Moving in the Dark: Progress through Uncertainty in Kriegspiel[1]

C0156

Abstract.

Kriegspiel is a Chess variant similar to wargames, introducing a factor of uncertainty in the original game. Kriegspiel increases the difficulty of Chess by hiding from each player his opponent's moves: in fact the two players play on different boards. A player only knows the position of his own pieces, while his opponent's pieces are "in the dark", ie. they are invisible. A referee evaluates a player's try and replies to both players with a message that contains some information, but not the real move. For instance, a try may result *legal* or *illegal*, and a legal move may lead to a *capture* or a *check* announcement. Thus a Kriegspiel player has to guess the state of the game being in the dark about the history of moves, but he can exploit the referee's messages.

Computer playing of Kriegspiel is difficult, because a program has to progress in the game even with scarce information on its opponent's position. In fact, in our knowledge there are no programs able to play Kriegspiel well.

This paper describes the rationale of a program to play some simple positions of Kriegspiel using a gametree-based approach. We show how we implement an evaluation function able to progress through uncertainty.

1 Introduction

Kriegspiel is a Chess variant similar to wargames. The players are not informed of their opponent's moves: they try a move being "in the dark", i.e. knowing nothing about the position of the opponent. Although it's a two person game, it needs a referee, who knows the real situation and whose task consists in accepting the legal moves and rejecting the illegal ones, with respect to the real situation. As the game progresses, each player tries to guess the position of his opponent's pieces by trying moves to which the referee can respond saying *"illegal"*, saying *"check"* or *"capture"*, or remaining *silent*. Both players can hear all referee's announcements, but they make different inferences from such statements.

Thus, Kriegspiel players have very partial and different knowledge about the current state of the game because usually a player does not know what his opponent has moved. Therefore Kriegspiel is a game of imperfect information, where players deal with *Metapositions*, namely positions where there in uncertainty about the position of one or several pieces. Normal Chess programs can be adapted to play Kriegspiel, however a novel problem has to be addressed. Evaluation functions for Chess positions compute a score evaluating both armies, whose position is well known, and then the minimax procedure progresses maximizing or minimizing the difference of score assigned to each army. In Kriegspiel this optimization is not possible, so progress (namely improving the program's army position with respect to the position of the adversary) becomes a problem. A player has to move being in the dark about the position of the enemy army.

In this article we propose a search algorithm which explores a Kriegspiel game tree made of nodes which are metapositions, and uses an evaluation function in order to judge each node and to implement a progress heuristic. We will deal with the basic endgames of Kriegspiel, i.e. those where a player (we assume Black) has left the King only. Thus, in the next sections we will consider White having a King and a Rook (in the ♔♖♚ ending), a King and a Queen (♔♕♚), a King and two Bishops (♔♗♗♚), a King, a Bishop, and a Knight (♔♗♘♚).

This paper has the following structure. In section 3 we describe a way to represent uncertainty using metapositions and the adjustments done on the game tree. In section 4 we propose the evaluation function for basic endgames. In section 5 we describe the search algorithm which use the evaluation function and in section 6 we refer to a rule based procedure proposed in [2]. Finally, in section 7 we make some tests in order to evaluate our approach.

2 Related works

Kriegspiel was invented in England in 1896, in order to have a Chess-based game similar to wargames. It has been a favorite game of well known game-theorists like John Nash or Lloyd Shapley. Although it is a fascinating game, played by hundreds of people every day on the Internet Chess Club, only a small number of papers have studied some aspects of Kriegspiel or Kriegspiel-like games. Below we provide some instances of related work.

Boyce proposed a procedure to solve the ♔♖♚ ending, that we have implemented to be able to evaluate our algorithm [2] . Ferguson analysed the endings ♔♗♘♚ ([6]) and ♔♗♗♚ ([7]), respectively. In ([]) it has been shown a rule-based program to play the ♔♙♚ ending according to some principles of game theory. Sakuta and Iida ([8], [9]) described a program to solve Kriegspiel-like problems in Shogi (Japanese Chess). Bud, Albrecht, Nicholson and Zukerman ([4], [3]) described an approach to the design of a computer player for a sub-game of Kriegspiel, called Invisible Chess. Finally, in ([]) it has been described a research on ♔♖♚ endings in Kriegspiel.

