SUPER-SOMA – Solving tactical exchanges in Shogi without tree searching

Jeff Rollason

47 Rickmansworth Road, Pinner, Middx, HA5 3TJ, England

jeffrollason@cs.com

Abstract. A key feature of programs that play games such as Chess and Shogi is the ability to evaluate the outcome of threatened tactical moves. In Chess this is usually solved using a combination of tactical and capture search. This works well as exchanges rapidly simplify and a solution can usually be quickly found. In Shogi (Japanese Chess) the problem is not so simple as captured pieces are immediately available for tactical drops and so tactical threats do not quickly simplify. Since the number of tactical threats in Shogi also tends to be much larger than in Chess, then this makes solving threats using tactical and capture search much more difficult.

In the Shogi-playing program SHOTEST I have taken a different approach to this and created a tactical exchange evaluator which can statically do the work of a tactical search. This approach has its ancestry in the well-known and simple SOMA algorithm used to determine single square exchanges. However the algorithm SUPER-SOMA described in this paper can also deal with multi-square captures, pins, ties, discovered attacks, promotions, defensive play, mate threats, mate ties and even positional moves.

Keywords: shogi, shotest, soma, super-soma, evaluation, search

1 Introduction

The game of Shogi has much in common with Western Chess (see section 4). However it poses problems for the AI-programmer that make it difficult to solve. In Chess there are two competing methods for controlling tree search. These are broadly:

1 Brute force, which generally looks at most moves and uses tree-search techniques to reduce the number of nodes evaluated. Each node is a simple and quick evaluation.

2 Selective search, which uses Chess knowledge to direct search down selected paths, performing more complex evaluations of a much smaller number of nodes.

In the early days of Chess the Brute force method was generally more successful as selected techniques proved unreliable for controlling search. In more modern implementations there is now higher selectivity, but the search still examines a high proportion of all moves at the shallower plies. The number of moves examined per ply is still largely controlled by search cutoffs. This tendency to consider most legal moves is only possible because the number of legal moves is usually only between 30 and 40. In Shogi this figure is nearer 106. This figure was derived from a test I performed on 100 games between two different computer programs that had progressed to 300 or more moves. This gave a mean of 106, a median of 90 and maximum of 340 moves (the theoretical maximum is 593). I only tested moves 1 to

300, and not beyond. The distribution of these showed two distinct peaks at around 31-40 moves and 71-80 moves. Since in order to solve Shogi it is necessary to be able to work at all stages of the game, it is reasonable to find a value for the number of legal moves which will encompass the larger proportion of positions. If this boundary is set at the bottom 90%, then a figure of 150 legal moves fulfils this. Using this we can calculate the impact this will have on conventional alpha-beta search.

Calculating the minimum number of nodes for a depth-first, fixed depth alpha-beta search (Knuth and Moore, 1975):

$$N=2 \text{ x } w^{(d/2)} \qquad \text{width “w” and depth “d”} \qquad \textbf{(1)}$$

# Moves	Depth	# Nodes
30	8	1,620,000
150	8	1,012,500,000
300	8	16,200,000,000
30	12	1,458,000,000

From the table we can see that at depth 8 increasing the number of legal moves from 30 to 150 is the equivalent of increasing the depth to 12 for the same width. Shogi still needs deep searches as Chess, but these are inevitably harder.

For this reason it is almost certain that to implement a Shogi-program it is necessary to consider selective techniques if the intended Shogi program expects to evaluate deep search trees. With this in mind I determined to create a Shogi program that used highly selective search methods. To reduce 300 candidate moves to a more manageable number would require a sophisticated evaluation with a strong idea of which moves were viable to consider.

In this paper I have, conforming to my Chess background, assumed white is to play next, which is contrary to normal practice in Shogi articles where Black normally plays first.

2 Design plan for SHOTEST

The primary design plan is to create Shogi program with an evaluation function to do two key things:

1 Reduce the need for search wherever possible
2 Use the evaluation to direct the search

As a part of this I planned to create an evaluation function that would achieve lookahead of captures and tactical moves, but without performing a tree search. This would do part of the job of a capture and limited tactical search. The expectation was that such an evaluator would be less expensive than search, but less accurate. This limitation need not compromise the ability of the program as this evaluator could be used within a limited tree search. Where the evaluator could see that the position was complex it could direct the search to examine the position for further plies. Since the evaluator should have a good understanding of the threats in the position it should be able to make a good choice of moves to search, and have a good idea when a position has stabilised.

If it is viable to substitute static evaluation for conventional capture search, then such a system would replace the evaluation of a large number of simple nodes by fewer more complex evaluations. This should have extra benefits. For example in Shogi positional components often exceed material values. Having a more complex node analysis is likely to make it easier to reliably assign such high values. In practice Shotest examines 50 times fewer nodes than (for example) the program YSS in the same time (Yamashita, H - 1997). If the search can afford to examine 1/50th the number of nodes, then the positional evaluation can afford to take 50 times longer than it would in a simpler search.

As a final component, if the tactical evaluator was successful, then it might also be possible to create a predictive engine that would combine both positional and tactical moves.

3 Design plan for SUPER-SOMA

The core of the proposed work is to attempt to predict exchanges across the whole board. There is already an algorithm called SOMA that does this (Michie, D - 1966), but only considers each square in isolation. This does not allow whole board situations to be assessed. SUPER-SOMA needs to find a mechanism to cross-reference these potential exchanges in such a way that sequences of moves from different parts of the board can be predicted. This will require each SOMA exchange to be linked in such a way that choices of captures can be prioritised. The following scheme does this.

A Generate the table "XREF" to contain a list of tactical moves

- A1. Do a basic scan to determine all attacks on all squares.
- A2. Identify all pins, ties and discovered attacks. These would include ties to defending material, prevention of promotion and threats of mate.
- A3. Create the table XREF with all exchanges on the board. This would use a variant of SOMA that would understand where pieces are pinned tied or activate discovered attacks.
- A4. Add further threats to XREF including forks, attacks on immobile pieces and promotions.

