Material Symmetry to Partition Endgame Tables

Abdallah Saffidine[†], Nicolas Jouandeau[‡], Cédric Buron[§†], and Tristan Cazenave[†]

[†]Lamsade, Université Paris-Dauphine, France,
[‡]Liasd, Université Paris 8, France,
[§]National Chiao Tung University, Taiwan

Abstract. Many games display some kind of *material symmetry*. That is, some sets of game elements can be exchanged for another set of game elements, so that the resulting position will be equivalent to the original one, no matter how the elements were arranged on the board. Material symmetry is routinely used in card game engines when they normalize their internal representation of the cards.
Other games such as chinese dark chess also feature some form of material symmetry, but it is much less clear what the normal form of a position should be. We propose a principled approach to detect material symmetry. Our approach is generic and is based on solving multiple relatively small sub-graph isomorphism problems. We show how it can be applied to chinese dark chess, dominoes, and skat.
In the latter case, the mappings we obtain are equivalent to the ones resulting from the standard normalization process. In the two former cases, we show that the material symmetry allows for impressive savings in memory requirements when building endgame tables. We also show that those savings are relatively independent of the representation of the tables.

1 Introduction

Retrograde analysis is a tool to build omniscient endgame databases. It has been used in chess to build endgame databases of up to six pieces [19, 20, 14]. It has also been used in a similar way to build endgame databases for checkers [17] that have helped solving the game [18]. It has also been successfully used for strongly solving the game of awari with a parallel implementation that evaluated all the possible positions of the game [15]. Other games it has helped solving are fanorona [16] and nine men's morris [8]. Retrograde analysis has also been applied to chinese chess [6], kriegspiel [4], and go [2].

An important limitation on the use of endgame tables is the size needed to store all the computed results. Consequently, elaborate compression techniques have been proposed and they have been instrumental in achieving some of the aforementioned results [17]. An orthogonal approach to alleviate the memory bottleneck is to use symmetries to avoid storing results that can be deduced from already stored results by a symmetry argument. There are two kinds of such symmetry arguments. *Geometrical symmetry* is the simplest case [5]: for

instance if a CHESS position p does not feature any pawn and castling rights have been lost then any combination of the following operations yields a position equivalent to p. Flipping pieces along a vertical axis between the 4th and the 5th columns, flipping pieces along a horizontal axis between the 4th and the 5th rows, or rotating the board. *Material symmetry* is typical from card games, in particular when suits play an equivalent role. In most trick-taking games, for instance, if a position p only contains Hearts, any position obtained by replacing all cards with cards of same rank from an other suit is equivalent.

State-of-the-art engines for card games such as SKAT or BRIDGE already use material symmetry detection in their transposition and endgame tables. However, recognising the most general form of material symmetry is not as straightforward in some other games, notably CHINESE DARK CHESS. In this paper, we propose a principled framework to detect material symmetry and show how it can be applied to three games: CHINESE DARK CHESS, DOMINOES, and SKAT. At the core of our method lies the sub-graph isomorphism problem which has been extensively studied in computer science [21, 12].

CHINESE DARK CHESS is a popular game in Asia [3]. One of the key feature of endgame databases in CHINESE DARK CHESS is that some combinations of pieces are equivalent. It means that in some endgames, a piece can be replaced by another piece without changing the result of the endgame. Therefore an endgame computed with the first piece can be used as is for the endgame positions containing the other piece in place of the first piece. This property reduces the number of endgame databases that have to be computed. Using the relative ordering of pieces instead of the exact values is similar to partition search in BRIDGE that store the relative ordering of cards in the transposition table instead of the exact values [9].

DOMINOES is also a popular game and in the endgame some of the values of pieces can be replaced by other values without changing the outcome of the game. We use this property to compute a reduced number of endgame tables for the perfect information version of the game.

SKAT is a popular game in Germany. Here again in the endgame, some cards can be replaced with other cards. We present in this paper the memory reduction that can be expected from using this property to compute complete information endgame tables. Perfect information endgame tables are important in SKAT since the Monte Carlo approach in Skat consists in solving many perfect information versions of the current hand [11, 1, 13]. Using endgame tables enables to speedup the solver.

