of a square

Some general rul

and 0 if the square is neutral. A square can either be occupied or empty and we will fi consider the case where the square is occupied. If a white piece occupies the square a is not attacked or defended, then the control of the square is defined as neutral. If the p is defended and not attacked then White controls the square. The complications arise w the piece is attacked by Black, where a capture sequence on the square needs to be ma

determine who controls the square. If Black can capture on the square, but loses mater he does so then the square is under White’s control. If however he gains material then square is under Black’s control. If Black neither gains nor loses material by capturing

the control of the square is neutral. Equivalent rules apply for a black piece occupyi square. If the square is empty, then White controls it if he can move a more valuable p to the square without loss of material and Black controls the square if he can move a m valuable piece there. The loss of material is again determined by capture sequences on square. The control of the square is said to be neutral if both sides can move a piec equal value to the square without loss of material, or neither side can move a piece to square.

To calculate the control of a square it is firstly necessary to

with another piece on the square and so could move there only after the first piece has so. Also taken into consideration is if the piece is pinned to the king, where a piece ca enter into a capture sequence or move if this exposes its king to check. So only

captures and moves are considered. When determining who controls the square, ca sequences are made on ascending material value of the capturing piece, so pawn cap are considered first and king captures last. The capture sequences can also be termin early (not all captures made) if this leads to a favourable evaluation for the captu side. For example, making all possible captures on a square for one player may lea an overall material loss, but making just the first capture could mean a material gain

following example in Fig. 3 may help to explain the process used to determine the co of a square.

In this position it is Black to move and we are wondering if he can move his knig g5, so we are just considering the capture sequences on this square. Note that the w knight on e4 is pinned to the white king and the white bishop on d2 can capture o

square only after the white queen has done so. If we were to consider all capture mov the square then the capture sequence would be:

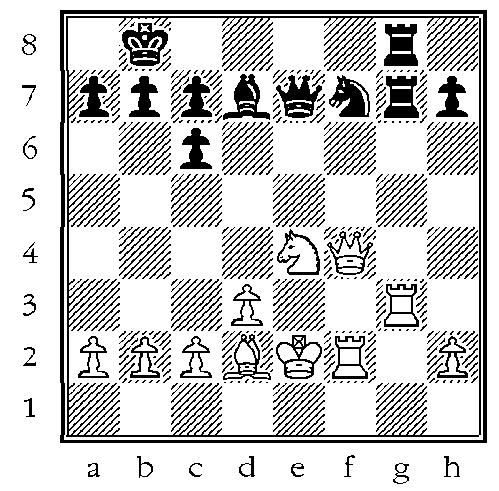
1. BNf7−g5. WRg3× g5.

3. BRg8× g5. WBd2× 4. BQe7× g5. WNe4×

And White has ended up a ro

a knight and be ahead in material. As White cannot safely move a piece to the sq the control of the square belongs to Black. To ensure that the correct capture seque are performed, not only stored is the material value of the pieces that attack a sq but also the conditions under which they can move to the square. Some pieces can m directly to a square and some can move to a square only after another piece has

so. For the white bishop in Fig. 3, the condition under which it could move to g5 w be AfterWhiteQueen and for the white knight it would be AfterBlackQueen, but Bla



knight could move directly to the square and would be given the condition Direct. It is

determining the control of a square we consider each square separately and do not w that capturing on one square could mean the loss of material on another square.

2.2. Chessmap

A chess

of each side. Appendix A illustrates the chessmap and the influence of a move gene for the position in Fig. A.1. If we consider Fig. A.1, then in this position the strategie both sides are well defined. White has the greater space on the queenside and attacks while Black plays for a kingside attack. The chessmap for this position, given in Fig. seems to strongly agree with these strategies and the following simple algorithm cou used to generate a chessmap for a position:

(1) For each square on the chessboard record which pieces attack the square an conditions under which they can move to the square.

(2) For each square on the chessboard then perform the following:

2(a) If the square is occupied then, if the piece is attacked, calculate the ca

sequence on the square to determine who controls it. If the piece is defe and not attacked then the control of the square belongs to the side that o the piece. If the piece is not attacked or defended the square is neutral.