B. Apply SUPER-SOMA to XREF

- B1. Apply an initial weight to each XREF entry depending on its expected net value as an isolated tactical move.
- B2. Cross reference each XREF entry, applying an enhanced weighting for each table entry based on its influence on other XREF entries. e.g. The move BxP might have the initial weight of one pawn, but if the bishop is also vulnerable to capture, then the BxP entry would be weighted as value of pawn+bishop.
- B3. Choose the best move from XREF for the side to play next. This might be a capture, promotion, fork or even a move to neutralise an opponent's threat. If there are no moves with a positive weighting then generate a "pass" move.
- B4. Update the XREF table after making the move, and change the side to play (if appropriate)
- B5. Repeat from step (B1) until both sides have played a pass move.

This complete process is quite complex. To make it comprehensible it is necessary to demonstrate some of the components in simple examples.

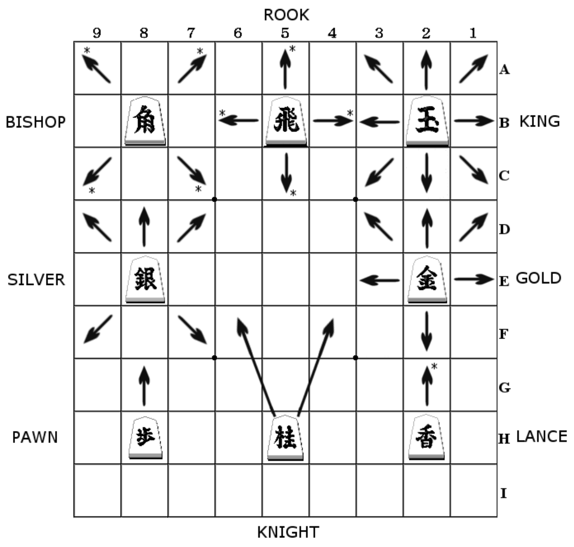
A core component in this is the simple SOMA algorithm. This is described in section 5.

4 Basic rules and notation of Shogi

Before proceeding with the body of this paper it is necessary to outline the rules of Shogi so that the examples can be understood. As indicated before, Shogi is in many respects similar to Chess with the same game objective. The key differences are: (1) the drop rule, which allows captured pieces to be dropped on the board as a move, (2) the similar but differing piece sets, (3) The 9x9 board and (4) the three rank deep promotion zone.

The examples that follow show Shogi positions using Japanese Kanji. This is hard on Western eyes but is the normal representation. I have chosen to use this in preference to westernised notation as the latter is not widely used, even by western Shogi players. I have made this slightly easier by showing actual pieces as they would be shown on a Shogi board rather than the plain Kanji notation used in Japanese Shogi texts. The latter assumes that the reader can distinguish between inverted and non-inverted Kanji characters, which is hard for new readers. The pieces and their moves are shown in Figure 1. Move arrows marked by "*" indicate multiple square moves are possible. I have not shown the pieces or moves of promoted pieces, as the examples below do not use any. The promoted versions of pawns, lances, knights and silvers all move in the same way as golds. Promoted bishops and rooks move as kings, in addition to the moves available to their unpromoted counterparts.

Figure 1. Shogi pieces and their moves



I have also adopted text notation in my discussion based on that used by Western Shogi players. It uses a coordinate system that reverses Chess by starting with a number before a letter, e.g. squares 4e, 5a etc.. rather than e4 and a5. Pieces are assigned letters, e.g. "P" pawn and "N" knight etc. Promoted pieces are shown as the original piece letter followed by a "+". Unlike Chess there are no real black and white pieces, but the idea of black and white sides is adopted to denote the side to move first and second respectively. In Shogi the same pieces are used for both sides and are distinguished on the board by the direction that they face. The pieces are pointed and lie flat on the board (rather than erect, as in Chess): A player's pieces face away and their opponent's towards the player. Promotion is achieved by turning the pieces over while keeping the direction the same. Promotion can occur when a pawn, lance, knight, silver, bishop or rook either move into, from or within the opponent's back three ranks. On the boards below, black plays from the bottom with the pieces pointing up the board.

In the text notation I have used I have also applied a small modification that shows white pieces in uppercase and black pieces in lowercase. This makes some of the discussion slightly easier to follow. Play proceeds in the same way as Chess with the object being to deliver checkmate. The drop rule allows a player to play (drop) a previously captured pieces on any part of the board as a move. Such a piece will be dropped in its unpromoted state. Knights cannot be dropped on the opponent's back 2 ranks, or lances on the opponent back rank. Pawns can also not be dropped in a file that an existing unpromoted pawn already occupies, i.e. pawns can never be doubled. These basic rules result in a game that is lively compared to Chess, with much less dependence on material values.

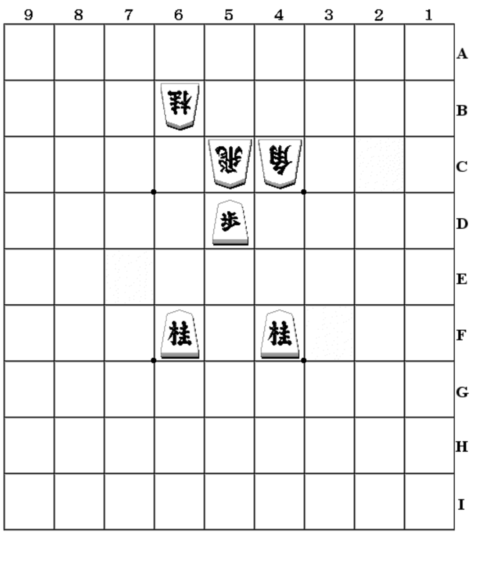
5 Basic operation of SOMA

A critical component in SUPER-SOMA is the fundamental SOMA algorithm. In order to understand SUPER-SOMA it is necessary to establish how this works, as follows:

The SOMA evaluation of exchanges on a single square require swapping off the pieces on that square, with each side having the option to stop the exchange at their most beneficial moment. For example in the position in Figure 2 the black pawn at 5d is attacked by the knight at 6b, rook at 5c and bishop at 4c, and defended by the knights at 6f and 4f. If the simplified values of P=1, N=2, S=3, G=3, B=5 and R=6 are used then the full exchange in this position would be:

Move	Net exchange value
Nxp	+1
nxN	-1
Bxn	+1
nxB	-4
Rxn	-2

Figure 2. Simple SOMA exchanges



From this exchange it can be seen that white is left having lost material. It has won 1 pawn + 2 knights, but lost a bishop + knight leaving white 2 points down. White could stop earlier by stopping the exchange Bxn. this leaves white 1 point down, which is worse. White can stop earlier by not performing the first Nxp, leaving 0 exchanges points. Therefore white is better off not making any exchange at all.