3 Metapositions

A *metaposition* is a position able to denote a *set* of normal Chess positions. For example, in the miniature depicted in figure 1 on the

[1] this paper is not under review or accepted for publication in another conference or journal

left, White is not sure where the Black king could be: multiple Black Kings represent possible positions of the King itself.

Metamoves are moves that the Black King can make and that lead him from a metaposition to another. Therefore a metamove allows White to update his reference board expanding all the possibilities for the Black.

We will also use the term *pseudomoves* to indicate those moves by White on a metaposition.

A Kriegspiel position can be described by a pair of diagrams, one for each for players; each diagram is a metaposition which represents the knowledge assumed by a player. We will refer to these metapositions using the term *reference boards*, which are boards annotated with all the possible positions of opponent's pieces.

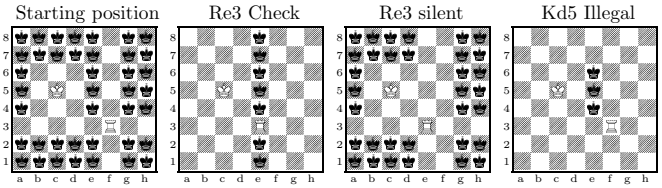

Figure 1. Possible referee's answers

Figure 1 shows on left an initial position where White has Kc5 and Rf3. We consider the reference board with greatest uncertainty, that is a board where each square not controlled by White might contain a Black king.

If White tries to move his rook to e3 and the referee says *Check*, he will update his reference board and he will assume that the Black king possible position is on the White's rook row or column, as shown on the second diagram of figure 1. If White tries to move his rook to e3, but the referee remains silent, he will update his reference board cleaning the squares around the king and along the Rook row or column, as depicted on the third diagram of figure 1. Finally, an attempt may be illegal because White tries to move his king to a square which is occupied by the Black king. In this case, White will consider as possible Black king positions those around his king. The rightmost miniature in figure 1 shows White's reference board updated after he tried to move his king to c5 and he received the illegal warning from the referee.

3.1 Number of metapositions

In order to highlight the numerical complexity of dealing with uncertainty by means of metapositions we calculate the number of metapositions for the rook ending. For Chess this number is about 28000 positions as shown in [5], while for Kriegspiel it can be calculated by fixing the position of White's pieces and considering the number of the ways to choose n Black king's positions among the remaining positions, which are not controlled by White. If we assume, as a worst case for White, rook on a1 and king on b1, we have 52 possible positions which are not controlled by White and the total number of metapositions became

$$\sum_{1 \leq n \leq 52} \binom{52}{n} = 2^{52} - 1 \qquad (1)$$

Thus the number of metapositions is extremely large. The reflections of the Black king position with respect to the diagonal a1 to h8, as described in [1], do not decrease the numerical complexity of the problem and prevent us from building a database of metapositions.

3.2 Game tree reduction

By merging all the Black King's positions into a single one, a first consequence is the transformation of the game tree. The number of pseudomoves is equal to the greatest number of legal moves that White can try. For example, on the board depicted in figure 2 White can move his King in seven different squares if the Black King is on g4 or h4, otherwise he can choose among only five squares. In this case the number of pseduomoves is the maximum between 7 and 5.

Therefore White has to try several attempts to guess the right opponent's position. The number of metamoves is constant, namely one, and it is just the expansion of the Black King's possibilities. We will distinguish the pseudomoves by the referee's answer, which can be *silent* (S), *check* (C) or *illegal* (I). During the search visit on the game tree, we give a first evaluation of the metaposition we are analyzing, using the function we discuss in Section 4, then we choose to expand the tree only on the worst answer we can get from the referee.

After each actual move White applies the metamove to his own reference board. Thus, the tree's branching depends on the pseudomoves and, for each pseudomove, on the worst answer that White could receive.

Figure 2 shows an instance of a game tree. With x, x', y, y', z, z' we indicate the vote given by the evaluation function to the metapositions reached after playing each pseudomove. This is done considering the referee's answers and choosing the worst one, that is the one with smaller grade.

