Solving Kriegspiel endings with brute force: the case of KR vs. K

Paolo Ciancarini and Gian Piero Favini

Dipartimento di Scienze dell'Informazione, University of Bologna, Italy
{ciancarini,favini}@cs.unibo.it

Abstract. Retrograde analysis is a tool for reconstructing a game tree starting from its leaves; with these techniques one can solve specific subsets of a complex game, achieving optimal play in these situations, for example a chess endgame. Position values can then be stored in "tablebases" for instant access, as is the norm in professional chess programs. While this technique is supposed to be only used in games of perfect information, this paper shows that retrograde analysis can be applied to certain Kriegspiel (invisible chess) endgames such as King and Rook versus King. Using brute force and a suitable data representation, one can achieve perfect play, with perfection meaning fastest checkmate in the worst case and without making any assumptions on the opponent.

1 Introduction

In a zero-sum game of perfect information, Zermelo's theorem [1] ensures that there is a perfect strategy allowing either player to obtain a guaranteed minimum reward. In many games, discovering the perfect strategy seems to be synonymous with exploring a major portion of the game tree, which is unfeasible under current and foreseeable computer technology. On the other hand, it is possible to explore significant subsets of the game tree in such a way that, if a particular position is encountered during gameplay, its value has already been computed and the best strategy is immediately available. Most serious programs for playing chess include a so-called "endgame tablebase". Unlike opening books, the same tablebase can freely be used by any number of programs even under tournament conditions, on the basis that it contains no creative work but simply large amounts of processor time.

Currently, tablebases exist for all six-piece chess endings, with seven-piece positions in the process of being computed for the next few years. In many cases, the perfection of tablebase-powered play is unapproachable by even the strongest evaluation function, or indeed the strongest human player. Positions that most experts would have considered draws turn out to be mates in 300 or 500 moves, and seemingly hopeless games can be drawn by repetition. Tablebases are usually obtained through retrograde analysis. Analysis starts from final nodes, the leaves in the game tree corresponding to checkmates and stalemates, and then moves backwards in time to find out how those positions were obtained,

until all positions of the desired type have been explored. The concept has been widely studied since the '60s, so there is a large bibliography devoted to chess tablebases and their creation. We cite, for example, Bellman's seminal paper [2] and Stiller's systematic work on a series of chess endgames in [3].

The aim of this paper is to show that the same concept can be usefully applied to a game of imperfect information, as well, though it is only limited to finding situations where a player can force victory with probability 1. We use Kriegspiel, or invisible chess, as an example. The game is identical to chess, except players can only see their own pieces and need to rely on messages from a referee to figure out where the opponent is. We give an algorithm for solving Kriegspiel endings that have so far only been approached with approximated or heuristic methods, and use it to build a Kriegspiel tablebase for the King and Rook versus King (KRK) ending.

The paper is structured as follows. In section 2, we describe Kriegspiel and summarize previous research in the field. Section 3 contains the actual algorithm, as well as considerations on its correctness, complexity and optimizations. Section 4 is about the actual run of the algorithm and construction of the tablebase. Finally, conclusions and future perspectives are given in Section 5.

2 Kriegspiel

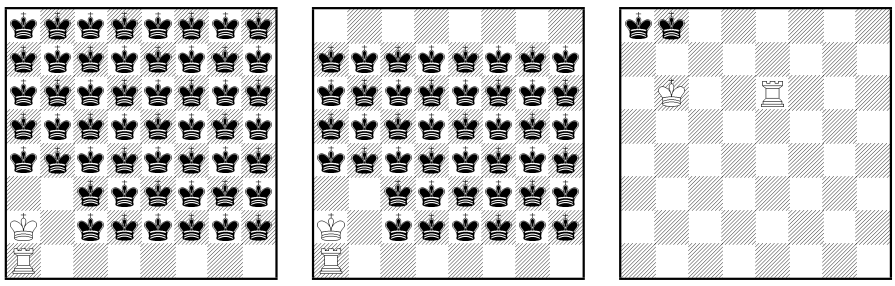
Kriegspiel is a chess variant invented at the end of the 19th century to make chess more like the 'war game' used by the Prussian army to train its officers. It is played on three chessboards in different rooms, one for either player and one for the referee. From the referee's point of view, a game of Kriegspiel is a game of chess. The players, however, can only see their own pieces and communicate their move attempts to the referee, so that there is no direct communication between them. If a move is illegal, the referee will ask the player to choose a different one. If it is legal, the referee will instead inform both players as to the consequences of that move, if any. Kriegspiel is not a standardized game, as there are several known sub-variants to the game; they differ in how much information the referee shares with the player with respect to pawn moves and captured pieces. Since our main concern in this paper is with a pawn-less endgame and a single capturable piece, the choice of ruleset is irrelevant. It suffices to remember that the referee will inform the players whenever a check or capture happens, when a move is illegal, and when the game ends.