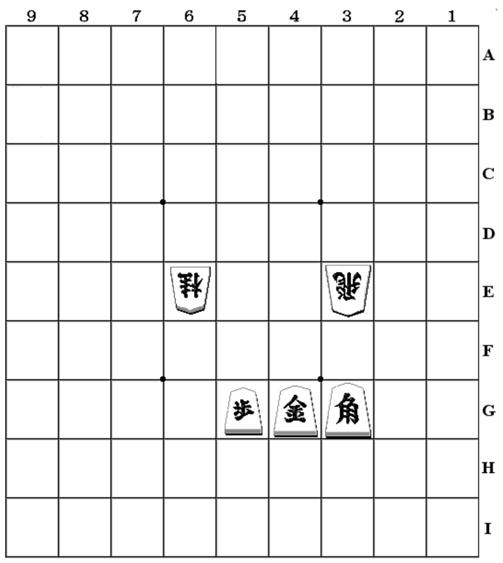
The underlying idea behind this is that each side can stop the exchange early by not making a capture. This is only worthwhile when stopping early improves on the current exchange total. Determining where the endpoint is requires iterating the exchange until neither side can find an improvement.

In Shotest this simple mechanism is made more complex because pieces change their value as they move, therefore an exchange may end with a Gold capture which leaves the Gold on an inferior square and so this results in a loss of material. An exchange may therefore gain only because it has disrupted the positions of the pieces.

6 Enhancements to simple SOMA

Shotest can also account for pins and ties in the SOMA evaluation, for example in Figure 3:

Figure 3. Pins and Ties with SOMA



Considering square 5g we have the exchange:

```
Nxp +1
gxN -1
```

Treating this as a simple SOMA exchange it is easy to see that no profitable capture is possible. However the Gold on 4g is tied to defending the Bishop on 3g, which is attacked by the Rook on 3e. The value of the tie is 5 points, the value of the Bishop. If this is incorporated into the SOMA exchange we have:

```
Nxp +1
gxN -1 +5 = +4
```

The consequence of this is that black's final move gxN leaves black with a net loss of 4 points, which is worse than no recapture. Applying the SOMA principle the exchange ends with Nxp with a net gain of 1 point for white. The tie to protecting the bishop has prevented the re-capture.

There are many subvariations possible here requiring special processing. In particular tied pieces and targets may be involved in the exchange square being considered. In these cases the tie or pin may be broken during the exchange. Another consideration is that pieces that are pinned or tied may effect the ordering of the exchange. If the first capturing piece is a tied knight and the second piece is a free rook, then the capture sequence may be re-ordered. Finally the target of a capture may be tied, in which case the XREF entry may generate a second triggered entry which only becomes active when the parent capture occurs. Discovered attacks are also processed in a similar fashion.

7 Whole board analysis using SUPER-SOMA with the XREF table

The examples above consider only captures around single squares. The SUPER-SOMA algorithm allows separate exchanges on the board to be linked and prioritised. The two examples below start with a simple case to demonstrate the principle. In these examples I have turned off the variable value of piece material as would make the examples harder to understand. I have instead assumed the following
piece values:

```
P   L   N   S   G   B   R   K
52  120  120  200  220  360  480 3999

P+   L+   N+   B+   R+
      240 230  230  460  550
```

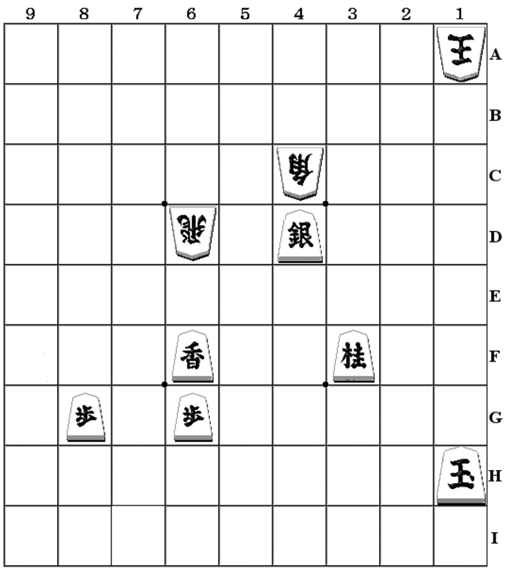
All examples assume white is to play next

8 Example 1 of whole board analysis

In the example in Figure 2 SUPER-SOMA finds 6 moves for the XREF table, as follows:

Move	Value	Type of move
l 6fxR6d	960	capture
s 4dxB4c	740	capture and promote
R 6dxs4d	-560	capture
R 6dxl6f	-720	capture
B 4cxp8g	152	capture and promote
s 4d- 5c	20	promote

Figure 2. Example 1 - Whole board analysis and SOMA



From this we can see that the greatest value move is L 6fxR6d with an exchange value of 960 points (value of black losing rook plus value of white gaining rook in hand). This is a move for black though and white is to play next. White's most valuable move is the capture and promotion move B 4cxp8g. White's other moves are R 6dxl6f, which loses a Rook for a Lance and R 6dxs4d losing a Rook for a Silver, so both have negative values.

If you apply SUPER-SOMA to this position it predicts the following:

```
      R 6dxs4d    400
      n 3fxR4d    960
      B 4cxp8g    152
       pass
       pass
      Net value -408
```

SUPER-SOMA has predicted that the best move is the sacrifice of the Rook for the Silver. In Shogi terms this makes sense as simply moving the Rook away from the threat by the lance will result in black then capturing the Bishop with the Silver, whereas sacrificing the Rook for the Silver also prevents the capture of the Bishop, so this is the correct result.

