ENDGAME ANALYSIS OF DOU SHOU QI

Jan N. van Rijn[1] *Jonathan K. Vis*[2]

Leiden, The Netherlands

ABSTRACT

Dou Shou Qi is a game in which two players control a number of pieces, each of them aiming to move one of their pieces onto a given square. We implemented an engine for analyzing the game. Moreover, we created a series of endgame tablebases containing all configurations with up to four pieces. These tablebases are the first steps towards theoretically solving the game. Finally, we constructed decision trees based on the endgame tablebases. In this note we report on some interesting patterns.

1. INTRODUCTION

Dou Shou Qi (meaning: "Game of Fighting Animals") is a Chinese board game first described by Pritchard and Beasley (2007). In the Western world it is often called Jungle, The Jungle Game, Jungle Chess, or Animal Chess. Dou Shou Qi is a two-player abstract strategy game. It contains some elements from Chess and Stratego as well as some other chess-like Chinese games (e.g., Banqi). Its origins are not entirely clear, but it seems that it evolved rather recently (around the 1900s) in China. Dou Shou Qi is played on a rectangular board consisting of 9×7 squares (see Figure 1). The columns are called *files* and are labelled *a*–*g* from left to right. The rows or *ranks* are numbered 1–9 from bottom to top (the board is viewed from the position of the white player).

There are four different kinds of squares: den, trap, water, and land. There are two *dens* (D) located in the center of the first and the last rank ($d1$ and $d9$). Each den is surrounded by *traps* (T). There are also two rectangular (3×2 squares) bodies of *water* (W) sometimes called *rivers*. The remaining squares are ordinary land squares. Each player has eight different pieces representing different animals (see below). Each animal has a certain *strength*, according to which they can *capture* other (opponent's) pieces. Only pieces with the same or a higher strength may capture an opponent's piece. The only exception to this rule regards the weakest (rat) and the strongest (elephant) pieces. Just like the spy in Stratego, the weakest piece may capture the strongest. The strength of the pieces, from weak to strong, is: 1 `R`, `r` — Rat (sometimes called mouse); 2 `C`, `c` — Cat; 3 `W`, `w` — Wolf (sometimes called fox); 4 `D`, `d` — Dog; 5 `P`, `p` — Panther (sometimes called leopard); 6 `T`, `t` — Tiger; 7 `L`, `l` — Lion; 8 `E`, `e` — Elephant. The initial placement of the pieces is fixed, see Figure 1. The capital letters are used to denote the white pieces. Players alternate moves with White moving first. Each turn one piece must be moved. Each piece can move one square

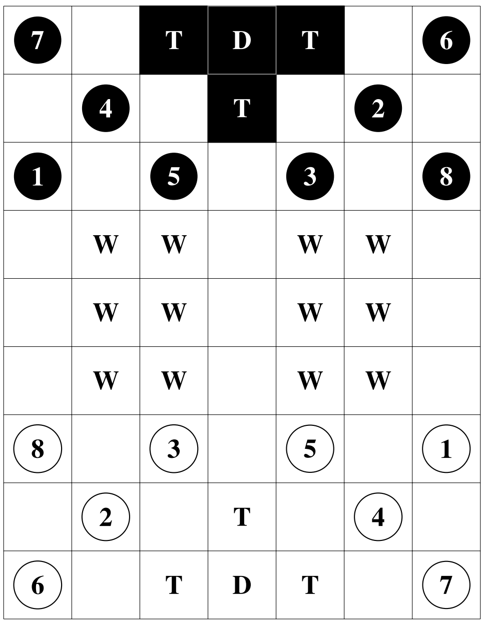


Figure 1: Dou Shou Qi game board.