The nature of Kriegspiel, being so similar to chess in some ways and yet completely different in others, caught the attention of some famous computer scientists and was known and played at the RAND institute. Several endings have been studied, though so far always with the aid of heuristics or *ad hoc* considerations. For example, [4] deals with KPK using a set of directives and distinguishes between algorithmically won endings, which can always be won, and statistically won ones, wherein victory is only achieved with probability $1 - \epsilon$, with ϵ small (arbitrarily small in the absence of the 50 move rule). It is shown that certain instances of the KPK ending are of the former type, and

some are of the latter. Ferguson studied two less common endings, KBNK in [5] and KBBK in [6]. These can be won algorithmically, provided White can set up his pieces in particular patterns and the black king is confined to certain areas of the board.

KRK is the most widely studied ending, probably because it is so simple in chess but not so simple in Kriegspiel, even though it is always won if White can secure his rook. Magari [7] gave an algorithm for solving KRK starting from a special position, and was the first to think of the black king as a 'quantic wave' whose actual location is not determined until the white pieces have moved. His definition would anticipate Sakuta's research in [8], which dealt with invisible Shogi and introduced the idea of metapositions. Boyce [9] had previously given his own algorithm for solving KRK, expressed as a series of directives in the natural language and without any formalization. While Boyce's conditions are more general than Magari's, the problem of reaching the starting position remains. Neither algorithm is shown to be optimal for White. Finally, Bolognesi and Ciancarini used metapositions, *ad hoc* evaluation functions and minimax-like tree search in [10] to solve KRK in the general case, showing it to perform better than Boyce's directives. However, success with this method cannot be guaranteed without trying out every single case. As a side note, Shapley and Matros improvised a solution to KRK in Kriegspiel on an infinite board while attending the ninth Game Theory convention in 1998.

This topic is a subset of the more general problem of checkmating the opponent in Kriegspiel, either forcing the mate or maximizing one's chance of doing so. Aside from Sakuta's aforementioned work on Shogi, this direction was explored by Russell and Wolfe in [11], who focused on efficient handling of large belief states in Kriegspiel. Research by Amir, Nance and Vogel is marginally related in that they attempt to reconstruct the state of the board in [12]. The even more general problem of computer Kriegspiel is beyond the scope of this paper; we refer to such papers as [13] and [14] for two completely different takes on the subject.



(a) (b) (c)

Fig. 1. (a) Highest uncertainty in KRK; (b) useless information: after two plies this board will be identical to the outcome of the same plies on (a); (c) mate in one.

3 A retrograde analysis algorithm

Let us formalize the problem as follows. $\mathbf{S}$ is the set of all possible game states (in this case, the set of legal chess positions limited to the KRK ending), and $i \in \mathbf{I} \subseteq P(\mathbf{S})$ is, at any given time, the information set for the player with the rook, who will be assumed to be White from now on, and contains all possible game states at this point in time. Elements of $\mathbf{I}$ will often be called *metapositions* in this context and are easily represented by placing a black king on every square where the king might be; we will follow this convention throughout the paper. Players may choose moves $m \in \mathbf{M}$, the set of *pseudolegal* moves for the current information set. Pseudolegal means that the move is legal according to the rules of chess in at least one state of the current information set i, though it may turn out to be illegal when tried. More specifically, we can define a referee function $r : (\mathbf{I} \times \mathbf{M} \times \mathbf{S}) \to (\mathbf{I} \times \mathbf{R} \times \mathbf{S})$, where $\mathbf{R}$ is a set of referee messages, consisting of {`victory, draw, silent, check, illegal`}. A game state appears both as an input and an output, representing the unknown real state, which only the referee can access. The referee changes our knowledge of the board and also returns a message in response to a move. The message set would have to be expanded to handle additional endings, but it is ample enough to treat KRK, as only one check type is compatible with any move. Function r obviously depends on the black king's strategy.