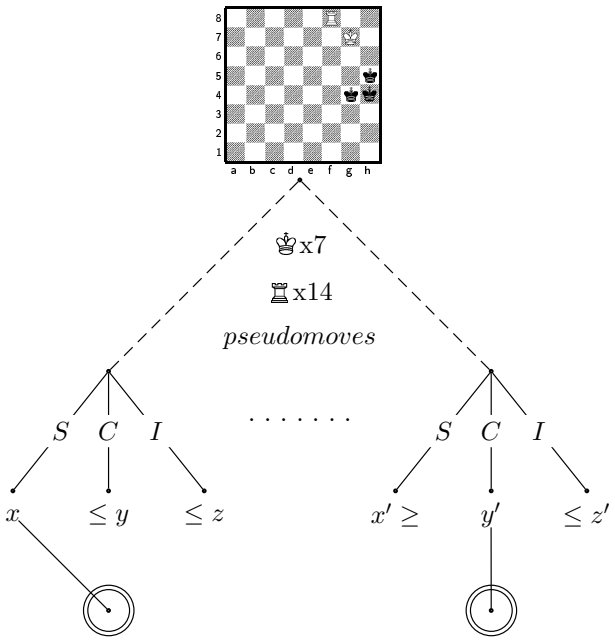

Figure 2. Game tree for Kriegspiel

4 The evaluation function

The evaluation function contains the rules which synthesize the notion of progress able to lead a player towards the victory. It is a linear weighted sum of features like the following

$$\text{EVAL}(m) = w_1 f_1(m) + w_2 f_2(m) + ... + w_n f_n(m) \quad (2)$$

where, for a given metaposition m, w_n indicates the weight assigned to a particular subfunction f_n. For example, a weight might be $w_1 = -1$ and $f_1(s)$ may indicate the number of Black kings.

The EVAL function is different according to each single ending, but it has some invariant properties: it avoids playing those moves that lead to stalemate and it immediately returns the move which gives directly checkmate, if it exists.

In the following subsections we briefly describe the evaluation function used for the basic Kriegspiel endings.

4.1 The rook ending (♔♖♚)

The evaluation function for this ending considers $n = 6$ different features.

1. it avoids jeopardizing the Rook: $w_1 = -1000$ and f_1 is a boolean function which is true if thw White Rook is under attack;
2. it brings the two Kings closer: $w_2 = -1$ and f_2 returns the distance (number of squares) between the two kings;
3. it reduces the number of Black Kings on the quadrants of the board as seen from the Rook: $w_3 = -1$ and $f_3 = c\sum_{i=1}^{4} q_i$ where $c \in \{1, 2, 3, 4\}$ is a constant which counts the quadrants that contains a Black king and q_i counts the number of possible Black kings on i^{th} quadrant;
4. it avoids the Black king to go between White rook and White king: $w_5 = -500$ and f_5 is a boolean function which returns true if the Black king is inside the rectangle formed by White king and White rook on two opposite corners;
5. it keeps the White pieces close to each other: $w_5 = +1$ and f_5 is a boolean function which returns true if the rook is adjacent to the king;
6. it pushes the Black King toward the corner of the board: $w_6 = +1$ and $f_6 = \sum_{i=0}^{63} v[i]$, where v is a numerical 64-element vector, shown in figure 3, that returns a grade for each squares which possibly holds the Black king or returns 0 otherwise.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -4 & -4 & -4 & -4 & 0 & 0 \\ 0 & 0 & -2 & -4 & -4 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figure 3. The numerical matrix $v[]$

4.2 The queen ending (♔♕♚)

The evaluation function is similar to the one described in section 4.1 where we consider the queen instead of the rook and we use two more features. Thus $n = 8$ and in the initial six the function is the same as in the rook case, while in the last two:

7. it penalizes those metapositions with a big number of Black kings: $w_7 = -1$ and f_7 is equal to the number of Black kings on White's reference board;
8. it reduces the number of Black Kings on the areas traced by the queen's diagonals: $w_8 = -1$ and $f_8 = c\sum_{i=1}^{4} a_i$ where $c \in \{1, 2, 3, 4\}$ is a constant which counts the areas that contains a Black king and a_i counts the number of possible Black kings on i^{th} area.

4.3 The ending with two bishops (♔♗♗♚)

In this ending we have to deal with two White pieces besides the King. The evaluation function exploits the same subfunctions previously analyzed, but it assigns different weights.