[1]Leiden Institute of Advanced Computer Science, Universiteit Leiden, The Netherlands. E-mail: j.n.van.rijn@liacs.leidenuniv.nl
[2]Leiden Institute of Advanced Computer Science, Universiteit Leiden, The Netherlands. E-mail: j.k.vis@liacs.leidenuniv.nl

either horizontally or vertically. In principle, a piece may not move into the water, and it is also forbidden to enter its own den ($d1$ for White, and $d9$ for Black). The rat is the only piece that can swim, and is therefore able to enter the water. It may also capture in the water (the opponent's rat). However, it may not capture the elephant from the water. Lions and tigers are able to leap over water (either horizontally or vertically). They cannot jump over the water when a rat (own or opponent's) is on any of the intermediate water squares. When a piece is in an opponent's trap ($c9$, $d8$, $e9$ for White and $c1$, $d2$, $e1$ for Black), its strength is effectively reduced to zero, meaning that any of the opponent's pieces may capture it regardless of its strength. A piece in one of its own traps is unaffected. The objective of the game is either to place one of the pieces in the opponent's den or to eliminate all of the opponent's pieces. As in Chess, stalemate positions are declared a draw. A threefold repetition rule is imposed in some variants of this game. However, the existence of such a rule is irrelevant for our analyses.

The game Dou Shou Qi is not extensively studied in the literature. Burnett (2010) attempts to characterize certain local properties of subproblems that occur. These so-called loosely coupled subproblems can be analyzed separately in contrast to analyzing the problem as a whole resulting in a possible speed-up in the overall analysis. The author also proposes an evaluation (utility) function for Dou Shou Qi, which we will use in our research as well. We will also present an engine without taking the loosely coupled subproblems into account. The game has been proven PSPACE-hard by van Rijn (2012) and Hoogeboom *et al.* (2014).

2. DOU SHOU QI ENGINE

To get a feeling for the search complexity of a Dou Shou Qi game, we present some numbers. An average configuration allows for ≈ 20 legal moves (out of a maximum of 32). In theory a complete game tree of all possible games from the initial configuration can be constructed by recursively applying the rules of the game. From the initial configuration the number of leaves visited per ply can be found in Table 1. Assuming an average game length of 40 moves (80 plies), there are approximately 20^{80} possible games.

In this section we introduce a Dou Shou Qi (analyzing) engine[3], similar to a Chess engine which is used to search through the game tree given a certain configuration. The seminal 1950 paper by Shannon and Hsu (1950) lists the elements of a chess playing computer (also known as an engine), which are also applicable to a Dou Shou Qi engine as both games are similar. Usually, an engine consists of three parts: a move generator, which generates a set of legal moves given a configuration; an evaluation function (or utility function), which is able to assign a value to the leaf of the game tree, and a search algorithm to traverse the game tree. As evaluation function we use the method constructed by Burnett (2010).

Table 1: The number of leaf nodes that are evaluated by the minimax method (no pruning) at a certain depth from the initial configuration. The performance was measured on an Intel i7-2600 with 16 GB RAM.

ply	time	number of leaf nodes
1	0.00	24
2	0.00	576
3	0.00	12,240
4	0.06	260,100
5	1.26	5,098,477
6	23.46	99,860,517
7	7:51.33	1,890,415,534

In most chess-playing engines today some form of the minimax algorithm is used. These engines try to minimize the possible loss for a worst case (maximum loss) scenario. As is well-known, the performance of the minimax algorithm can be improved by the use of alpha-beta pruning. Here, a branch is not further evaluated when at least one of the immediately following configurations proves to be worse (in terms of the evaluation function) than a previously examined move, cf. Knuth and Moore (1975). In our case, using the same machine, we are able to search the game tree within the eight minutes to a depth of 14 plies (instead of 7 plies by the minimax method). We remark that the aforementioned search methods operate on trees, while the actual search space is an acyclic graph. Configurations that have been considered previously might be considered again by means of so-called transpositions. A reordered sequence of the same set of moves results in the same configuration. This is especially true for end game configurations where a few pieces can move around in many ways to form equal

[3] For the implementation of the engine and the retrograde analysis see: `http://www.liacs.nl/home/jvis/doushouqi`.

configurations. By storing evaluated configurations in memory we can omit expensive re-searches of the same configuration. Commonly, a hash table is used. Zobrist (1970) introduced a hashing method for chess which can easily be extended to a more general case.

Our engine consists of the alpha-beta algorithm augmented with a (large) transposition table using the Zobrist hashing method. This engine has proven its usefulness. For instance, we constructed endgame tablebases (see Section 3), which in turn can be used to improve the engine.