2(b) If the square is not occupied then make each legal move to the square in and record the highest valued piece for each side that can move to the sq

without loss of material on the square. The side that can move the higher va piece to the square then controls the square. The control of the square is ne if the piece values are equal or neither side can move a piece to the squar

2.3. Influence of a move

A new definition

number of sectors. The influence of a move is meant to represent a player’s inten when he moved a piece to a certain square. Unfortunately, when automatically gener the move influence, a lot of sectors are included in the influence that are not really rele to the move at all. So this process is a bit fuzzy, but as will be seen the definition used seems to be slightly more accurate than the previously used definition, which can be f in [6]. A computer program has been written to automatically process the chess posi and moves. Every time a move is made, an array in the computer program is upd This array is called valueboard and stores for each sector the sum values of all pieces directly or indirectly attack that sector. The value of any piece that sits in the sector is added to the sum value for that sector. It is possible to take the valueboard values b and after the move and use this as the influence of the move in some way. 2 Con

2 I would like to mention Dr. Piyush Ojha who supervised my D.Phil. research on this work. He suggest

the time.

the position in Fig. A.1, where the valueboard for this positio

in Fig. A.4. Fig. A.5 then shows the differences in the two valueboards. Table A.1 g the values for each piece that are used in the valueboard array, or stored as the ma

value of the piece for the capture sequences and these values may look a bit strang is necessary to give the rooks and bishops of each side different values. This is bec when we move one rook this may set a certain condition to true in a capture sequence we need to be able to determine that one rook move is not confused with the other

moving. The same applies for the bishops. Note that the king is given a value of ±4 the valueboard array when being used as an attacking piece but ±200 when included

capture sequence.

The influence of the move can then be defined as follows: If a sector (or square fo

sectors) in the valueboard array is changed in favour of the side to move (positivel White or negatively for Black), then the influence of that sector is set to 1. If the val unchanged, then the influence is set to 0 and if it is changed in favour of the opposing then the influence is set to −1. Fig. A.6 illustrates what the influence of the move R position of Fig. A.1 would be and the following algorithm could be used to determin move influence:

(1) Store the valueboard array values before the move.

(2) Make the move on the board and then re-calculate the valueboard array values.

(3) Subtract the old valueboard values from the new valueboard values.

(4) The resulting values for each sector will then determine the move influenc

follows:

4(a) If it is a White move, then if the difference in the valueboards for a sect

positive then the move influence for that sector is +1. If it is negative the move influence is −1 and if the difference is 0 then the move influen also 0.

4(b) If it is a Black move, then if the difference in the valueboards for a sect negative then the move influence for that sector is +1. If it is positive

the move influence is −1 and if the difference is 0 then the move influen also 0.

This new definition only stores the new sectors influenced by the move and no sectors influenced. As we generally move a piece to attack a new square this may he

remove some of the inaccuracy of the older definition that included all sectors influe by the move. Including all sectors may have made the definition more fuzzy, as a gr number of sectors that are not really associated with the move may also have acciden been included in the move influence. Note that with 64 sectors, the sector that the p moves to would not be considered as being influenced with the new definition.

definition also gives a negative value to sectors that are weakened by the move, w helped when training a neural network, but does not affect any move ordering. As be explained in Section 3, when ordering moves only sectors with a positive valu considered.

3. Training and testing a neural network

A feedforward neural network was

position test set. These data sets were generated from complete and randomly chosen c games taken from master and grandmaster play. For each position in the training or tes the input for the neural network was a 70-element vector and the desired output was

element vector. The input vector consisted of the values 1, 0 or −1. The first 64 elem represented the chessmap values of the position and the final 6 elements represente relative positions of the kings. The king positions were included in the hope that they w help the neural network to determine when to suggest attacking or defensive strategie possibly in the endgame. Earlier tests showed that the inclusion of these pre-comp features improved performance only very slightly, but that they had an influence on the sectors were ordered. Each king was defined as being either on the queenside, i centre or on the kingside. This required 3 input nodes, one for each area, where a v of 1 meant that the king was in that area of the board and a value of −1 meant th was not. Thus to represent the positions of both kings, a total of 6 extra input nodes

required. The desired output was a vector quantifying the influence of the move play that position. Each output node represented a sector and each element of the desired o vector was either 1, 0 or−1 as defined by the move influence definition of Section 2.3 input and desired output values for the position of Fig. A.1 in Appendix A can be f in Figs. A.7 and A.6 respectively and these are the sort of values that the neural net would be trained on.