1. it avoids jeopardizing the Bishop: $w_1 = -1000$ and f_1 is a boolean function which is true if White bishop is under attack;
2. it brings the two Kings closer: $w_2 = -1$ and f_2 returns the distance (number of squares) between the two kings;
3. it avoids the Black king to go between White rook and White bishop: $w_3 = -500$ and f_3 is a boolean function which returns true if the Black king is inside the rectangle formed by king and bishop row or king and bishop column;
4. it keeps White bishops closer: $w_4 = +2$ and f_4 is a boolean function which returns true if the bishops are adjacent to each other;
5. it pushes the Black King toward the corner of the board: $w_5 = +1$ and $f_5 = \sum_{i=0}^{63} b[i]$, where b is a numerical 64-element vector, shown in figure 4, that returns a grade for each squares which possibly holds the Black king or returns 0 otherwise.

$$\begin{pmatrix} 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -100 & -40 & -40 & -50 & -50 & -40 & -40 & -100 \\ -50 & -40 & -40 & -40 & -40 & -40 & -40 & -50 \\ -10 & -10 & -40 & -40 & -40 & -40 & -10 & -10 \\ 0 & -10 & -50 & -100 & -100 & -50 & -10 & 0 \end{pmatrix}$$

Figure 4. The numerical matrix $b[]$

6. it keeps White king on the bishop's row or column: $w_6 = +1$and f_6 is a boolean function which returns true if the king and the bishop are on the same row or column;
7. it reduces the number of Black Kings on the areas traced by the bishop's diagonals: $f_7 = c\sum_{i=1}^{4} a_i$ where $c \in \{1, 2, 3, 4\}$ counts the areas that contains a Black king and a_i counts the number of possible Black kings on i^{th} area, and

   if $f_6(m) < -600$ $w_8 = -4$;

   otherwise $w_8 = \frac{1}{6}$
8. it prefers some particular positioning (we will refer to with the term *key bishops' positions*) for the White king and bishops, highlighted in figure 5; for example ♔c7, ♗c4 and ♗c5. Therefore $w_8 = +30$ and f_8 is a boolean function which is true if the bishops and the king are arranged in one of the key positions.

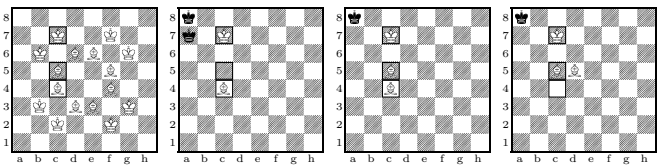

Figure 5. Key bishops' positions

4.4 The ending with bishop and knight (♔♗♘♚)

The evaluation function for the White Bishop is the same as in section 4.3. For the Knight we can't consider any division of the board,

so the evaluation for this chessman consists in reducing the number of Black kings on White's reference board and in supporting the bishop.

We also used a large set of key metapositions similar to those for the bishops ending shown in figure 5. Unfortunately, this was not enough to obtain a good evaluation function for the ♔♗♘♚ endgame, as we will explain in section 7.

5 The search algorithm

As seen in section 3.2 we consider that each node of the game tree consists of a metaposition. For example, suppose that White's reference board is the one depicted in figure 2 and that it's White turn to move. The search algorithm proceeds by generating all the pseudomoves and, for each metaposition reached, it creates three new metapositions according to the three possible answers from the referee. Then it chooses the one with smaller grade assigned to it by the evaluation function. In the example we have 21 pseudomoves which leads to 63 metapositions, but after filtering the information from the referee we obtain again 21 nodes.

Then, if the search algorithm has reached the desired search depth it simply returns the grade which refers to the best node, that is the max value, otherwise it applies the metamove on each nodes, it decrements the depth of search and it recursively calls itself obtaining a value from the subtree.

Finally, it retracts the pseudomove played and adds to the metaposition's grade the vote which is returned by the recursive call. Then it updates the max on that particular search depth.

When the algorithm terminates visiting the tree, it returns the best pseudomove to play. Since it may happen that the same candidate pseudomove is proposed in two different sequential turns to move, the algorithm avoids to choose those pseudomoves, which appear in the history of recently played moves.

6 A rule-based implementation for the ♔♖♚ ending

In order to evaluate our algorithm, we have implemented a rule based program which plays the procedure proposed in [2] to win the Rook ending. Boyce showed a way to force checkmate by considering positions where both Kings are in the same quadrant of the board as seen from the rook, or where the Black King is restricted to one or two quadrants of the board. Thus, we have implemented a program which uses a search algorithm and a small evaluation function with the aim to obtain an initial position, in order to apply the rule based procedure from [2].