3. ENDGAME TABLEBASE CONSTRUCTION

An endgame tablebase describes for every configuration with a certain number of pieces, relevant precalculated information. For this game it states which side has a theoretical win or that the game is drawn. Endgame tablebases exist for many well-known games, most notably for Chess by Thompson (1986) and Checkers by Schaeffer *et al.* (2004). The technique we use for constructing this database is retrograde analysis, similar to the technique used by Thompson (1986).

The resulting endgame tablebase contains for each configuration up to four pieces the game theoretic value, the amount of moves until the theoretic value has been achieved, and the first move (best move) that will lead to this result. Some general statistics about the endgame tablebase are shown in Table 2. Each row summarizes the number of configurations it contains, and the distribution of obtainable results for the player to move. Furthermore, the longest sequence of moves that leads to a forced win for either player is also displayed.

Table 2: Summary of the endgame tablebase up to four pieces.

pieces	positions	wins	losses	draws	longest sequence
2	160,068	82,852	64,501	12,715	34
3	54,354,684	30,297,857	23,369,820	687,007	67
4	9,685,020,510	5,468,841,129	4,001,236,829	214,942,568	117

The endgame tablebase can be split into various *partitions*, based on which pieces are involved. After extracting *game features* from the configurations, a decision tree can be constructed for each partition using a technique such as described by Quinlan (1993). Given a configuration, we can calculate the game features from it and using the decision tree associated with the pieces in the configuration, determine the game theoretic value. Some resulting decision trees are shown in Figure 2. The features that were used for constructing these trees are listed in Table 3. Storing such a decision tree in memory takes less space then storing the entire endgame tablebase. Furthermore, this representation can yield interesting observations about the game.

Table 3: Game features extracted from the endgame tablebase used as split criteria.

feature	values	description
closest	$\{\mathrm{white}, \mathrm{black}\}$	the player that can reach the opposing den first
unopposed$_{\{\mathrm{w,b}\}}$	boolean	the piece $\{\mathrm{w}, \mathrm{b}\}$ can reach the den unopposed
sector$_{\{\mathrm{w,b}\}}$	$\{\mathrm{top}, \mathrm{mid}, \mathrm{bot}\}$	top: rank ≥ 7, mid: $3 <$ rank < 7, bot: rank ≤ 3; rank of piece $\{\mathrm{w}, \mathrm{b}\}$
distance$_{\mathrm{d}}$	integer [0–11]	Manhattan distance between the white piece and the white den
distance$_{\mathrm{p}}$	integer [0–14]	Manhattan distance between the two pieces
parity	$\{0, 1\}$	distance$_{\mathrm{p}}$ $\mathrm{mod}\, 2$
adjacent	boolean	distance$_{\mathrm{p}}$ $= 1$
trapped	boolean	Black is trapped
can cross	boolean	White can reach rank 7 unopposed on the shortest route to the black den

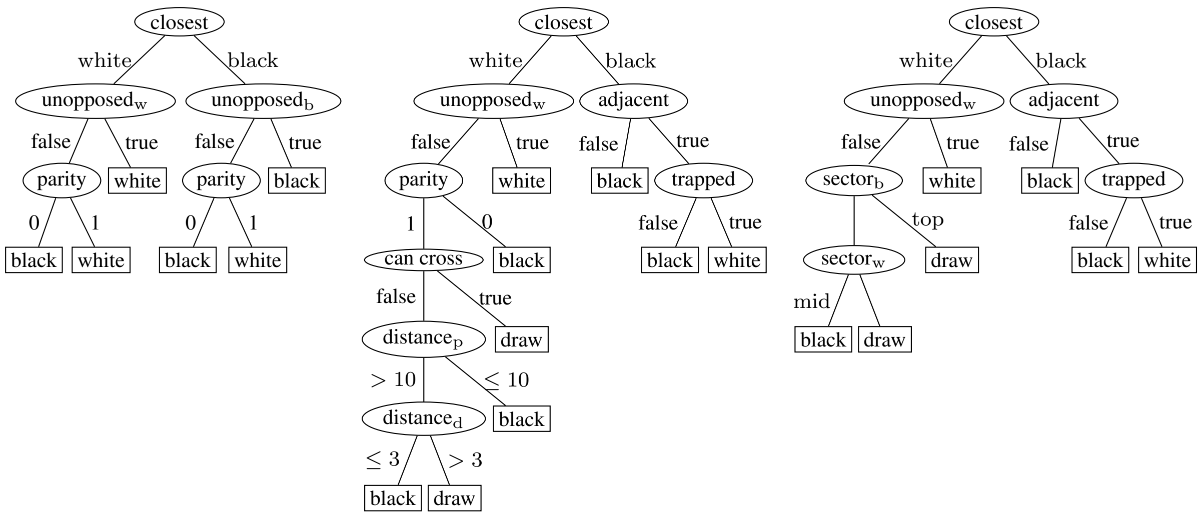
The feature *closest* determines which player has a piece closest to the opposing den, and therefore can reach it first. When both pieces are at the same distance, White is marked as closest since he moves first. The feature *unopposed* determines whether the player can reach the opposing den before the other player can get there. For White this is true iff there exists a black trap for which the manhattan distance between the white piece and that trap is smaller than the manhattan distance between the black piece and that trap (Black cannot move through its own den). The feature *sector* is derived directly from the rank of a piece, and divides the board evenly into three sectors. The feature *can cross* states whether the white piece can reach rank 7 unopposed, in such a way that it still minimises the number of moves to the opposing den.