We can look to see how SUPER-SOMA does this.

8.1 First Iteration of SUPER-SOMA - Example 1

The following is the table XREF used by Shotest as it would appear after the first step "B3" above. It needs some explanation:

```
                    Key for table XREF:

The header BDRI+SRDX below shows "-" for "off" and lowercase
letter for "on"

B = Threat can be blocked
D = Threat can be defended by some other piece
R = Threatened piece can run from threat
I = Threatened piece has no safe moves (Immobile)
+ = Move gives check
S = Threat can be neutralised by some means (BDR above)
R = Target is tied to defending another piece
D = Threat does not immediately capture, e.g. Fork
X = Entry is currently active (may be triggered later)

Val  = Net value of exchange
1st  = Value if exchange after first capture
Weight = Weighted priority after cross-reference "B2" above
        If followed by "n" then this is a neutralising step
        rather than a capture
```

Table XREF:

BDRI+SRDX			Val	1st	Weight	
--r--s--x	l	6fxR6d	960	960	960	n
--r--s--x	s	4dxB4c	740	740	944	n
--r--s--x	R	6dxs4d	-560	400	1160	
--r--s--x	R	6dxl6f	-720	240	240	
--r--s--x	B	4cxp8g	152	152	892	
-d---s--x	s	4d- 5c+	20	20	20	n

From the table above we can see how the top weighted move is R 6dxs4d with 1160 points, despite its low initial value. If we examine the cross-referencing that occurred in step "B2" we can understand how this value was derived.

```
Step B2:  Cross-referencing of table

B 4cxp8g gives  152 to s 4dxB4c  capturing piece is our target
l 6fxR6d gives  960 to R 6dxs4d  our piece is target of other capture
s 4dxB4c gives  740 to R 6dxs4d  capturing piece is our target
s 4d -5c gives   20 to R 6dxs4d  capturing piece is our target
l 6fxR6d gives  960 to R 6dxl6f  our piece is target of other capture
                                 capturing piece is our target
s 4dxB4c gives  740 to B 4cxp8g  our piece is target of other capture
```

The first choice is R 6dxs4d as previously indicated, but the second choice is to simply neutralise the move l 6fxR6d by moving the Rook away. The first "R" column is marked by "r" indicating that the Rook can safely run and so this neutralising move is possible. In some cases a threat cannot be fully neutralised, e.g. if the piece has no safe squares then it may simply need to be defended. This might prevent the capture, or may just allow re-capture of the attacking piece, for example LxR above could be defended by a pawn drop behind the Rook. This would not stop the loss of the Rook, but would allow capture of the Lance in compensation. This partial defence would be reflected in the cross-reference weighting given.

The third choice above is the initially positive capture B 4cxp8g. SUPER-SOMA will therefore predict the move R 6dxs4d above and proceed to step "B4" to update the table.

8.2 Second Iteration of SUPER-SOMA - Example 1

After step B4 above and iterating B1, B2 and B3 again, the XREF will now contain the following:

BDRI+SRDX	Val	1st	Weight	
--r--s--x n 3fxR4d	960	960	960	
--r--s--x B 4cxp8g	152	152	152	n

Now four of the table entries have gone as they are now void, for example the capture s 4dxB4c is no longer possible as the Silver has been captured. Black is now the side to play. There are only two moves and these are not linked. Black can either neutralise the capture B 4cxp8g by moving the pawn or capture the Rook on 4d. The latter has a much higher weight and so is chosen.

8.3 Third Iteration of SUPER-SOMA - Example 1

After n 3fxR4d the table contains just one move. This has a positive value and is selected. After this move both black and white will play pass moves and the sequence ends.

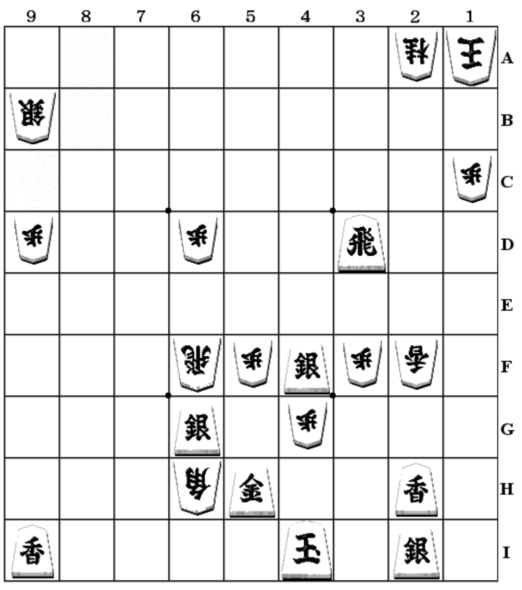
BDRI+SRDX	Val	1st	Weight
--r--s--x B 4cxp8g	152	152	152

8.4 Discussion of Example 1

The example above is relatively simple, but demonstrates the basic mechanism of SUPER-SOMA. This position would be very easy to resolve using a simple capture search.

9 Example 2 of whole board analysis

Figure 3. Example 2 – Further whole board analysis and SOMA. White has 1G and 1B in hand



The following example looks at a position with many more tactical features. In Figure 4 we have a much more complex example, which includes defensive and forking moves and ties to mate and promotion. This rather artificial example, which does not pretend to resemble a realistic game position, has been contrived to demonstrate a wide range of SUPER-SOMA features within one compact example. This makes it possible to show how SUPER-SOMA links these features together. In this particularly volatile example there is a 14-ply mate threat, which SUPER-SOMA

does not statically detect, but it would not be reasonable to expect SUPER-SOMA to be able to do this
A key feature in this position is the Gold on 5h which is tied to defending the threat of a Gold drop on 4h giving mate. Therefore the Gold cannot capture the Bishop on 6h or defend the Silver on 6g. This uses a static assessment of the mate threat based on the simple use of bitmaps. To a limited extent some multiple move mates can also be detected, and also unproven or incomplete mate threats, e.g. a square may be threatened by a Gold drop which leaves the king with two free squares. The evaluator may tie the defender to this square with a value less than mate (e.g. a value of a Silver). This assessment of mate threats could use a tree search, but this an expensive component to build into a node evaluator.