Since the black king is the only unknown piece, it can readily be seen that any $i \in \mathbf{I}$ contains at most 52 elements; an example of this is given in Fig. 1, (a). While 52 states might not seem like a large amount, in order to provide an optimal solution to this endgame we need to be able to handle every possible instance of $\mathbf{I}$ optimally. Counting the subsets of the information set in Fig. 1 alone, we obtain $2^{52} - 1$ possibilities. Even using mirroring along the two axes and a diagonal, there are over 400 ways to place the white king and rook, each allowing on average 40 to 50 positions for the black king, and as such 2^{40} to 2^{50} instances of $\mathbf{I}$. An approximate calculation leads to about 10^{16} or 10^{17} possible information sets, making the KRK ending in Kriegspiel not that far behind the whole game of checkers in sheer size. Bolognesi and Ciancarini [10] mention, among other things, these figures to motivate their decision to use a heuristic, approximated approach to the problem. Their problem does not fully overlap with ours, as they try to provide strategies for playing even when victory is not guaranteed; for example, in KRK, when the rook does not start near the king and cannot immediately approach it. On the other hand, this algorithm is only concerned with algorithmically won positions, although it can be extended with heuristics to treat such a case.

When considering the size of the problem, it is clear that many of these Kriegspiel information sets are really redundant as they contain no information that can be exploited by the player in order to speed up the road to victory, at least in the worst case. Figure 1, (a)-(b) shows an example wherein removing states from the information set does not make any difference. If this is true for a majority of $\mathbf{I}$, then only a fraction of the actual state space contributes to the solution and needs to be explored, making a brute force approach feasible.

We would just have to note the largest set for which a given move is optimal; this move would be optimal for its subsets, as well, except those explicitly listed in a separate entry. This reasoning makes the trivial assumption that reducing the number of states, and hence uncertainty, cannot worsen the worst-case performance of any move; we can simply ignore the additional information.

Roughly speaking, the algorithm proceeds through iterative retrograde analysis and creates a table of entries whose elements contain an information set or metaposition, an optimal move associated with it, and a maximum number of moves this strategy will take to achieve certain victory. First, we find all metapositions $i_1 \in \mathbf{I}$ such that checkmate is possible in one move. Figure 1, (c) shows one of the only examples of single-move checkmates in KRK with more than one king location. These mates are very simple to find; it suffices to search all piece configurations and the corresponding legal moves for checkmate positions according to chess rules.

After solving the problem at depth 1 as described above, we look for all $i_2 \in \mathbf{I}$ such that checkmate is possible in at most two moves no matter what the referee's response is; this includes situations always leading to subsets of states found in the first step. This procedure is repeated several times at each depth, in order to deal with illegal moves which require analysis of same-depth metapositions, and then proceeds to the next depth when the current depth is exhausted. We also discard metapositions that are subsets of previously added metapositions, since they are not optimal. When the algorithm fails to generate any new entries for the next depth, execution ends.

The algorithm is summarized in Fig. 2. The basic concept is that, if we know that, no matter the referee's response message, we are going to mate the king in at most n moves, then the present position is won in at most $n+1$ moves. If the method by which a new position is constructed is correct, then the correctness of the whole algorithm is easily proved by induction. One needs to be careful in how $(n+1)$-depth entries are constructed from the n-depth (and below) entries, because the danger of "strategy fusion", as defined by Frank and Basin [15], is always around the corner. Strategy fusion can be briefly summarized as the pitfall of having plans for dealing with each specific case successfully, but not knowing which of those cases we are in. Within the context of Kriegspiel endings, this happens if we mistakenly assume that, only because we can solve metapositions $i_1, i_2 \in \mathbf{I}$ in n moves, we can also solve $i_1 \cup i_2$ in n moves. This is not true even if the optimal strategies for i_1 and i_2 start with the same move; there is no guarantee that the same move will also solve $i_1 \cup i_2$ optimally, or that it can be solved at all. Such a pitfall is showcased in the KPK ending in [4], where victory cannot be achieved with probability 1 under certain circumstances.