The classification task to be learned by the neural network was then as follows: G the chessmap representation of a chess position and the king positions, the neural net

would learn to recognise what areas of the chessboard were important to the position what areas were not. It did this by attempting to learn the move influence pattern fo position in question, so that when the same position occurred again it would sugges same move influence. As already stated, this move influence pattern is a bit fuzzy. How if a relation does in fact exist between the square control and the influence of a move, similar positions should have move influence patterns that include a similar core nu of important sectors. Each move influence pattern may then also contain a number of o sectors, but because these will not occur in as many patterns, they will not be recogn by the neural network as being as important. The weight values associated with the

sectors for a particular type of position will then be reinforced the most and these se will obtain the largest output values when a position of this type is encountered again positions were considered from the White side, so when it was Black to move the pos was reversed. Testing suggested that a 3-layer architecture was to be preferred to layer architecture, with 16 hidden nodes being a good number. Thus the architecture o neural network was 70-16-64. After training, the continuous valued actual output valu the neural network were used as the sector ordering, where sectors corresponding to n with larger output values were ordered first. In this way the neural network was being

3 The neural network simulator used was an older version of Don Tveter’s backpropagation package

more like a statistical classifier, to produce something similar

being used to order the sectors, it was important to consider not just the error value, but to obtain a good spread of values over all of the output nodes. Without this a good ord could not be obtained. Thus weight values could be chosen before the neural network reached its minimum error if this produced a good spread of the output values.

After the neural network was trained on the training set, its performance was meas as follows: For each position in the test set, the sectors were ranked by the neural netw The highest ranked sector that was influenced by the player’s move in that position then calculated. Percentage values were then calculated for the entire test set that indic how accurate the sector ordering would be if only a certain number of sectors we be used. For example, 20% of the time the move played may have influenced the hig ranked sector, but 30% of the time it may have influenced one of the two highest ra sectors, and so on. Note that when calculating these percentage values, or when ord

the moves, only sectors positively influenced by the move are considered. The percen values generated for the test set for just the top 15 sectors are given in Table 1 in ‘% Accuracy for a neural network’ row. The values in this row can be interprete follows: 25.3% of the time the move played would have influenced the highest ra sector. If the move played did not influence the highest ranked sector, then 13% o time it would have influenced the second highest ranked sector, and so on. So summin percentage values for a particular number of sectors will give some idea of how acc the sector ordering is likely to be for that number of sectors. These values are slig down on those published previously, but the new definition for the influence of a m includes fewer sectors, making it more accurate. The second row of this table give

average number of moves that are grouped together for each of the 15 highest ra sectors. So when looking at the highest ranked sector there were an average of 4.6 m that influenced it. There were then an average of 3.2 moves that did not influence this s but influenced the second highest ranked sector, and so on.

These values can be compared with the likely values produced by a random m ordering. There were an average of about 32 moves per position for the entire tes

so if we consider a random move ordering, then the move played is equally likely

ordered from position 1 to 32. This means that there is a 50% chance that it will be ord in the top half of all moves, or in the top 16 moves. Looking at the values of Table

can see that the top 4 sectors have an accuracy of 54.8% and the number of moves lo at for these sectors is 12.6. Or alternatively, the top 6 sectors would look at a total ave of 16 moves and the percentage accuracy for this is 64.3%. Although these values ar Table 1

Values indicating the accuracy of the sector ordering for the test set generated by a neural network and the av number of moves looked at for each sector

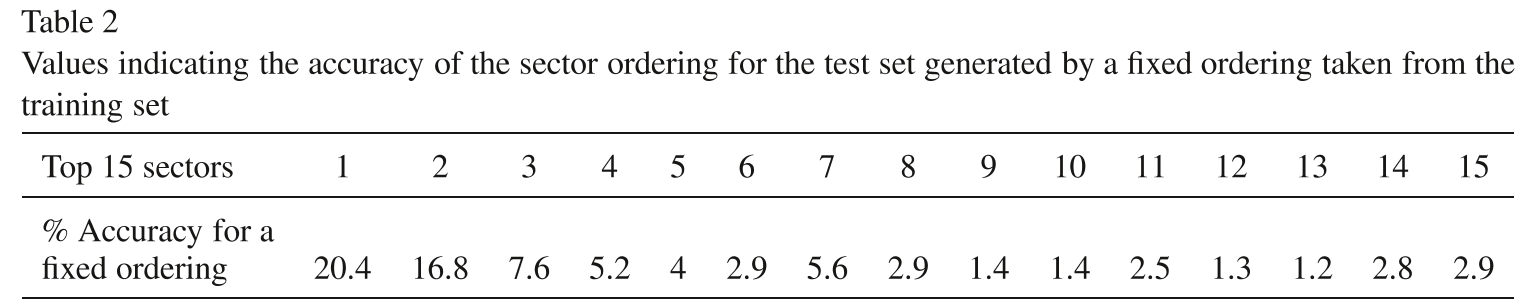
Top 15 sectors 1 2 3 4 5 6 7 8 9 10 11 12 13 14