There are no less than 9 pins and ties in the position. These are:

```
l9i --> P9d --- S9b Pin  (Pawn cannot move outside plane of pin)
r3d --> P6d --- R6f Tie
s6g --> P5f --- R6f Tie
R6f --> s6g --- g5h Tie
G*4h--> -4h --- g5h Tie to mate threat
P5f --> -5g --- s4f Tie to prevent promotion
P3f --> -3g --- s4f Tie to prevent promotion
L2f --> L2h --- s2i Pin
L2f --> L2h --- s2i Tie
```

SUPER-SOMA predicts the following move sequence:

```
 R 6fxs6g+   470
 r 3dxP3f    104  <-- this is the key move
 L 2fxl2h    240
 s 2ixL2h    240
(l 9ixP9d)     0  neutralise threat pass
 B 6h- 7i+   100
  pass
  pass              Net value 456
```

9.1 First Iteration of SUPER-SOMA - Example 2

We look at the moves in XREF ordered by initial value. Since there are many moves I have separated the white moves out first.

```
BDRI+SRDX            Val  1st Weight

----+s--x B*1fxr3d  960  960    960    White
-d---s--x B 6hxs4f  500  500    876
--r--s--x R 6fxs6g  470  470   1430
--r--s--- P 3f- 3g  376  376    188  (inactive)
--------x B 6h- 7i+ 100  100    100
--r--s--x L 2fxl2h    0  240    240
--------x R 6f- 6e -104    0      0
```

The top move here is the fork B*1fxr3d (B*1f, R1fxr3d) forking Rook and King. This move has no cross-reference with other moves so its final weight and initial value are the same.

The move B 6hxs4f is weighted from 500 to 876 because it is removing the Silver at 4f which is tied to preventing the promotion of the Pawn at 3f to becoming a Tokin at 3g. If this move gets played then the 4th entry P 3f- 3g above would become activated in the table.

The third move R 6fxs6g is the actual move chosen. It is weighted from 470 to 1430 because it stops the capture s 6gxR6f.

The remaining moves are not interesting. The only cross-referenced entry is L 2fxl2h which is weighted because it prevents l 2hxL2f.

We can now look at the black moves in the table:

```
BDRI+SRDX            Val  1st Weight

-d---s--x s 6gxR6f  960  960   1430   n  Black
bdr--s--x l 2hxL2f  240  240    240   n
bdr--s--x l 9ixP9d  104  104    104   n
bd---s--x r 3dxP3f  104  104   1564   n
bd---s--x r 3d- 3a   70   70   1030   n
--r--s--x s 6gxP5f -192  104    278   n
--------x g 5hxP4g -312  104   -312
--r--s--x r 3dxP6d -752  104    208   n
```

The first move s 6gxR6f is the opposite of R 6fxs6g above and receives the same weighting. Note that to neutralise this move by simply moving the Rook away would be a viable choice for white, although the rook is both tied to defending the pawns on 6d and 5f. In this instance white has other better moves.

The second and third moves have no cross-reference linkage. The fourth r 3dxP3f is the top weighted move and also the most interesting as it is weighted for stopping the fork B*1fxr3d and also defending the Silver on 4f, preventing B 6hxs4f. This is the first example of a move being credited for performing a defensive move. This is complex behaviour way beyond simple SOMA. The fifth move r 3d- 3a promotes the Rook and prevents the fork. The remaining moves are not interesting.

SUPER-SOMA will select the top white move R 6fxs6g and update the table.

9.1 Further Iterations of SUPER-SOMA - Example 2

The second iteration retains the same top black move as found after iteration 1, the move r 3dxP3f defending the Silver and avoiding the fork.

Iteration 3 selects the move L 2gxp2h which generates a threat against the Silver on 2i. The XREF table entry for this white move is converted into the black move s 2ixL2h for the next iteration.

Iteration 4 selects the re-capture above. Iteration 5 find white with no good attacking moves and so the black move l 9ixP9d is neutralised by moving the Silver to 9c. Iteration 6 and black has no moves, so passes. Iteration 7 and white simply promotes the Bishop by B 6h- 7i, and the sequence is ended after both sides pass.

You may like to input this position into your own Shogi program. If this position is analysed by Shotest with tree search it predicts the forced win:

```
1. [  1] B  * 1f
```

```
 2. [  1] S 2i-3h
 3. [  1] L 2fx2h+
 4. [  1] G 5hx4g
 5. [  4] R 6fx6g+
 6. [  1] P  * 1b
 7. [  1] K 1ax1b
 8. [  1] R 3d-3b+
 9. [  1] K 1b-1a
10. [ 10] R+3bx2a
11. [  1] K 1ax2a
12. [  6] N  * 3c
13. [  2] K 2a-3b
14. [  1] K 4i-4h
```

after examining 4857 nodes. The numbers in brackets are the position in the search that that move was considered. For example 10. [10] R+3bx2a above indicates that 9 other moves were considered before R+3bx2a.

9 Extensions to SUPER-SOMA

The two examples above demonstrate the most important features of the SUPER-SOMA algorithm and show how the general mechanism is used to cross reference and predict tactical sequences. However there are many other features that are not shown in these examples, as they would have made them unnecessarily complex.

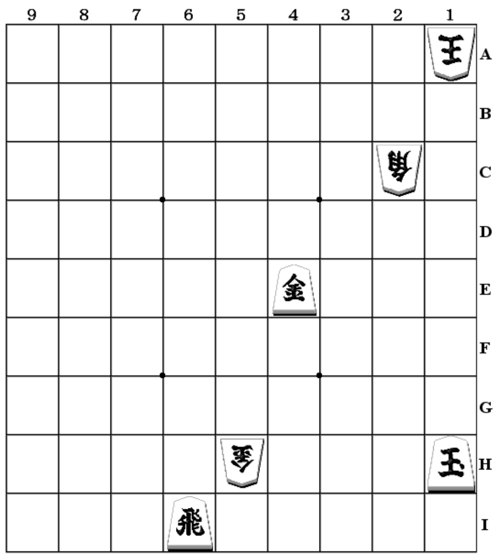
Some of these extra features are demonstrated below within simpler examples, as the general mechanism has already been described. In all cases there are no pieces in hand.