The second section describes CHINESE DARK CHESS, DOMINOES, and SKAT. We then show in Section 3 how material interaction graphs can be constructed for these domains. Section 4 recalls the principles of endgame table construction. The fifth section gives experimental results, and the last section concludes.

2 Domains addressed

2.1 Chinese dark chess

CHINESE DARK CHESS is a stochastic perfect information game that involves two players on 4×8 rectangular board. Each player starts with one king, two guards, two bishop, two knights, two rooks, two cannons and five pawns that are the same pieces as in CHINESE CHESS. The pieces can be denoted with Chinese characters or numbers as shown in Table 1. Pieces can move vertically and horizontally from one square to an adjacent free square. Captures are done on vertical and horizontal adjacent squares except for cannons that capture pieces by jumping over another piece. Such jump is done over a piece (called the *jumping piece*) and on a piece (called the *target piece*). Free spaces can stand between its initial position and the jumping piece and between the jumping piece and the target position. A piece can capture another piece according to the possibilities mentioned in Table 1. Considering numbers representation, pieces can capture lower or equal numbers (except the king that cannot capture any pawn, and except the cannon that can capture any piece).

Table 1 summarizes alternative pieces' names, each side's icons and possible captures. Column titled "Capture rule" lists the pieces that a piece can capture.

Table 1: Pieces representation and capture rules in CHINESE DARK CHESS.

Names	Representation	Capture rule	Note
King (K)	帥 將, 7 7	all opponent piece except pawns	
Guard (G)	仕 士, 6 6	all inferior opponent pieces	
Bishop (B)	相 象, 5 5	all inferior opponent pieces	
Knight (N)	馬 馬, 4 4	all inferior opponent pieces	
Rook (R)	車 車, 3 3	all inferior opponent pieces	
Cannon (C)	炮 炮, 2 2	all opponent pieces	jump capture
Pawn (P)	兵 卒, 1 1	opponent king and pawns	

At the beginning, pieces are randomly placed on the board, facing down. Typical positions thus allow 3 types of moves, flipping an unknown piece, moving a piece to an adjacent free square, capturing an opponent piece.

If a player runs out of legal moves, possibly because all their pieces have been captured, the game ends and that player loses the game. The game ends with a draw if no capture is made for 40 plies.

Fig. 1 shows the resulting board situation after the following 10 turns (Parentheses indicate revealed piece for flipping moves. Moving and capturing moves indicate two coordinates. Unknown pieces are represented with white circles.):
`b5(k) d8(P); d7(R) d4(p); c7(G) c6(c); c4(C) c4-c6; c5(p) c3(B); b6(r) c3-c4; b6-c6 b7-b6; c2(C) c4-d4; b3(g) d4-c4; b3-c3 c4-d4;`
It is now first player's turn to play. First player is white. First player possible moves are: `b5-b6; c3-b3; c3-c2; c3-d3; c5-c4`. Second player possible moves are : `c2-c5; c6-c5; c6-b6; c6-c7; d4-c4; d7-c7`.

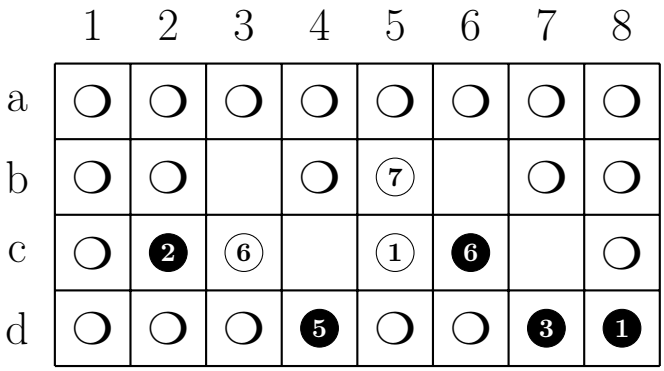


Fig. 1: Sample CHINESE DARK CHESS position after 10 turns of play.