% Accuracy for a

neural network 25.3 13 9.2 7.3 5.4 4.1 3.6 3.1 2.8 2.4 2 1.6 1.6 1.5

Average number

Table 2



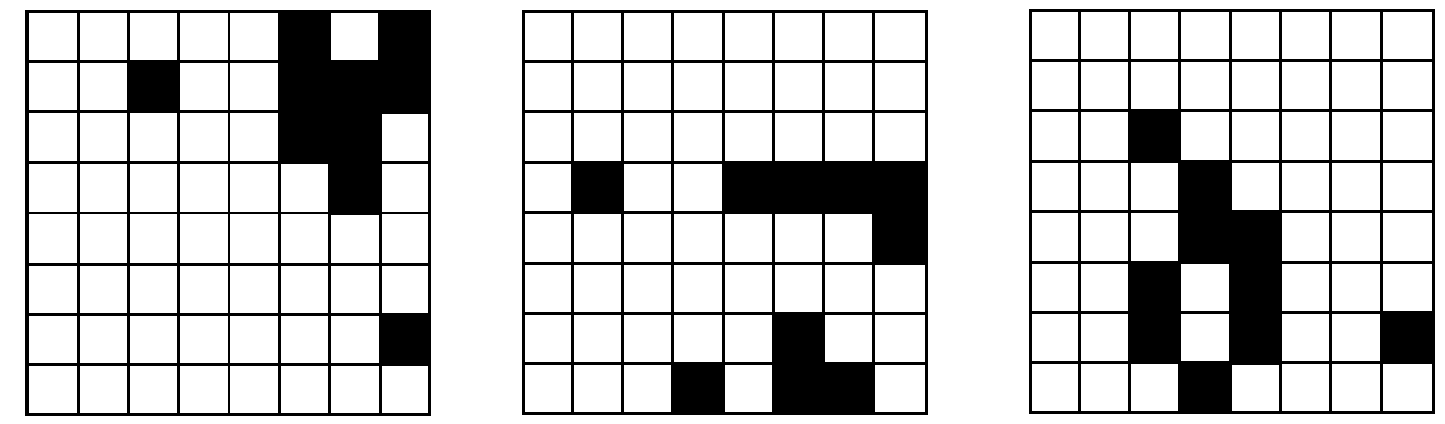
from the training set results. For the entire training set, the number of times each s was ranked first was calculated. A fixed sector ordering was then calculated by ord

the sectors on descending order of how many times they were ranked first. For exam the sector relating to the e4 square was ranked first 1372 times out of a total of 10000 so would be ordered first by the fixed sector ordering. The results of using this fixed s

ordering to predict the move influence in the test set is given in Table 2. The value actually slightly worse than using a neural network, where the difference for the top s is most significant. In this case the total for the top 4 sectors is 50% exactly and for th 6 sectors it is 56.9%, where these values are also worse than when using a neural netw but better than a random ordering.

Some other tests were also performed to try and gain further insight into the kin information that the neural network had learned. Centralisation can be tested by looki which sectors the neural network most often ranks first in any position. For the test se sectors most often ranked first in decreasing order were: e4, d3, e2, d5, d4, e5, c2, d and c5. The number of times these squares were ranked first was 1423 times out of a to 10000 for the e4 square, down to 318 times for the c5 square. So the neural network sh a strong tendency towards centralisation. In another test all chessmap values were mad same, either all 1, 0, or−1 and the neural network produced a sector ordering for this. was done to see how the neural network would interpret complete domination by e side or a completely neutral chessboard, from White’s point of view. The chessboar Figs. 4–6 show the top 10 ranked sectors from these 3 sets of input values. When W controls the whole chessboard the sectors ranked first are very far advanced, consi with an attacking strategy. When the chessboard is completely neutral the sectors are

centralised to defensive and slightly more defensive again when Black controls the w



chessboard. This suggests that the neural network might ha

The chessmaps heuristic only provides a general guide for

move ordering method behind other more precise heuristics. Because it is posit in nature, one option would be to extract tactical threats to complement the posit evaluation. With this in mind, the moves are divided into three types: forced, capture other moves. All of the information required to determine these moves is saved in chess position or when creating a chessmap, so only a small amount of extra processi required to produce this move-ordering scheme.