10.1 Checks

A check is a special kind of tactical move that requires a forced response. This complicates the assessment of the move to be chosen. In the position in Figure 4 there are 4 moves in the table:

Figure 4. Dealing with check

BDRI+SRDX		Val
--r--s--x	G 5hxr6i	960
-d--+s--x	B 2cxg4e	440
--------x	g 4ex-3e	-152
----+---x	r 6ix-6a+	70



The only cross reference is G 5hxr6i prevents the promotion move r 6i-6a+. This leaves G 5hxr6i as the top move. However SUPER-SOMA chooses B 2cxg4e before G 5hxr6i because the bishop capture gives check which, after the king moves, leaves white with the next move. The cross-reference recognises this and forces the move selection by simply adding a large weight to B 2cxg4e.

10.2 Secondary captures

A move which captures a piece may then also generate a further capture threat which can be entered into the table. This is illustrated in Figure 5.

This generates the following table entries.

```
BDRI+SRDX           Val

--------x B 2bxl9i+ 330
--r--s--- I 9ixn8i  240 (inactive entry)
--r--s--x l 9ix-9a  110
```

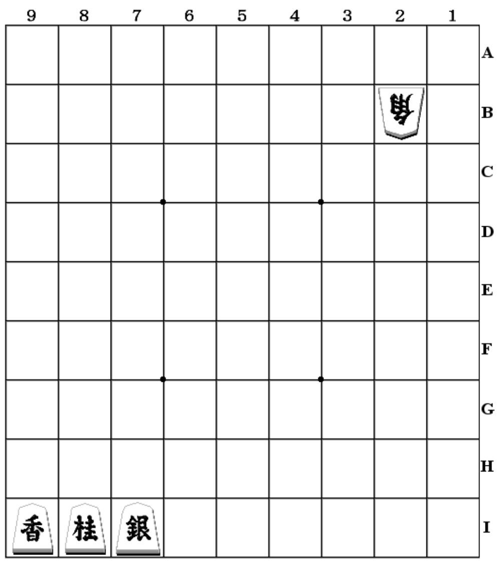


Figure 5. Generation of further captures

In this case the table starts with two active and one inactive entry. When the move B 2bxl9i+ with promotion is selected the follow-on potential capture entry I 9ixn8i is activated (triggered). During cross-reference the initial move is credited with the value of the triggered entry. Once activated the new move is thereafter fully cross referenced as any other table entry. This mechanism is limited as the potential extra move B+8ixs7i is not recognised, as this mechanism only examines to a depth of one level. To try and extend this further does not really make sense as it becomes increasingly complex to detect deeper captures because of pieces moving, gaining or losing protection and blocking / unblocking of attacks. The results of a deeper application of this idea would be increasingly susceptible to errors.

The more recent version of this mechanism is also used to create secondary captures when a discovered attack or pin/tie is activated.

10.3 Multiple Attacks

This is perhaps obvious, but a capture on a square may be by one of several pieces, therefore each possible first attacking piece is considered as a separate entry. This allows a heavily committed piece to not be selected to capture, even though it is the smallest piece. This becomes critical when the first obvious choice to capture can also capture another piece, in which case an uncommitted piece captures first.

In the example in Figure 6 the silver might capture first, but it is committed to attacking the gold, therefore the bishop captures first. A variation on this occurs where a piece is tied to defending another piece, but in this instance the favourite choice to make the capture is selected first anyway and would generally work even with a single entry in the table per capture square.

Figure 6. Multiple attacks

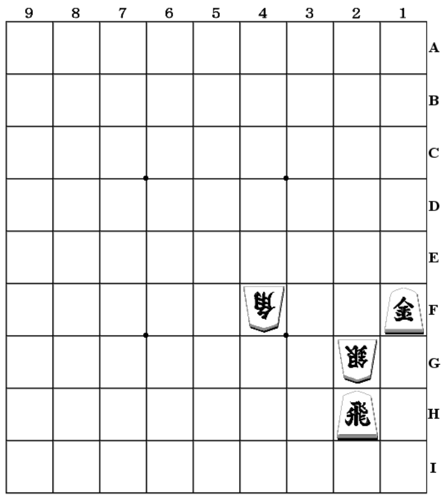
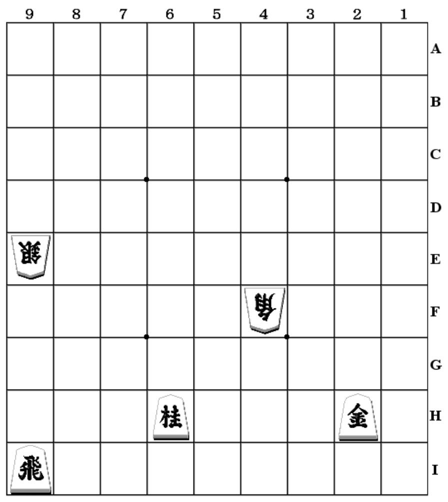


Figure 7. Defending moves



10.4 Defending moves

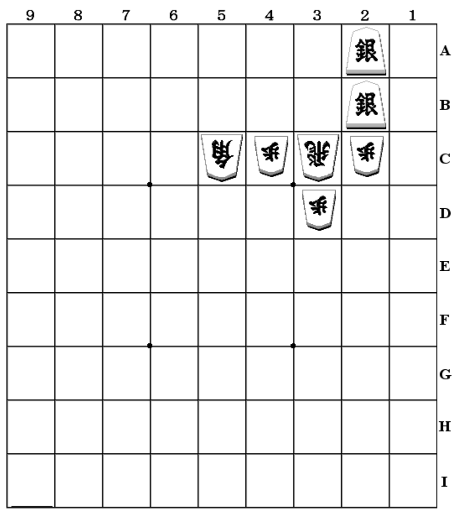
This is similar to the idea of creating secondary attacks, but differs in the sense that this information is only generated after a complete XREF table has already been created. This looks at tactical moves and detects whether the move has some secondary defensive role. For example in the board in Figure 7 the bishop on 4f might capture the gold on 2h. If instead it captures the knight on 6h, it also defends the silver on 9e, so this will be the preferred move.