Sometimes, the outcome of the game depends only on player's turn, even if a player has stronger piece, as shown in Fig. 2. If it is black to play, then white wins (for example: `b2-b1 c3-c2; b1-a1 c2-b2; a1-a2 b2-a2`; white wins). If it is white to play, then it is a draw game (for example : `c3-c2 b2-b3; c2-b2 b3-c3; b2-b3 c3-c2; b3-c3 c2-b2...` until draw).

2.2 The game of Dominoes

The game of DOMINOES is an ancient game, played with rectangular tiles called dominoes. We do not know precisely when or where the game appeared, but it is widely played all around the world.

The tiles of the game are rectangular and are divided into two sides. Each side represents a number of points between 0 (empty) and 6, as shown in Fig. 3a. They are called *dominoes* or simply tiles.

In this article, the tiles will be represented by the two numbers that are written on it. For instance the domino in Fig. 3a is represented by $(1, 5)$. The tile (i, j) is the same as the tile (j, i) and only appears once in the set. Thus, there are 28 tiles: $(0, 0)$, $(0, 1)$, ..., $(0, 6)$, $(1, 1)$, $(1, 2)$, ..., $(6, 6)$.

The board of the game of DOMINOES consists of a chain of dominoes already laid down. At the beginning of the game, the first player puts his "strongest"

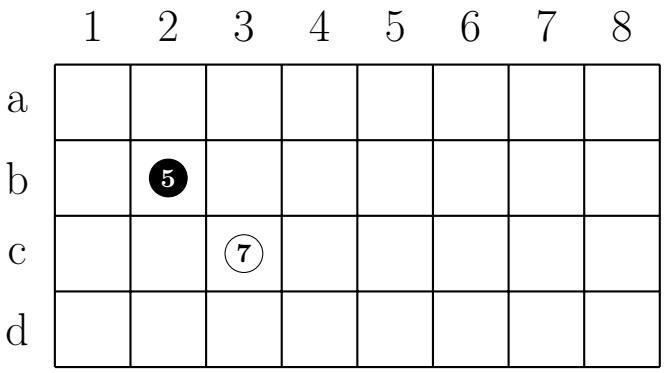
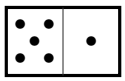


Fig. 2: Sample CHINESE DARK CHESS endgame position where the outcome depends on who's next turn to play.

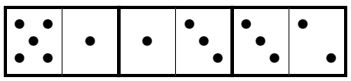
domino (the one which has the most points) on the table and starts the chain. Then, the players take turn laying down one tile they possess according to the following matching constraint. To play a tile, it must have at least one side with the same number of points of one of the tiles at the end of the chain on the table (Fig. 3). If a player cannot play a tile, they draw a tile from the stock or pass when there are no stock tiles remaining.

If a player has laid down all their tiles, or if the stock is empty and no player can play, the game ends and the player with fewer points in hand wins.

While there are multi-player variants of the game of DOMINOES, we focus in our experiments on the two-player case. Additionally, since we focus on the endgames, we assume that the stock is empty and, as a consequence, that the game is perfect information. Our method to detect material symmetry extends easily to other settings.



(a) (1, 5) domino.



(b) Chain of three tiles after a player has played (3, 2).

Fig. 3: A tile and a chain of matching tiles in DOMINOES.

2.3 Skat

Skat is a card game for three players. Skat is played with 32 cards using 8 ranks (7–10, Jack, Queen, King, and Ace) and 4 suits (Club ♣, Spade ♠, Heart ♡, and Diamond ♢). Each card is associated to a suit and a rank, for instance we have

the Jack of Heart (♡J) or the Ten of Spade (♠T). A game of skat is divided into a bidding and a playing phase. Before bidding, each player receives 10 cards and the two remaining ones are placed on the center, and called the *Skat*.

Bidding The bidding system is quite complex and can lead to three different types of game: *suit game*, *grand game*, or *null game*.[1] In each type, the player who made the highest bid gets the skat and plays against the two other ones.

In the *suit game*, a suit is chosen as trump. The order of cards is so: ♣J, ♠J, ♡J, ♢J, then the trump suit: A, T, K, Q, 9, 8, 7, then each of the remaining suits, in the same order.

In the *grand game*, there is no trump suit, but the Jacks are still considered as trumps. The order for the other cards are the same as in the suit game.