A forced move is defined here as one where a piece is forced to move in order to a it being captured with a loss of material. Note that this is different to the idea of a fo move in common chess terms, where a forced move is the only good or acceptable mo the position. Forced moves are ordered on descending value of the piece forced to mov king moves are considered first and pawn moves last. The forced moves for a single p are then ordered depending on which sectors they influence. Capture moves are ord on the material difference between the capturing and the captured piece, with the gre material difference in favour of the capturing side being ordered first.

All other moves (not forced or capture) are then ordered by the chessmaps heur where we look at each sector in turn and record which remaining moves influence it. group of moves for each sector can be further ordered by a heuristic called the close heuristic, which measures how close a move moves a piece to a sector. The piece mu aligned with the sector (a bishop or queen along a diagonal, and a rook or queen along a or column) and then the number of squares between the piece and the sector is calculat the piece’s closeness value. Moves for each sector are then ordered on ascending close value. Note that the influence of a move includes the case where one piece can mov of the way of another piece, so that the second piece now attacks a sector that it previo did not. This change would be recorded in the valueboard arrays and also means tha piece that has just moved may not influence the sector in question itself. As the piece moved does not directly influence the sector, it is given a default closeness value

Pawns, knights and kings can only have a closeness value of 0 (in the sector), 1 (one m away), or the default value. The closeness heuristic is not particularly accurate, but at

it allows for an ordering of the moves grouped for a sector and there is some log moving the piece closer to the relevant area. If the sector chosen is not the most impo then maybe another sector in that general area is, so this could help to concentrate fo in the correct area.

The moves can be further sub-divided into safe and unsafe, where a safe move doe result in loss of material on the square that the piece is moved to and an unsafe move d These six categories of moves can then be looked at in whatever order is considered and the ordering used for this paper is:

(1) Safe capture moves. (2) Safe forced moves.

(3) Safe other moves.

(5) Unsafe forced moves. (6) Unsafe other moves.

Appendix B gives a sector ordering and move ordering for an example chess posi

As the majority of moves are ordered by the chessmaps heuristic, it would be possib add other more precise heuristics without making the chessmaps heuristic redundant. I next section, another heuristic that is tried is experience-based and records the last s that caused a cutoff at each depth.

Testing was done on a 266 MHz Pentium PC and two sets

ordering and also with the history heuristic [11]. As the complete move ordering me is not used here, the moves are only divided into safe and unsafe and then ordered by sector ordering. All strategies performed a brute-force negamax search [10] to depth 5

a quiescence search [2] at the leaf nodes, where the quiescence search consisted of ma just the safe capture moves for each side. These were ordered on material differenc the same way as the capture moves of the complete move ordering method. Test se runs were performed on the 24 Bratko–Kopec positions [7] and the average numb nodes searched in each position is given in Table 3. For all strategies a large percentag nodes searched came from the quiescence search. The chessmaps heuristic and the his heuristic reduced the search size by nearly 80% compared to the random move orde The history heuristic searched only slightly fewer nodes than the chessmaps heuristic because of the time required for the chessmaps heuristic to generate the move ordering history heuristic can perform the search in much less time. Just over 25% less time required. However, the new definition for the influence of a move has definitely helpe

chessmaps heuristic, and reduced the number of nodes searched by approximately 9 compared to the old definition.

The second set of tests involved testing the complete move-ordering algorithm ag the currently popular approach of firstly extracting capture moves and then ordering remaining moves using the history heuristic. These tests were done on 54 posit consisting of the Bratko–Kopec positions and 30 other middlegame positions chose

