COMPUTING TIGERS AND GOATS

Lim Yew Jin[1] *J. Nievergelt*[2]

Singapore Zurich, Switzerland

ABSTRACT

Bagha Chal, or *Moving Tiger*, is an ancient Nepali board game also known as *Tigers and Goats*. We describe the game, some of its characteristics, and insights gained from an incomplete computer analysis. As in some other games such as Merrill's, play starts with a placement phase where twenty pieces are dropped on the board, followed by a sliding phase; during both phases pieces may be captured. The endgame-sliding phase is analysed exhaustively using retrograde analysis. After removal of symmetries, the complete database of all positions that arise consists of 88,260,972 entries. The placement phase involves a search whose game-tree complexity is estimated to be of the order 10^{41}. Partial results show that Tigers appears to have the better part of the opening, but Goats may catch up in the long run.

1. INTRODUCTION

Bagha Chal, or *Moving Tiger*, is an ancient Nepali board game that has recently attracted attention among game fans under the name *Tigers and Goats*. Murray (1952) describes a family of related games from South-East Asia, all sharing the characteristic that a few tigers face a larger number of goats (or men: 'orang-orang'). The goal is for the goats to immobilize the tigers, and for the tigers to kill a certain number of goats. This game between two opponents, which we call "Tigers" and "Goats", is similar in concept to a number of other asymmetric games played around the world - asymmetric in the sense that the opponents fight with weapons of different characteristics, a feature whose entertainment value has been known since the days of Roman gladiator combat.

On the small, crowded board of 5×5 grid points (shown in Figure 1), four tigers face up to twenty goats. A goat that strays away from the safety of the herd and ventures next to a tiger gets eaten, and the goats lose if too many of them get swallowed up. A tiger that gets trapped by a herd of goats is immobilized, and the tigers lose if none of them can move. Various games share the characteristic that a multitude of weak pieces tries to corner a few stronger pieces, such as *Fox and Geese* in various versions, as described in *Winning ways* (Berlekamp, Conway, and Guy, 2001) and other sources.

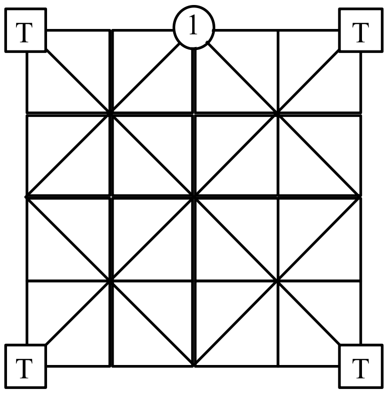

Figure 1a: The position after the only (modulo symmetry) first move by Goats that avoids an early capture of a goat.

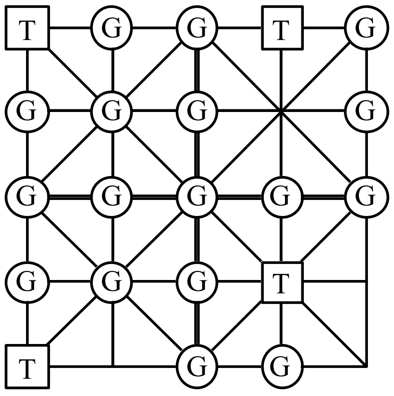

Figure 1b: Tigers to move can capture a goat, but thereafter the goats suffocate the tigers with a forcing sequence of 5 plies (challenge: find it).

[1] School of Computing, National University of Singapore, Singapore. Email: limyewji@comp.nus.edu.sg.
[2] Dept. Informatik, ETH Zurich, Zurich, Switzerland. Email: jn@inf.ethz.ch.

The rules of *Tigers and Goats* are simple. The game starts with the 4 tigers placed on the 4 corner spots (grid points), followed by alternating moves with Goats to play first. In a *placement phase*, which lasts 39 plies, Goats drops his[3] 20 goats, one on each move, on any empty spot. In that phase Tigers moves one of his tigers according to either of the following two rules:

- a tiger can slide from his current spot to any empty spot that is adjacent and connected by a line; or
- a tiger may jump in a straight line over any single adjacent goat, thereby killing the goat (i.e., removing the goat from the board), provided that the landing spot beyond the goat is empty.

If Tigers has no legal move, Tigers loses the game; if a certain number of goats have been killed (typically five), Goats loses. If neither of these conditions ever arises, the outcome is a draw by repetition. These rules are illustrated in the examples of Figure 2. They also show that Goats loses a goat within 10 plies unless his first move is on the centre spot of a border.

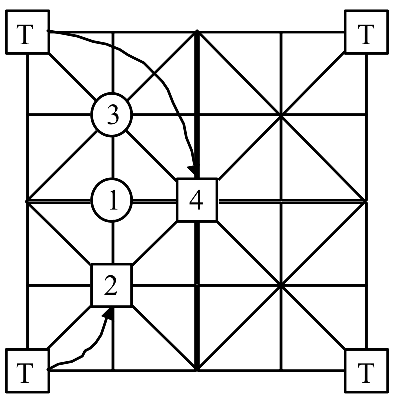

Figure 2a: Goats has 5 distinct first moves (ignoring symmetric variants). All but the first move shown in Figure 1a lead to the capture of a goat within at most 10 plies, as these two forcing sequences show.