7 Tests and comparisons

7.1 Rook ending comparison

Figure 6 shows a graph which depicts the result of all the 28000 matches which can occur considering all the possible initial metapositions for the rook ending from the White's point of view, starting with greatest uncertainty, that is starting from metapositions where each square not controlled by White may contain a Black king. The number of matches won is on the ordinate and the number of moves needed to win each match is on the abscissa. The graphic shows the distribution of the matches won normalized to 1000.

The rule based program (described in Section 6) spends the first 25 moves looking for one of the initial positions; when it reaches one

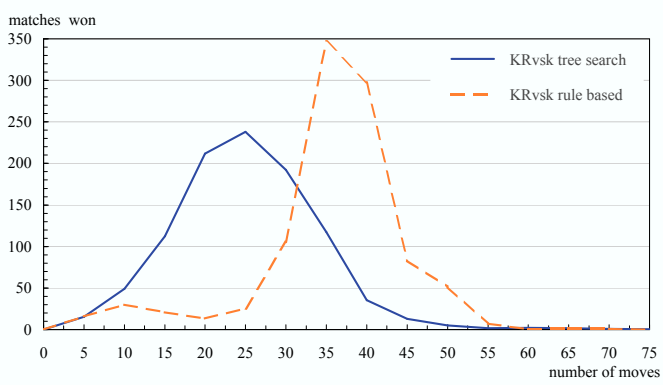

Figure 6. The rook ending comparison

of these positions the checkmate procedure is used and the program wins very quickly. However, the average of moves needed is around 35. Our program based entirely on the search of game tree wins with a better average, around 25 moves.

This is due to the fact that the program analyzes from the beginning each position trying to progress to checkmate. On the other hand, the rule-based program is faster in deciding the move to choose, with to the tree-searching program. The rule-based program has a constant running time, whereas the second one has a running time exponential on the game tree depth.

7.2 Evaluating the search algorithm

Figure 7 shows a comparison between the search algorithm and the evaluation function when analyzing some different basic endings. We performed a test choosing random metapositions with greatest uncertainty for ♔♕♚, ♔♗♗♚, and ♔♗♘♚ endings; then we normalized the results to 1000 and we merged them to produce the ♔♖♚ figure.

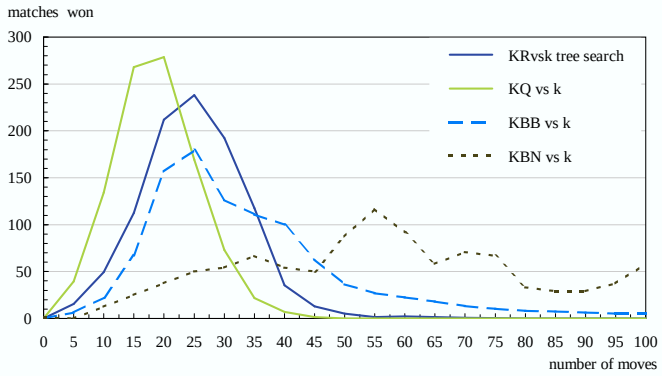

Figure 7. Basic endings comparison

In figure 7 we see that the program wins the ♔♕♚ ending quicker than the ♔♖♚ ending. This result was expected, because the queen is more powerful than the rook: the queen controls more space so metapositions have a lesser degree of uncertainty.

The case ♔♗♗♚ is instead more difficult with respect to ♔♖♚. In fact, the former is won on average in a larger number of moves: sometimes our program needs more than 100 moves.

Finally, we see that the behavior of our program in the ♔♗♘♚ ending is not good at all. The program often spends more than 100 moves to win and the distribution of victories does not converge: we conclude that in this ending our program is not really able to progress.

7.3 Progress through Uncertainty

An effective way to analyze the progress toward victory consists in considering how the value of White's reference board changes after playing each pseudomove. Figure 8 shows the trend of evaluations assigned to each reference board reached during a whole match for the ♔♕♚ ending. The number of attempts needed during the game is shown on the abscissa, while the grades assigned by the evaluation function are on the ordinate.

We see that, at each step, the value of metapositions increases. From White's point of view, this means that the state of the game is improving and this is actually a good approximation for the real situation.