In the *null game*, there is no trump. The order of the cards is not the same as in the other games: A, K, Q, J, T, 9, 8, 7.

Playing The playing part of the game is a succession of tricks. Each trick, the player who won the previous trick plays first, and decides which suit is played in the trick. The other players have to play this suit. If a player cannot play the suit the first player played, he must play a trump. If he has neither the first suit played nor trump, he can play any other suit.

If there is a trump in the trick, the highest trump wins the trick. If there is not, the highest card of the suit the first player played wins. The player who won the trick put the cards of the trick behind him and begins the next one.

Counting the points When all the cards have been played, each team count their points. The team which got most of the points wins the round. The points are distributed as follows. A: 11 points, T: 10 points, K: 4 points, Q: 3 points, J: 2 points, 9,8, and 7: 0 points

3 Detecting material symmetry

Multiple games feature classes of endgames that are globally equivalent one to another. For instance, in a trick-taking card game such as BRIDGE, the class of 8 cards endgames where all cards are Hearts is equivalent to the class of 8 cards endgames where all cards are Spades, and both are equivalent to that where all cards are Clubs and to that with Diamonds. Indeed, we can exhibit a mapping from one class C to an equivalent class C_0 that will associate to each position $p \in C$ a position $p_0 \in C_0$ such that the score and optimal strategy in p can easily be deduced from the score and optimal strategy in p_0.

As a result, it is only necessary to build one endgame database for each equivalence class of endgames, rather than one for each class of endgames. A

[1] For more details, we refer to the rules from the International Skat Players Association: `http://www.ispaworld.org/downloads/ISkO-rules-2007-Canada.pdf`.

first approach to avoid building unnecessary endgame databases is to proceed to some form of normalization. However, the normalization process to be adopted is not always trivial. We propose here a principled method to find representatives of the equivalence classes.

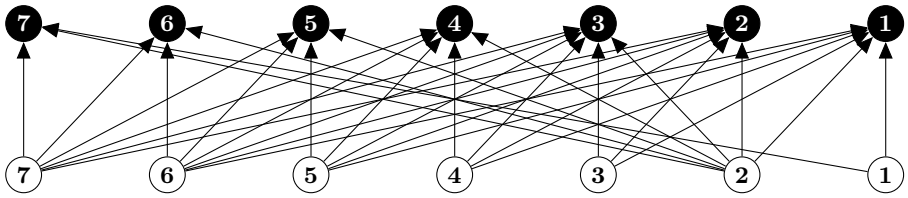


Fig. 4: Capture relationship in CHINESE DARK CHESS. The allowed captures for black were not represented so as to avoid cluttering the graph.

The basic idea is to represent interactions between pieces in a graph. The interaction can be specific to a game. For example in CHINESE DARK CHESS the interaction is a capture whereas it is a connection in DOMINOES. Fig. 4 gives the graph for the pieces of CHINESE DARK CHESS. There is an arrow from one piece to another if the first piece can capture the second one. Fig. 6 gives the graph for DOMINOES and Fig. 5 gives the graph for SKAT assuming a suit game with Clubs as trump.

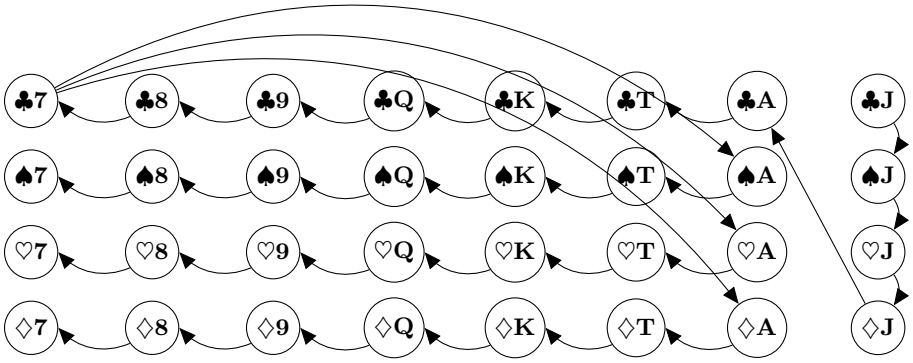


Fig. 5: Trick winning relationship for cards in SKAT assuming Clubs (♣) are trumps. To avoid cluttering, not all edges are drawn. The full graph is obtained by taking the transitive closure of the graph represented.