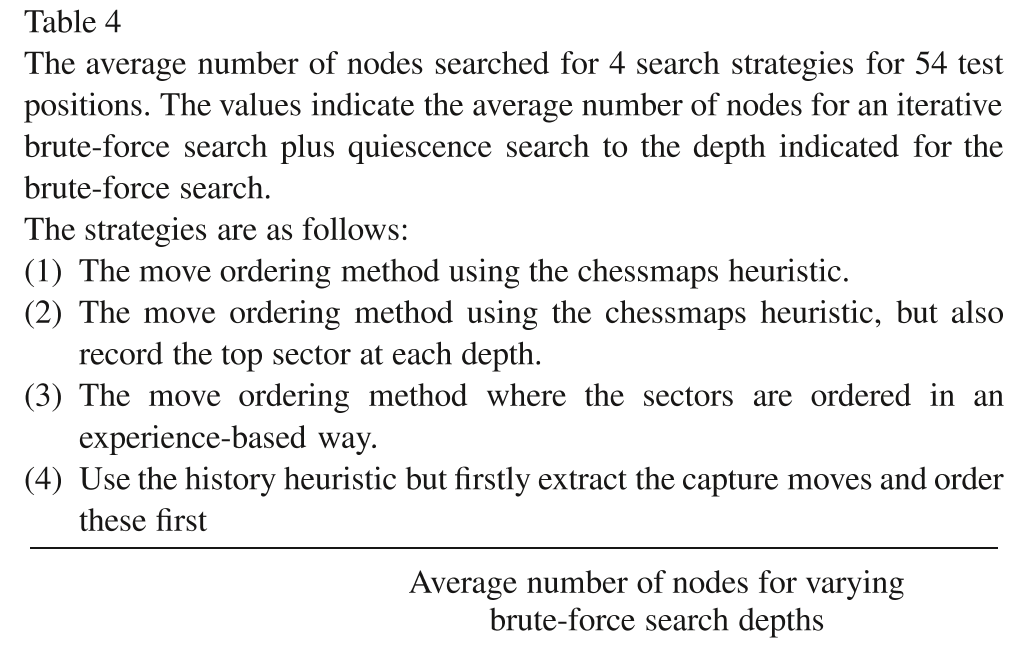
Table 3

Kopec positions for a brute-force

a quiescence search at the leaf nodes

|  |  |
| --- | --- |
| Search strategy | Average number of no |
| Random move ordering | 2173388 |
| Chessmaps heuristic | 472375 |
| History heuristic | 441240 |

Table 4



Strategy 3 3395 19683 120615 6

Strategy 4 4278 25779 155554 9

were tried, from depths 3 to 6, with a quiescence search at the leaf nodes. The result four different search strategies searching to these four depths are presented in Tab where this table again gives the average number of nodes searched in each position. first strategy was to use the move ordering method by itself. The second strategy use move ordering method, but also added another simple heuristic. This heuristic store last sector that caused a cutoff at each depth. This sector was then retrieved and ord first in any sector ordering, overriding the output of the neural network. The third stra used the move ordering method, but ordered the sectors in an experience-based way. W each node in the search tree returned a move, the sector that the move was ordere was recorded and its value in an array was incremented by 1. Thus sectors that are

important will be incremented more often and achieve larger values and this will bui a picture of where the more important areas of the chessboard are. Two arrays store sectors, one for White and one for Black; so separate strategies were recorded for side. Note that forced or capture moves did not increment any sectors if they were f to be best. The final strategy firstly extracted capture moves, which were ordered firs then ordered the remaining moves using the history heuristic. This is another experie based approach that is known to produce good results and could be used as a benchma compare other methods against. Iterative deepening [11] was also included in this set o runs, so while the experience-based heuristics would benefit from the previous iterat the knowledge-based approach of using a neural network would not. So a search to d 3 would consist of a brute-force search to depth 1 followed by a quiescence search,

a brute-force search to depth 2 followed by a quiescence search and finally a brute-

When looking at the search sizes we can see that all strategies are in the same o for all search depths. The history heuristic with capture moves (strategy 4) searche

most nodes. The move ordering method using just the chessmaps heuristic (strateg searched the second largest number of nodes. Ordering the sectors in an experience-b way (strategy 3) searched the third largest number of nodes and the strategy of ord the sectors using the chessmaps heuristic, but also recording the top sector at each d (strategy 2) searched the least number of nodes. With regard to the search times, the m ordering method using just the chessmaps heuristic (strategy 1) was the slowest. The

3 strategies were very close to each other for search depths of 3–5 ply. At a search dep 6 ply all strategies would take too long to be practical in a game-playing program, bu history heuristic with capture moves used appreciably less time. This was because o increases in the sizes of the brute-force search tree. Recording the top sector at each d would seem to be very effective and it helps to compensate for the times when the n network gets the ordering badly wrong. Without this the experience-based approach w prove to be more reliable, which may be the case anyway.