10.5 Running to safer squares

A piece may be attacked but have no safe square to run to. In this instance it may be beneficial to move to a square where a recapture is possible.

In the position in Figure 8 the rook on 3c is attacked by the silver on 2b and is not defended. The rook can legally move to 3b, where it will still be captured without re-capture. However it can also move to 3a, which is defended by the bishop. In this instance this is the selected move as it results

Figure 8. Running to safer square



in some material compensation after the rook is lost. This move would be hard to generate in a conventional capture search.

10 Correcting errors in SUPER-SOMA

A clear flaw in the mechanism described so far is its dependence on getting the correct choice of move at each iteration. A tree-search can experiment with moves selected without any strong need for picking the correct move as its first choice to search. It is clear that it would not be hard to create example positions where SUPER-SOMA picks the wrong line.

The first consideration here might be to find out how often the selected first move is actually the best choice. The correct way of doing this would be to run a series of tests that compared the choice made by the algorithm with the results of a full search. Note that this would not be a simple capture search, but would instead have to include non-capturing tactical moves. It would therefore be a full search. This experiment has not yet been done, but an intermediate test has been performed.

The current version of this algorithm runs twice for each node. The second run simply rejects the first choice made by the first run and then predicts a second sequence based on a different first move. The program then compares the results of these two tests and in minimax style chooses the move with the highest score.

In tests run over a complete game the first choice move was retained in about 95% of the positions analysed. This only considered positions where there were at least two moves that seemed worth playing, i.e. it would not compare a situation where the first move was obviously good, but the second choice was a meaningless sacrifice. The result indicates that the algorithm seems to be happy with the first choice most of the time. Of course this result depends on an incestuous test as the testing mechanism has the same intrinsic flaws as the mechanism being tested. However this test is still valid as the tree-search is much more exhaustive and will almost always find the correct move, even if errors occur in the tree.

12 Using SUPER-SOMA with tree search

Limitations on the length of this paper do not allow a thorough explanation of how this operates, so the following gives a general outline of how SUPER-SOMA is used.

The tree search used with SUPER-SOMA uses a plausibility analysis that orders moves to examine at the next ply. SUPER-SOMA adds an ordered list of likely moves to this, which may include defensive as well as capturing and other tactical moves. These suggested moves get very high priority in move choice. In addition to this the search frequently chooses to terminate early. SUPER-SOMA provides a estimation of the likely outcome of the exchanges in a position and this is used to assess whether the position is likely to fall outside of the alpha-beta window. The complexity of the position is also assessed by a linear value which sums the complexity of the line predicted by SUPER-SOMA. This is also used to determine whether it is safe to attempt to terminate the search.

SUPER-SOMA is particularly effective in generating defensive and blocking moves. A capture search cannot do this and a full width search may be overwhelmed by huge moves lists that are just too expensive to search. A general purpose search may be able to spot all defensive moves, thus narrowing the full list, but to routinely search all of these is still too expensive. SUPER-SOMA will decide whether active tactical or defensive moves are the best choice and thereby reduce the chance that inappropriate lines are searched.

13 Comparing SUPER-SOMA with other techniques

From the examples above it can be seen that SUPER-SOMA is sophisticated. It can obviously detect complex features, but is limited in its capacity to correct errors made in choosing moves. This means that each choice of move is very critical as there are limited options to try out alternatives.

It is not easy to compare SUPER-SOMA with other techniques because its function does not exactly match any alternative method. Also it is not used as a complete alternative to search-based methods, as indicated in the previous section, but is intended to be used in co-operation with a searching mechanism.

On a simple level the basic SUPER-SOMA can be compared to capture search and also to general tactical search. I will consider these in turn.

13.1 Capture Search

In positions where simple captures dominate then SUPER-SOMA looks inadequate as the move selection is not very deep. SUPER-SOMA cannot deal with captures that trigger a long chain of captures. The later version of the algorithm is capable of generating secondary captures, but only for one generation. For example, a promoted bishop capturing on 2b can then detect the possibility of a further capture on 1a, but not the following capture than might occur on 2a.

In compensation for this it can predict other types of moves such as forks, mates, promotions and defensive moves; predicting plausible move sequences. These are common situations which capture search cannot easily deal with.

13.2 Tactical Search

A normal tactical search is always going to out-perform SUPER-SOMA alone, as SUPER-SOMA is limited to considering a single line of moves and will make many mistakes. However SUPER-SOMA can make a reasonable assessment very quickly. If SUPER-SOMA is used inside a selective search then it may well be much more effective than tactical search because at any node in the tree it can still make reasonable predictions for the outcome of tactical sequences.

14 Development of SUPER-SOMA

The design of this mechanism has been driven by the nature of the position types that it expects to have to analyse in normal Shogi games. For example I have not tried to create an algorithm that can deal reliably with situations with very long chains of linked captures, which would test my algorithm to the limits. I have instead considered the types of position that commonly occur and tried to deal with these. The guiding consideration is therefore a question of probability: A simple feature might be analysed in great depth simply because it is very commonly met. This not a purist approach to development. It has allowed compromises to be made to simplify the mechanism at the expense of true generality. A more general approach would be more satisfying but would probably make the algorithm too complex to be practical and therefore would not result in the creation of a viable playing algorithm.