Once the graph has been built for a game, it can be used to detect equivalent endgames. The principle is to extract the sub-graph of the pieces of a

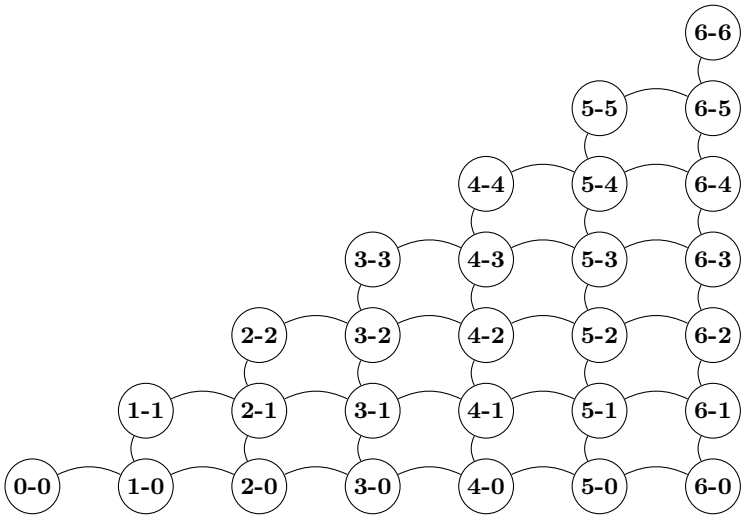


Fig. 6: Playability relationship for tiles in DOMINOES. To avoid cluttering, not all edges are drawn. The full graph is obtained by adding edges between every tile in a given column as well as between every tile in a given row.

first endgame and to compare it with the sub-graph of the second endgame. The comparison consists in finding if the two sub-graphs are isomorphic. Sub-graph isomorphism is a hard problem, but for the games that we address a naive algorithm is fast enough to compare sub-graphs of interest [21].

4 Endgame Tables

Retrograde analysis is completed in two steps. The first step consists in enumerating all winning positions for one player. The second step consists in repeatedly finding the new winning positions two moves away from existing ones. When no new position is found the algorithm stops and the endgame table has been computed. For each combination of pieces a different endgame table is computed.

There are multiple ways to generate the new winning positions two moves away from already computed winning positions. A naive approach consists in running through all possible positions and checking if they are two moves away from already computed ones. A more elaborate approach consists in doing unmoves from already computed winning positions to find a restricted set of candidate new winning positions.

5 Experiments

In this section we give experimental results for CHINESE DARK CHESS and DOMINOES. We also experimented with SKAT but only for small numbers of cards

because our generic implementation using sub-graph isomorphism is slower than existing domain specific implementations, so these results are not included.

The hash code of a position has to be computed in two steps. First, obtain the correspondence between the pieces and their representatives, then combine the representative pieces with their locations into a hash code. In our implementation, the sub-graph isomorphism part is not optimized. However it is possible to pre-compute all sub-graph isomorphims between tables and to have a fast correspondence between a table and its representative.

5.1 Equivalence classes

Assume that the endgame table is split in multiple tables depending on the number of game elements (pieces/tiles/cards). Assume also that the resulting tables are split further according to the sets of elements that constitute the positions. Such a decomposition is natural and makes distributing the workload easier as there is no dependency between different sets of elements if they have the same number of elements [10]. Thus, each set of game elements has a corresponding endgame table which can be stored in its own file.

We begin by comparing the number of possible files without using the reduction and the number of remaining files with our method. Material symmetry allows us to reduce significantly the number of needed files. For instance, for CHINESE DARK CHESS with 4 elements, there are 1737 sets of elements that are consistent with the rules of the game. However many such sets are equivalent, and if we detect material symmetry, we can reduce the number of needed sets down to 186, which represent a reduction factor of 9.34. Table 2 shows the number of sets, the number of sets needed when material symmetry is used, and the resulting reduction factor for a given number of game elements in CHINESE DARK CHESS and DOMINOES.