Therefore, the white player needs to know, at any given time, what metaposition he is in. This is accomplished by the innermost *for* loop, where all compatible metaposition entries in the current list (that is, all entries with the correct placement of white pieces) are assigned to the possible referee messages that may result from the move being examined. In the least optimized case, all possible combinations might be tested. For example, if move Ka2-b2 can gener-

```
kriegRetrograde(entryList,depth)
begin
 added = false;
 for each placement of white pieces P do
  fill P with black kings
  for each pseudolegal move M do
   messages = possibleMessages(P,M);
   for each assignment A of entries from entryList to messages do
    reduceBlackKings(P,A);
    if (checkAndAdd(entryList,P,A)) added = true;
   od
  od
 od
 if (added) kriegRetrograde(entryList,depth);
 else if (entriesWithDepth(depth+1)) kriegRetrograde(entryList,depth+1);
 return entryList;
end
```

Fig. 2. Pseudocode listing for main retrograde function.

ate a check, a silent response and an illegal notice, the algorithm will generate $n_1^2 \cdot n_2$ assignments, where n_1 is the number of metapositions already in the database with the white pieces placed just like the current board after Ka2-b2 (squared because there are two messages to consider), and n_2 is the number of metapositions where the white pieces are set up like the current board (for an illegal outcome). For each assignment of metapositions $i_1, \ldots, i_k$ to messages $m_1, \ldots, m_k$, the algorithm determines the largest metaposition such that every message m_j will result in a subset of the corresponding metaposition i_j. Since every possible outcome has already been computed as won, this new metaposition is also won. Since all assignments are considered, optimal assignments will also be found, and sub-optimal assignments will be discarded.

Method `reduceBlackKings` generates new metapositions in a subtractive way, starting with a full board and taking away black kings as they are found to be incompatible with a message-outcome pair; this means that the algorithm decides what message m_j would be generated if the black king were there, and if the king's positions after black's next move do not entirely lie in i_j, that square has to be cleared. If the resulting metaposition is empty, it is discarded. If it is a subset of an existing entry, it is also discarded. On the other hand, if the new metaposition is original, it is added to the entry list together with the tested move and depth information. If a new entry is added, the algorithm will have to go through an additional pass; illegal moves may generate new mates of depth n instead of $n + 1$, which need to be searched again. Most depth levels in KRK require 3-5 passes, after which the next depth is considered.

Once the algorithm has finished, it returns a full list of metapositions, each having a best move and a distance to mate in the worst case. This database is used as follows. The player searches it with a metaposition representing the

current state of the game. All entries that are supersets of it are returned, and if none exist, it means that it is not possible to force a mate from here. Among these entries, the player selects the one with the shortest distance to mate; in the event of a tie, he will pick the one with the lowest number of states (black kings). He will then proceed to play the corresponding move.

3.1 Complexity and optimizations

The computational complexity of a single depth step of the algorithm is $O(b^p n^m)$, where b is board size, p is the number of white pieces on the board, n is the average size of the metaposition list, and m is the number of messages the referee can output. Estimating n is best done through the actual experiment, which shows that its increase in KRK is exponential and quite regular until about depth 30, after which there is a sharp slowdown. This algorithm can become very slow if there are many possible referee messages. This will be the main problem to overcome with the KQK ending, since the queen can check in four different ways, and most moves can result in three (as opposed to just one in KRK): file, rank, short diagonal and long diagonal.

The following optimizations are possible and have been implemented.

- Game-ending referee messages are handled implicitly. Black king positions leading to instant checkmate are automatically added to any new entry, and stalemate or drawn positions are excluded before anything else takes place. This means that two referee messages can be taken off the list, leaving only three in KRK - silent, illegal and check.
- Not every metaposition assignment is considered, but only those that will create a metaposition of the desired depth. For example, if we are filling the database with depth 10 entries, it is useless to try an assigment leading to depth 9 or less, because it will already have been considered at the appropriate step. In other words, in order to generate a depth 10 entry, we need to have at least one depth 9 entry assigned to the silent or check messages *or* another depth 10 entry linked to the illegal move message. Any assignment violating this rule is skipped.
- Not every single metaposition in the list is checked. In particular, at the end of each step, metapositions that are found to be subsets of others are "dumped" into support files, regardless of their distances to victory; the only exception is metapositions generated during the last step. This is related to the above point; when determining depth 10 entries, there will always be at least one depth 9 or depth 10 metaposition in the assignment, but as for the others it is irrelevant whether their depth is 8 or 2. One can simply take the larger one and dump the smaller. As a consequence, the final database is obtained by merging all the dump files with the collection that remains after the last iteration of the algorithm.
- A further optimization to reduce the number of metapositions being considered is to perform an intersection between each entry and the legal positions of the black king after the corresponding message. In this way, only the

relevant parts of the metaposition are considered, and duplicates can be computed only once.