4. ENDGAME ANALYSIS

Figure 2a shows the decision tree for each partition in which the two players have a piece of the same strength that cannot make leaps, e.g., white elephant vs. black elephant. It has been observed by van Rijn and Vis (2013) that these games do not end in a draw. Instead, there is a notion of *parity*, which determines the outcome of the game. This is illustrated in Figure 3a (White to move). Although White moves first and therefore can potentially reach the black den first, it cannot take the path between the rivers, since it will be captured by Black (recall that pieces of the same strength can capture each other). The white player cannot defend its own den due to the parity, and the black player has a straightforward win in 6 moves. Since tigers and lions can leap over the rivers, covering 3 squares in one move, they can reverse the parity. Still, no draws occur, but the decision tree is more complex. Figure 3b illustrates this. Although Figure 2a classifies this situation as a win for Black, this is actually a win in 10 for White. Playing a sequence[4] of: 1. Ta6 ta8, 2. Td6 ta7, 3. Td5 tb7, 4. Td4 tb3, 5. Td3 ta3 (…tb2, 6. Tc3 tb1, 7. Tc2 ta1, 8. Tb2 tb1, 9. Txb1), 6. Tc3 ta2, 7. Tc7 ta1, 8. Td7 tb1, 9. Td8 tc1, 10. Td9, White is able to reach the black den just before Black can reach the white den.

Figure 2b shows the decision tree for partitions in which the black player has a stronger piece. This tree does not apply to rats, tigers, and lions. When Black can reach the opposing den first, it is certainly a win for Black, except for the situations in which the black piece is trapped and the white player starts next to it. In the situations in which White is closer to the den, but cannot move unopposed to it, the situation is more complex. The strategy for White is to cross the board, and threaten to enter the black den. If (s)he can reach rank 7, then depending on (1) parity, (2) distance between the pieces, and (3) distance to the den for Black, the white player might be able to force a draw. Figure 3c illustrates an exceptional example in which White can force a draw. White cannot cross using the shortest route, but it can threaten the black den by a longer route. Black is forced to defend its den.

The situation gets more complex when lions or tigers are involved. Figure 2c shows the decision tree for partitions with a white tiger or lion versus a black elephant. Tigers or lions can reverse the parity making it less important. Like in the previous tree, if White is closest to the black den, but not unopposed, Black's role is to defend, forcing a draw. When White is in the mid sector, in some situations Black can force White out of the way, enabling him to reach the white den. In contrast to the other trees, this tree does not perfectly describe all configurations; 16 are misclassified. Figure 3d illustrates one of these misclassifications. Although Black is in the top sector, (s)he is able to force White out of the way as described before.