Table 2: Total number of tables and number of representatives.

Game	Elements	Total	Representatives	Reduction factor
CHINESE DARK CHESS	2	49	8	6.13
	3	378	46	8.22
	4	1737	186	9.34
	5	5946	672	8.92
	6	16,524	2240	7.38
	7	39,022	6694	5.83
	8	80,551	17,662	4.56
DOMINOES	3	4032	20	201.6
	4	30,303	61	496.8
	5	180,180	185	973.9
	6	868,140	563	1542.0

5.2 Perfect hashing representation

The reduction factors presented in Table 2 do not necessarily match with the memory actually needed to store the endgame tables. Indeed, different sets of pieces might lead to tables of different size and equivalence classes might have different number of elements. For instance, in CHINESE DARK CHESS the Kkm file corresponds to 29,760 positions, while the kPP file corresponds only to 14,880 positions because the two pawns are indistinguishable. In this section, we assume that we store the result for every position in the corresponding file using a perfect hashing function, or index.

We did not use any advanced compression mechanism [17]. For instance, CHINESE DARK CHESS has 2 geometrical symmetries and we used 2 bits to encode whether a position was won, lost, or draw. Therefore, for each file we need half as many bits as the number of positions.

Table 3 shows the size needed for endgame tables with various number of elements using a perfect hashing representation. In this table, we added the different possible positions of each file for a given number of elements with and without the reduction, and then calculated the real reduction factor. Notice that the reduction factor in Table 3 are close to those in Table 2 but are not always the same.

Table 3: Size of the endgame database with a perfect hashing representation.

Game	Elements	Total		Representatives		Reduction
		Positions	Memory	Positions	Memory	factor
CHINESE DARK CHESS	2	4.961×10^{4}	2.97 KB	7.936×10^{3}	496 B	6.13
	3	9.999×10^{6}	610.3 KB	1.131×10^{6}	69.02 KB	8.84
	4	1.140×10^{9}	67.92 MB	1.142×10^{8}	6.81 MB	9.92
	5	9.036×10^{10}	5.26 GB	1.013×10^{10}	603.8 MB	8.92
	6	5.440×10^{12}	316.7 GB	8.002×10^{11}	46.58 GB	6.80
	7	2.601×10^{14}	14.78 TB	5.247×10^{13}	2.98 TB	4.96
	8	1.014×10^{16}	576.1 TB	2.756×10^{15}	156.7 TB	3.68
DOMINOES	3	21,168	2.58 KB	92	11.50 B	230.09
	4	550,368	67.18 KB	996	124.50 B	552.58
	5	8,026,200	979.76 KB	7,854	981.75 B	1021.93
	6	82,556,550	9.84 MB	53,790	6.57 KB	1541.99

5.3 Won positions

The following alternative representation for endgame tables can be used, we will show that reduction factors are similar under this representation as well. For a given set of pieces, store the list of won positions. We can detect that a position is lost if the position with reversed colors is in the table. Finally, a position is

a draw if it and the reverse position are not in table. Again, we can split the endgame table according to the set of pieces that constitute it and use one file for each possible set.

We have computed endgames tables with only won positions. Table 4 gives the total number of won positions that would need to be stored for a given number of game elements. It also displays the number of positions that need to be stored if we use material symmetry and only store one representative file for each equivalence class. The last column indicate the savings in terms of number of stored positions when material symmetry is taken into account.

Table 4: Size of the endgame database when only storing won positions.

Game	Elements	Total	Representatives	Reduction factor
CHINESE	2	6448	744	8.67
DARK	3	1,650,763	171,516	9.62
CHESS	4	156,204,805	15,418,377	10.13
	3	22,785	104	219.09
DOMINOES	4	409,234	747	547.84
	5	5,706,631	5,741	994.01

6 Conclusion

We have presented a general method based on sub-graph isomorphism that allows detection material symmetries. We have shown that it could be applied to SKAT, DOMINOES, and CHINESE DARK CHESS. Detecting material symmetry makes it easier to build larger endgame databases. While finding a representative for positions that are equivalent under material symmetry (that is, defining a normal form) for SKAT and DOMINOES is relatively easy, the CHINESE DARK CHESS case is more intricate as the relationship between the different pieces is more elaborate. Our approach solves this problem in CHINESE DARK CHESS and allows a reduction factor between 5 and 10 in the size needed for storing the endgame tables.