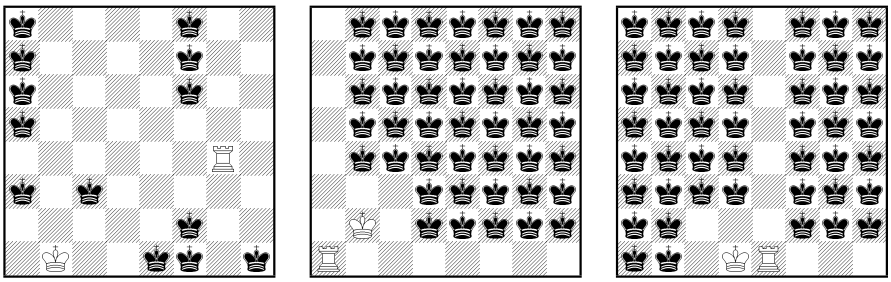
4 Solving KRK

We ran the algorithm described in the previous section in order to ascertain whether it is computationally feasible for the KRK ending, and if so, whether existing directives such as Boyce's and Magari's can be improved on. Finding a measure of the complexity for this problem (that is, how many metapositions it takes to fully describe KRK out of the theoretical 10^{16}) and the longest forced KRK mate were also goals for this computation. We implemented the algorithm in the Java programming language and executed it on a single machine. The algorithm did not feature any peculiar optimizations besides those described in the previous section, and it ran on a single processor. Its run terminated after about 12 days of uninterrupted execution.

Using mirroring on the x, y and diagonal axes, the problem of KRK in Kriegspiel is described with a tablebase of 1,087,599 metapositions, or about 10^6 from the hypothetical 10^{16}. KRK in chess is fully described with about 23,000 positions, making the equivalent Kriegspiel problem about 40-50 times as complex. Results may vary to a degree, depending on storage policies for metapositions that are subsets of other entries; in particular, the tablebase can be compressed to unify a subset with its superset if they have the same best move, but doing so loses information on the strict upper bound on the moves till victory, which has to be recalculated on the fly with more accesses to the tablebase.

The longest forced mate sequences in Kriegspiel KRK are 41 moves long, making the 50-move rule irrelevant in this endgame, and there are only two. One of them is shown in Fig. 3, (a). The other mate in 41 is almost identical, except that the rook is in h4 instead of g4; the solution is the same. It is readily seen that such a position cannot occur during a normal game. The same can be said for a majority of entries in the tablebase, especially at the later depth levels. After depth 35, most entries resemble man-made puzzles and problems rather than situations that a player is likely to encounter. In this particular problem, the player must maneuver his pieces around the enemy kings until he safely brings the rook next to the king. In the optimal solution, it takes nine moves to accomplish this: Rf4, Kc2, Rf8, Kd3, Rg8, Rh8, Rh1, Rd1, Kc2.

Boards (b) and (c) represent situations that are much more likely to happen in a real game. In particular, board (b) is the starting position for Boyce's algorithm given in [9], and board (c) is the starting position for Magari's algorithm, from [7]. Boyce's directives are based on trapping the king in a single quadrant of the board with the rook and then using the king to push back the opponent. Magari's method consists of starting from (c) and isolating the king on one side of the board by playing Kd2, Re2, Kd3, Re3, and so on, scanning the board until a check reveals the location of the enemy king. The tablebase shows that Boyce's method is a better approximation of the shortest mate. Boyce's posi-



(a) (b) (c)

Fig. 3. (a) Longest forced mate in KRK, mate in 41; (b) Boyce's starting position, mate in 30; (c) Magari's starting position, mate in 34.

tion is a mate in 30, four moves shorter than Magari's position. There are only two positions completely filled with enemy kings that can be won faster, in 29 moves, and both have the rook in the a1 corner, just like (b); the king is in c2 and c3, respectively. Thus, Boyce has a good understanding of a convenient starting position. Moreover, optimal play from (c) does not follow Magari's algorithm. Instead, the player immediately runs to the nearest corner, reaching a Boyce-like position through, for example, Ke2, Kf2, Rg1, Kg3, Rh1 - from here White mates in 29, as mentioned. A full run of the algorithm still reminds of Boyce's directives, though it is noticeably more quirky and difficult to summarize.