This paper suggests an

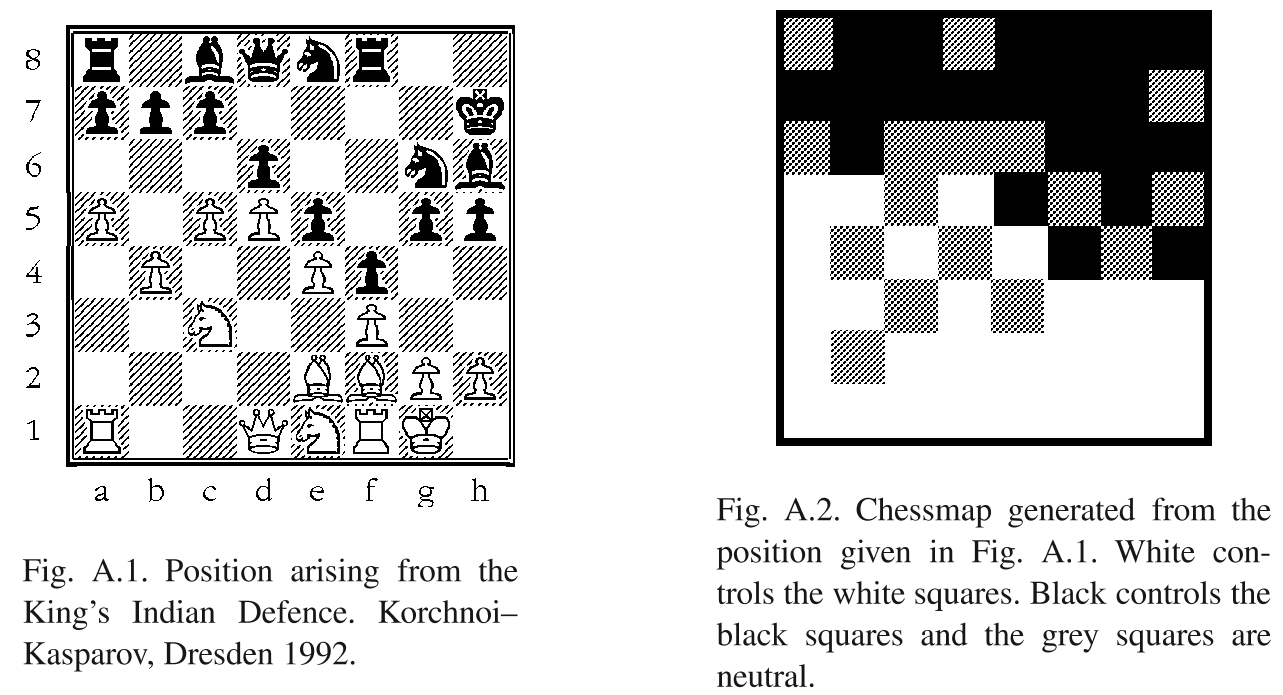
fewer nodes but does not reduce the search times. It is also a complicated algorith implement and so the history heuristic may be preferred for a chess game-playing pro if it can produce a similar or better performance. However, this new move-ord algorithm may be of interest from a research point of view. It could have the pat oriented chessmaps heuristic as its basis, which is in line with the pattern-oriented appr of human chess players. The chessmaps heuristic attempts to represent the attack-def relationships between the pieces, but only in a very general way. Another thing that hu players do is to look for immediate tactical threats in the position and this is partly cov by firstly extracting capture and forced moves. Concerning the argument for an experie based or knowledge-based approach, it would seem that the experience-based appr is still to be preferred. This can also be demonstrated in the move-ordering algor by replacing the chessmaps heuristic with an experience-based method for orderin sectors. To perform as well as the other methods, the chessmaps heuristic needs the of another simple experience-based heuristic that stores the top sector at each depth. move-ordering algorithm has not been implemented in a chess game-playing program so it is not possible to give any definite comment about its performance, but these re suggest that it might be a competitive way of producing the move ordering.

There are still various things that can be tested with this move-ordering algorithm. thing would be to try and improve the performance of the neural network. A new defin for the influence of a move has been presented in the paper, so maybe other param could also be changed. No matter how accurate the neural network is however, the s ordering can only attain a certain level of accuracy, as it only groups moves together (w

are then ordered by the closeness heuristic). So another promi

suggest specific moves.