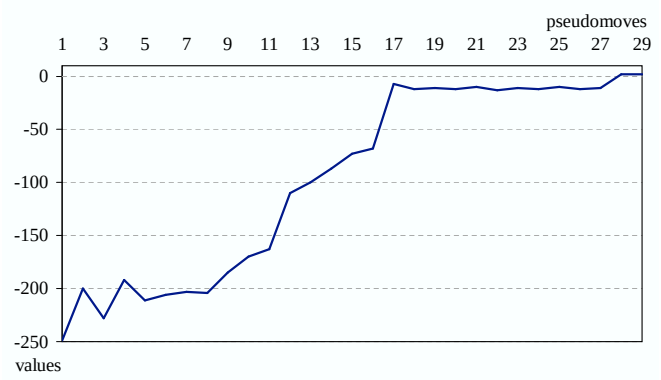

Figure 8. Trend of evaluations assigned to metapositions crossed during ♔♕♚ ending

We have performed the same test for the case of ♔♗♘♚ ending, whose result is depicted in Figure 9. Here the progress is not controlled by White, in fact he does not improve the state of the game at each step. The graph shows how the evaluations of the reference board change during a match which ends with the win of White: the value of metapositions does not increase at each pseudomove, but at some lucky situation for White. Thus the program basically wins by chance, that is by exploiting lucky metapositions or its opponent's errors.

We conclude that our program is able to progress to victory when we deal with pieces able to divide the board in separate areas, which can then be reduced to trap the Black King; whereas when we use a piece which has not this peculiarity, like the Knight, the behavior of the program is not fully satisfactory.

8 Conclusions

This is the first time that an evaluation function has been defined for Kriegspiel. We have devoted special care to implement a notion of progress inside such an evaluation function. We have tested such a function on some simple endings, with good results except for the ♔♗♘♚ case. Future work will lead us to adapt the program to more complex endings, where both players have a larger number of

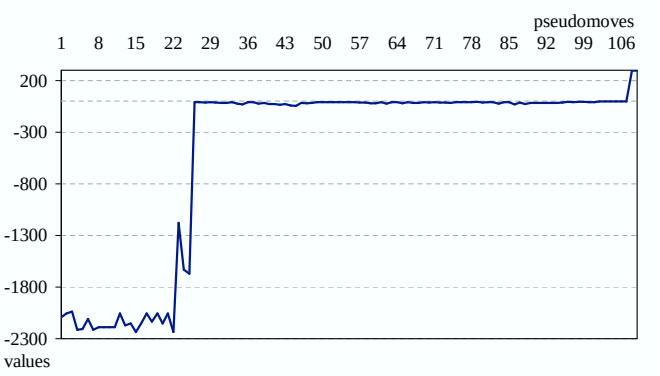

Figure 9. Trend of evaluations assigned to metapositions crossed during ♔♗♘♚ ending

pieces on the board. Our aim consists in writing a complete program for the whole game of Kriegspiel.

REFERENCES

[1] M. Bain, *Learning Logical Exceptions in Chess*, Ph.D. dissertation, Dept. of Statistics and Modelling Science, University of Strathclyde, Glasgow, Scotland, 1994.

[2] Jim Boyce, 'A Kriegspiel Endgame', in *The Mathematical Gardner*, ed., D. Klarner, 28–36, Prindle, Weber & Smith, (1981).

[3] A. Bud, D. Albrecht, A. Nicholson, and I. Zukerman, 'Information-theoretic Advisors in Invisible Chess', in *Proc. Artificial Intelligence and Statistics 2001 (AISTATS 2001)*, pp. 157–162, Florida, USA, (2001). Morgan Kaufman Publishers.

[4] A. Bud, D. Albrecht, A. Nicholson, and I. Zukerman, 'Playing "Invisible Chess" with Information-Theoretic Advisors', in *Proc. 2001 AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*, pp. 6–15, California, USA, (2001). American Association for Artificial Intelligence.

[5] M.R.B. Clarke, 'A Quantitative Study of King and Pawn against King', in *Proc. First Conf. on Advances in Computer Chess*, ed., M.R.B. Clarke, pp. 108–118, Edinbourgh, Scotland, (1977). Edinbourgh University Press.

[6] T. Ferguson, 'Mate with Bishop and Knight in Kriegspiel', *Theoretical Computer Science*, **96**, 389–403, (1992).

[7] T. Ferguson, 'Mate with two Bishops in Kriegspiel', Technical report, UCLA, (1995).

[8] M. Sakuta, *Deterministic Solving of Problems with Uncertainty*, Ph.D. dissertation, Shizuoka University, Japan, 2001.

[9] M. Sakuta and H. Iida, 'Solving Kriegspiel-like Problems: Exploiting a Transposition Table', *ICCA Journal*, **23**(4), 218–229, (2000).