Note that although we do not report any time measurement, a significant reduction factor can also be expected for the time needed to compute the endgame tables. In our framework, we propose to store a file mapping sets of pieces to their representative. This file is much smaller than the corresponding endgame tables. Thus, the only runtime cost induced by our method is an additional indirection when looking up a position in the database.

A generalisation of transposition tables has been proposed by Furtak and Buro in order to regroup cases that have similar structures but different payoffs [7]. Determining to which extent the proposed method for detecting material symmetry can be extended to account for payoff similarity seems to be a promising line for future work.

References

1. Michael Buro, Jeffrey R Long, Timothy Furtak, and Nathan Sturtevant. Improving state evaluation, inference, and search in trick-based card games. In *21st International Joint Conference on Artificial Intelligence (IJCAI2009)*, 2009.
2. Tristan Cazenave. Generation of patterns with external conditions for the game of Go. *Advances in Computer Games*, 9:275–293, 2001.
3. Bo-Nian Chen, Bing-Jie Shen, and Tsan-sheng Hsu. Chinese dark chess. *ICGA Journal*, 33(2):93, 2010.
4. Paolo Ciancarini and Gian Favini. Solving kriegspiel endings with brute force: the case of KR vs. K. *Advances in Computer Games*, pages 136–145, 2010.
5. Joseph C Culberson and Jonathan Schaeffer. Pattern databases. *Computational Intelligence*, 14(3):318–334, 1998.
6. Haw-ren Fang, Tsan-sheng Hsu, and Shun-chin Hsu. Construction of chinese chess endgame databases by retrograde analysis. *Computers and Games*, pages 96–114, 2001.
7. Timothy Furtak and Michael Buro. Using payoff-similarity to speed up search. In *22nd international joint conference on Artificial Intelligence (IJCAI2011)*, pages 534–539. AAAI Press, 2011.
8. Ralph Gasser. Solving nine men's morris. *Computational Intelligence*, 12(1):24–41, 1996.
9. Matthew L Ginsberg. Partition search. In *National Conference On Artificial Intelligence (AAAI1996)*, pages 228–233, 1996.
10. Mark Goldenberg, Paul Lu, and Jonathan Schaeffer. Trellisdag: A system for structured dag scheduling. In *Job Scheduling Strategies for Parallel Processing*, pages 21–43. Springer, 2003.
11. Sebastian Kupferschmid and Malte Helmert. A skat player based on Monte-Carlo simulation. *Computers and Games*, pages 135–147, 2007.
12. Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 313–320. IEEE, 2001.
13. Jeffrey Richard Long. *Search, Inference and Opponent Modelling in an Expert-Caliber Skat Player*. PhD thesis, University of Alberta, 2011.
14. Eugene V. Nalimov, Guy McCrossan Haworth, and Ernst A. Heinz. Space-efficient indexing of chess endgame tables. *ICGA Journal*, 23(3):148–162, 2000.
15. John W Romein and Henri E Bal. Solving awari with parallel retrograde analysis. *Computer*, 36(10):26–33, 2003.
16. Maarten PD Schadd, Mark HM Winands, Jos WHM Uiterwijk, H Jaap Van Den Herik, and Maurice HJ Bergsma. Best play in fanorona leads to draw. *New Mathematics and Natural Computation*, 4(03):369–387, 2008.
17. Jonathan Schaeffer, Yngvi Björnsson, Neil Burch, Robert Lake, Paul Lu, and Steve Sutphen. Building the checkers 10-piece endgame databases. In *Advances in Computer Games 10*, pages 193–210. 2003.
18. Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007.
19. Ken Thompson. Retrograde analysis of certain endgames. *ICCA Journal*, 9(3):131–139, 1986.
20. Ken Thompson. 6-piece endgames. *ICCA Journal*, 19(4):215–226, 1996.
21. Jeffrey D. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.