Appendix A. Figures illustrating the control of the squares and the influence of



used in a capture sequence. Bishop1 refers to the black squared bishops and bishop2 refers to the squared bishops. Rook1 and rook2 refer to the rooks starting on a1 and a8 or h1 and h8 respective

pawn knight bishop1 bishop2 rook1 rook2 queen king

White pieces 10 29 30 31 49 50 90 200 or

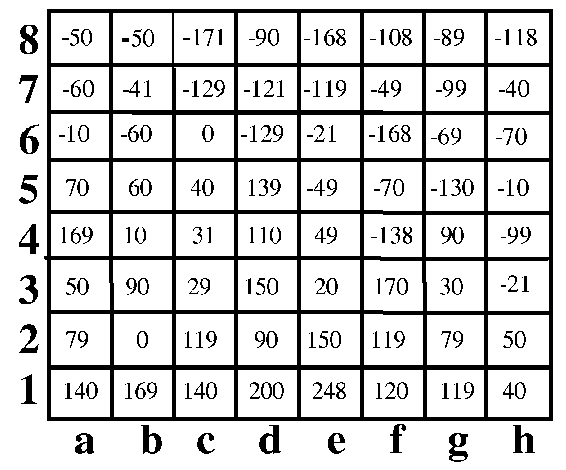


Fig. A.4. Values in the valueboard

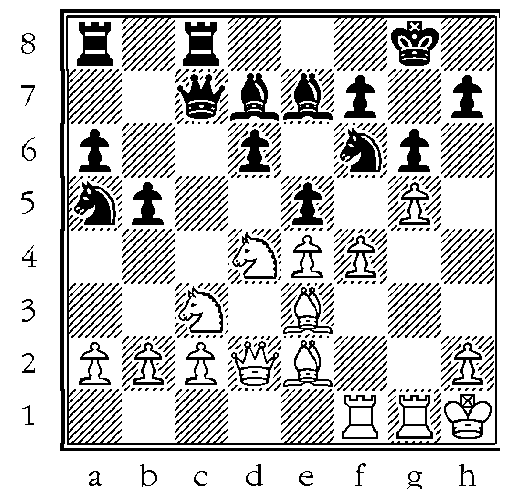
Rc1.

|  |  |
| --- | --- |
| . | A.5. The difference in the valueboards of F |

A.6. A vector representing the influence of the move Rc1 on the position in Fig. A.1. T

Fig. A.7. A vector representing the control of the squares for the chess position of Fig. A.1. No

the neural network.



sectors ordered by a neural network from the most to least impor

d4 a4 a5 h6 d3 f2 b5 c5 e8 c2 e4 a3 b2 e3 h1 e5 g3 g5 h2 b3 c3 a1 d1 b1 d2 c1 f1 a2 e g1.

The moves ordered and placed in their separate categories

WPg5× f6 WPf4× e5.

(2) Safe forced moves:

WNd4−

(3) Safe other moves:

WQd2

WNc3−d1 Wrg1−g3 WKh1−g2 WRf1−f3 WPb2−b3 WQd2

WBe2−f3 WBe2−d3 WRf1−f2 WBe2−d1 WPh2−h4 Wbe3

WQd2−e1 WRf1−d1 Wrg1−g2 WRf1−c1 WQd2−c1 WRf

WRf1−a1 WRf1−e1.

(4) Unsafe capture moves:

WNc3× b5

(5) Unsafe forced moves:

WNd4−f5

(6) Unsafe other moves:

WPh2−h WPa2−a

[1] J.A

[3] H. Berliner, C. Ebeling, Pattern knowledge and search: The SUPREM architecture, Artificial Intelligen (1989) 161–198.

[4] A.D. de Groot, Thought and Choice in Chess, Mouton, The Hague, 1965.

[5] J. Fürnkranz, Machine learning in computer chess: The next generation, ICCA J. 19 (3) (1996) 147–1 [6] K.R.C. Greer, P.C. Ojha, D.A. Bell, A pattern-oriented approach to move ordering: The chessmaps heu

ICCA J. 22 (1) (1999) 13–21.

[7] D. Kopec, I. Bratko, The Bratko–Kopec experiment: A comparison of human and computer performa

chess, in: M.R.B. Clarke (Ed.), Advances in Computer Chess 3, Pergamon Press, Oxford, 1982, pp. 5 [8] R.A. Levinson, A self-learning, pattern-oriented chess program, ICCA J. 12 (4) (1989) 207–215.

[9] E.M. Morales, Learning playing strategies in chess, Computational Intelligence 12 (1) (1996) 65–87.

[10] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley, Re MA, 1984.

[11] J. Schaeffer, The history heuristic and alpha-beta search enhancements in practice, IEEE Trans. Pattern Machine Intelligence 11 (11) (1989) 1203–1212.

[12] H.A. Simon, M. Barenfeld, Information-processing analysis of perceptual processes in problem so Psychological Review 76 (5) (1969) 473–483.

[13] D. Wilkins, Using patterns and plans in chess, Artificial Intelligence 14 (2) (1980) 165–203.