5 Conclusions and future work

We have established an algorithm that will always win the KRK endgame if victory can be achieved with probability 1, and instantly provide a strict upper bound on the number of moves until checkmate. This approach can be extended to other Kriegspiel endgames such as KQK, KPK and others. Currently, the KRK tablebase occupies about 80 megabytes of hard disk space, and a different structure might be needed in order to deploy it into Kriegspiel-playing programs, as looking up the best move in a given situation is not as straightforward as in chess: on average, the program has to examine 25,000 metapositions and find the compatible candidate with the shortest route to mate. Extension to queen endgames will also probably require more optimizations, as preliminary tests have shown an almost tenfold decrease in speed due to the exponential nature of the search algorithm.

Another area of improvement would involve choosing among equivalent lines of play (in terms of worst-case performance) by reasoning and making assumptions on the opponent. This feature could be helpful in shortening mate sequences if Black does not play near-optimal moves. It is to be noted that playing almost like an oracle is not too difficult in this endgame; a king trying to maintain posi-

tion by moving back and forth near the center of the chessboard is often playing the best strategy, or close to the best.

References

1. Zermelo, E.: Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In: Proceedings of the Fifth International Congress of Mathematicians. Volume 2., Cambridge, UK (1913) 501–4
2. Bellman, R.: On the application of dynamic programing to the determination of optimal play in chess and checkers. Proceedings of the National Academy of Sciences **53**(2) (1965) 244–247
3. Stiller, L.: Some Results from a Massively Parallel Retrograde Analysis. ICCA Journal **14**(3) (1991) 129–134
4. Ciancarini, P., DallaLibera, F., Maran, F.: Decision making under uncertainty: a rational approach to Kriegspiel. In van den Herik, J., Uiterwijk, J., eds.: Advances in Computer Chess 8, Univ. of Rulimburg (1997) 277–298
5. Ferguson, T.: Mate with bishop and knight in Kriegspiel. Theoretical Computer Science **96** (1992) 389–403
6. Ferguson, T.: Mate with two bishops in Kriegspiel. Technical report, UCLA (1995)
7. Magari, R.: Scacchi e probabilità. In: Atti del Convegno: Matematica e scacchi. L'uso del Gioco degli Scacchi nella didattica della Matematica, Forlì, Italy (1992) 59–66
8. Sakuta, M.: Deterministic Solving of Problems with Uncertainty. PhD thesis, Shizuoka University, Japan (2001)
9. Boyce, J.: A Kriegspiel endgame. In Klarner, D., ed.: The Mathematical Gardner. Prindle, Weber & Smith (1981) 28–36
10. Bolognesi, A., Ciancarini, P.: Searching over metapositions in Kriegspiel. In van den Herik, J., Björnsson, Y., Netanyahu, N., eds.: Computer and Games 04. Volume 3846 of Lecture Notes in Computer Science., Ramat-Gan, Israel, Springer (2004) 246–261
11. Russell, S., Wolfe, J.: Efficient belief-state AND-OR search, with application to Kriegspiel. In: Int. Joint Conf. on Artificial Intelligence (IJCAI05), Edinburgh, Scotland (2005) 278–285
12. Nance, M., Vogel, A., Amir, E.: Reasoning about partially observed actions. In: Proceedings of 21st National Conference on Artificial Intelligence (AAAI '06), Boston, USA (2006) 888–893
13. Ciancarini, P., Favini, G.: Representing Kriegspiel states with metapositions. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07), Hyderabad, India (2007) 2450–2455
14. Parker, A., Nau, D., Subrahmanian, V.: Game-tree search with combinatorially large belief states. In: Int. Joint Conf. on Artificial Intelligence (IJCAI05), Edinburgh, Scotland (2005) 254–259
15. Frank, I., Basin, D.: A theoretical and empirical investigation of search in imperfect information games. Theoretical Computer Science **252** (2001) 217–256