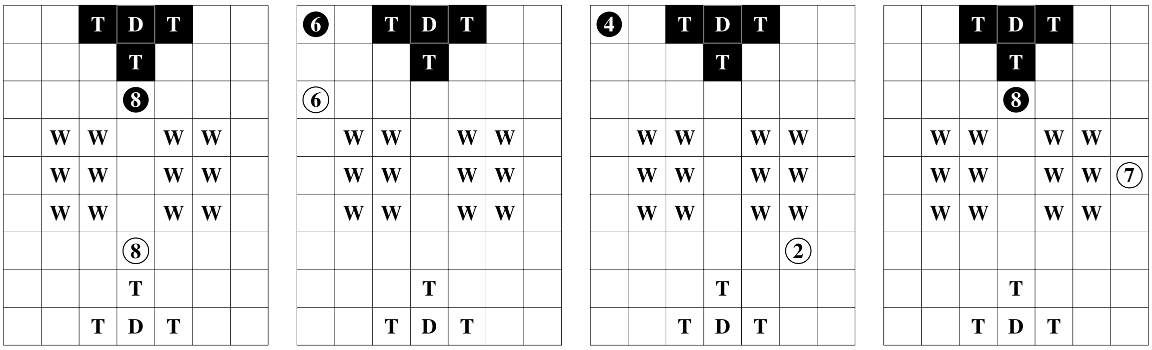


(a) Equal material (b) Black stronger (no rats, lions or tigers) (c) White lion vs. black elephant

Figure 2: Decision trees for partitions containing two pieces.

Using a set of reasonable game features (see Table 3), we can construct perfect (i.e., no misclassifications) decision trees for all partitions of two piece endgame tablebases. Moreover, we can construct a tree describing the whole two-piece endgame tablebase in a straightforward way. In some situations it is preferable to have a simpler tree, at the expense of a few misclassifications, such as in Figure 2c.

[4] Similar to the algebraic notation in chess, the characters denote which piece moves and to which square. An x indicates a capture.



(a) Elephant vs. Elephant (b) Tiger vs. Tiger (c) Cat vs. Dog (d) Lion vs. Elephant

Figure 3: Various endgames with two pieces.

5. CONCLUSIONS

We created a playing engine and constructed endgame tablebases for up to four pieces for the game Dou Shou Qi. Both can be used to gain novel insights into the intricacies of the game. It can also be considered as a first step towards theoretically solving Dou Shou Qi in the same way Schaeffer *et al.* (2007) solved Checkers. We have represented the two-piece endgame tablebases as decision trees, using a set of reasonable game features. From these trees some interesting insights have been gained, most notably the *importance of parity* and the *absence of draws* in equal-material endgames with two pieces only. Expanding the tablebase to more than four pieces is considered to be future work, as is the construction of decision trees for endgames with more than two pieces. We are convinced that we then find even more interesting patterns.

6. REFERENCES

Burnett, J. W. (2010). Discovering and Searching Loosely Coupled Subproblems in Dou Shou Qi. M.Sc. thesis, Tufts University, Medford.

Hoogeboom, H. J., Kosters, W. A., Rijn, J. N. van, and Vis, J. K. (2014). Acyclic Constraint Logic and Games. *ICGA*, Vol. 37, pp. 3–16.

Knuth, D. E. and Moore, R. W. (1975). An Analysis of Alpha-Beta Pruning. *Artificial Intelligence*, Vol. 6, No. 4, pp. 293–326.

Pritchard, D. B. and Beasley, J. D. (2007). *The Classified Encyclopedia of Chess Variants.* Beasley.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rijn, J. N. van (2012). Playing Games: The complexity of Klondike, Mahjong, Nonograms and Animal Chess. M.Sc. thesis, Leiden University.

Rijn, J. N. van and Vis, J. K. (2013). Complexity and Retrograde Analysis of the Game Dou Shou Qi. *Proceedings of the 25th Benelux Conference on Artificial Intelligence*, pp. 239–246.

Schaeffer, J., Björnsson, Y., Burch, N., Lake, R., Lu, P., and Sutphen, S. (2004). Building the Checkers 10-Piece Endgame Databases. *Advances in Computer Games*, pp. 193–210. Springer.

Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., and Sutphen, S. (2007). Checkers Is Solved. *Science*, Vol. 317, No. 5844, pp. 1518–1522.

Shannon, C. E. and Hsu, T. (1950). Programming a Computer for Playing Chess. *National IRE Convention.*

Thompson, K. (1986). Retrograde Analysis of Certain Endgames. *ICCA Journal*, Vol. 9, No. 3, pp. 131–139.

Zobrist, A. L. (1970). A New Hashing Method with Application for Game Playing. Republished (1990) (under the same title). *ICCA Journal*, Vol. 13, No. 2, pp. 69–73.