15 SUPER-SOMA in use

SUPER-SOMA has been in use in the program Shotest since version 2.0. This program is unbalanced as it still needs much more Shogi knowledge. Despite this Shotest came 3rd twice in the 1998 and 1999 CSA World Computer Championships and 7th in 2000, competing with between 35 and 45 programs. The experience of these events indicates that Shotest currently performs badly in the opening and endgame, where Shogi knowledge is important, but that it performs well in the middlegame where positions can become very complex. Even though Shotest examines some 50 times fewer nodes than the other top programs, it seems to play well in this stage of the game. On a PII-400 Shotest only analyses about 3000 nodes/sec. It is likely that Shotest's match performance is somewhat impaired by the lack of very strong Shogi knowledge to the development, and my own weak knowledge of the game. Another substantial deficiency is the lack of any tsume (mate) search, which instead depends on general purpose search to detect mates.

A likely large part of Shotest's success is owed to the ability of SUPER-SOMA to handle complex positions without needing to grow enormous game trees. This is important in Shogi as late middlegame and endgame positions commonly leave large numbers of pieces hanging. This is shown in Figure 9 from examining 2365 positions from 50 randomly selected human games, ranging from amateur to top profession level. This calculates the number of squares where a profitable captures, promotions and forks can be made, starting from move 50 to avoid the quiet opening phase.

```
Number of captures                                               Frequency
11 *                                                                     6
10 **                                                                   14
 9 ****                                                                 26
 8 ********                                                             54
 7 **************                                                      101
 6 ************************                                            168
 5 ******************************                                      214
 4 **********************************************                      329
 3 *********************************************************           406
 2 ****************************************************************    457
 1 **************************************************                  357
 0 *********************************                                   233
```

Figure 9. Number of captures, promotions and forks from move 50 onwards in 50 human games

From the figures in Figure 9 it can be calculated that 90% of positions have profitable captures. Over 75% of positions have 2 or more captures and 38% have 4 or more. This contrasts sharply with Chess where a limited hand examination of just 20 games gave a figure of 24% for positions with profitable captures with only 5% with 2 or more profitable captures.

Figure 10 shows another similar test over 50 Shotest games against other programs, which interestingly shows that positions with no captures are much more common than in the human games in Figure 9. These figures give 79% with profitable captures, 59% with 2 or more and 28% with 4 or more. This may reflect Shotest's inclination to resolve tactical threats rather than leave them open.

```
Number of captures                                              Frequency
11                                                                      3
10 *                                                                   10
 9 *                                                                   18
 8 **                                                                  34
 7 *******                                                             99
 6 *************                                                      181
 5 ************************                                           341
 4 **************************************                             540
 3 **********************************************                     654
 2 ******************************************************             778
 1 *************************************************************      873
 0 ****************************************************************   919
```

Figure 10. Number of captures, promotions and forks from move 50 onwards in 50 computer games

The broad conclusion from this is that tactically active positions are the norm in Shogi. If the evaluation can deal with this then this can greatly reduce the burden on tree search.

Whether or not SUPER-SOMA works well depends on how reliably it can predict moves. In the combined sample of the 100 games above SUPER-SOMA predicted the next move in the game score 65% of the time. This only considered moves where either SUPER-SOMA or the game record predicted an active move (i.e. a defensive or attacking move). Of the failures, 37% occurred when a pawn capture was predicted, but was left hanging in the actual game record.

In use, SUPER-SOMA averages a lookahead of 4.2 plies at the end of the primary continuation over a large number of games. This is quite deep considering that in the early part of the game in many positions there are no predicted captures. SUPER-SOMA commonly predicts sequences of 10 moves or more. When playing to a time limit of 3 seconds per move (on a PII-400) this average depth extends the average 6 ply primary depth to 10.2 plies.

Since SUPER-SOMA predicts a sequence of moves, it can be used with some positional elements. At present each piece has a value assigned for each square it can occupy, both for its material value and also its value in the currently selected castle. These values can be traced inside SUPER-SOMA, so that an exchange will also change positional values. This allows SUPER-SOMA to predict sequences that disrupt a castle structure. This could probably be extended in many ways, as yet unexplored. At present, of course, the castle mechanism is badly tuned, so this extension currently has limited value. Shotest also has other features that strongly contribute to its success, but these are outside the scope of this paper.

16 Conclusions

As far as I am aware, SUPER-SOMA represents the first attempt to generate an evaluation function that can completely analyse a position in chess-like games, which is accurate enough to allow plausible play without speculative tree-search. Many programs exist that can evaluation positional features, but such programs cannot cope with exchange threats without search.

I originally tested this idea 20 years ago with the Chess-playing program Merlin. The mechanism then was crude and it seemed that the narrow trees generated by Chess favoured other methods, so the idea fell into disuse. The new implementation for Computer Shogi is much more elaborate and has not been exhaustively compared with other methods, but it at least seems to perform well. The existing interface with the tree-search needs improvement. The implementation is still young and so the full potential of this mechanism has not yet been fully realised. When SUPER-SOMA is more developed I will be able to perform more rigorous comparative tests.

In conclusion I would assess that SUPER-SOMA with directed search looks as if can make a viable alternative to conventional tactical search. Its sophistication makes it very good for reliably detecting quiescence in a position. It also makes it much easier to generate good moves deep in the tree. Conventional brute force methods for doing this depend on exhaustively detecting good moves in one branch so that they can be tried in neighbouring branches. SUPER-SOMA can reach a completely new position deep in the tree and immediately have good candidate moves to search.

The greatest weakness in SUPER-SOMA is its limited single lines of analysis. Currently SUPER-SOMA allows two alternative moves to be considered for the first move. This is a somewhat limited concession to correcting errors, but this mechanism is usually backed by real search. As SUPER-SOMA is developed it should be able to predict where it is likely to make errors and indicate to the search that the position is unstable and should be searched further.

As more work is done it will be easier to assess whether this or more conventional methods are the way to go for Computer Shogi.

References

1. Knuth, D.E. and Moore, R.W. (1975), An Analysis of Alpha-Beta Pruning, *Artificial Intelligence*, Vol. 6, pp. 293-236. ISSN 0004-3702

2. Yamashita, H and Matsubara, H (1998).Computer Shogi – ISBN4-320-02892-9\2300

3. Michie D (1966): Game Playing and Game Learning Automata. In Advances in Programming and Non-Numerical Computation, Ed. Fox L, pp 183-200. Oxford, Pergamon.