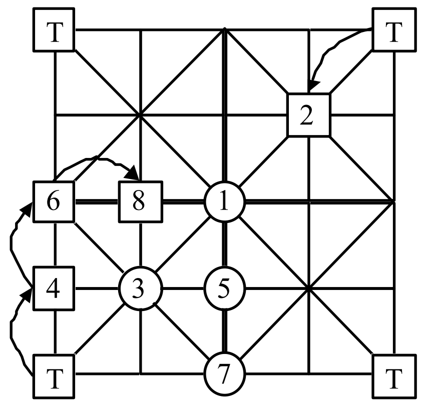

Figure 2b: Tigers' last move 8 sets up a double attack against the two goats labelled 1 and 3.

The 39-ply placement phase is followed by the *sliding phase* that can last forever. Whereas the legal Tigers moves remain the same, the Goats rule changes: on his turn to play, Goats must slide any of his surviving goats to an adjacent empty spot connected by a line. If there are 17 or fewer goats on the board, 4 tigers cannot block all of them and such a move always exists. In exceptional cases, which arise only if Goats cooperates with Tigers, the 4 tigers can surround 18 or more goats in a corner and prevent any goat moves. In this artificial scenario where Goats has no legal moves, Goats loses the game.

Although various web pages describing *Tigers and Goats* (see References) offer some advice on how to play the game, we have found no expert know-how about strategy and tactics. Plausible rules of thumb about good and bad play include the following. First, it is obvious that the goats have to hug the border during the placement phase - any goat that strays into the centre will get eaten, or cause the demise of some other goat, without any apparent benefit in return for the sacrifice. Therefore, Goats' strategy sounds simple: first populate the borders, and when at full strength, try to advance in unbroken formation, in the hope of suffocating the tigers. However, this recipe is simpler to state than to execute. Second, in contrast to Goat's strategy, we have not found any active strategy for Tigers. It appears that the tigers cannot do much better than to wait, "doing nothing", i.e., moving back and forth, until near the end of the placement phase. Their goal is to stay far apart from each other, for two reasons: (1) in order to probe the full length of the goats' front line for gaps; (2) so as to make it hard for the goats to immobilize all the four tigers at the same time. Tigers' big chance comes during the sliding phase, when the compulsion to move causes some goat to step forward and offers Tigers a forcing sequence that leads to capture. Thus, it seems that Tigers' play is all tactics, illustrating chess Grandmaster Tartakover's famous pronouncement: "Tactics is what you do when there is something to do, strategy is what you do when there is nothing to do."

[3] In contexts where the gender of a non-neutral third person is irrelevant, we will always use *he* and *his* to avoid the more cumbersome *(s)he* and *her/his*.

We are in the process of conducting an exhaustive analysis of this game with the goal of solving *Tigers and Goats* in the sense of determining the outcome: win, loss, or draw, under optimal play. We have partial results that shed light on the intricacies hiding inside this game that appears to admit such simple guidelines as "goats cautiously hug the border, tigers patiently wait to spring a surprise attack". For example, we know that Tigers can force the capture of 2 goats within 38 plies, i.e., within the placement phase, and can even force the capture of 3 goats. Despite Tigers' initial success, we do not know whether he can win the game - in fact, we conjecture that Goats has the upper hand and can at least force a draw. These results explain the seemingly arbitrary number five in the usual winning criterion "Tigers wins when 5 goats have been killed". This magic number '5' must have been observed as the best way to balance the chances. We know that Tigers can kill some goats, so Tigers' challenge must be more ambitious than "kill any one goat". In addition, experience suggests that once half a dozen goats are gone, they are all gone - Goats lacks the critical mass to put up resistance. But as long as there are at least 16 goats on the board, i.e., at most 4 goats have been captured, the herd is still sufficiently large to have a chance at trapping the tigers.

This game does not lend itself to mathematical analysis, other than to offer symmetries which we have fully exploited to reduce the size of the search space. The results have been obtained by the judicious use of standard search techniques: exploring the space from both ends, the starting position and the terminal positions. Retrograde analysis is used to build databases of all board positions during the sliding phase, with anywhere from 16 to 20 goats (Goats has lost when there are only 15 goats left). Forward searches from the starting position are used to explore the placement phase, ending in the database after 39 plies.

A 39-ply search with a branching factor that often exceeds a dozen legal moves is a big challenge. Therefore, most of these searches are designed to falsify a specific hypothesis or to verify a challenging assertion, such as "Tigers can kill at least one goat". By confronting a heuristic that generates good moves by Tigers against all the defensive Goats' strategies we verify the assertion using a search that is effectively at most 20 plies deep - a big reduction as compared to a 39-ply search. Similarly, a result like "Tigers can capture one goat at the latest at ply x, but cannot force a capture sooner", is reduced to two searches of depth only about x/2: a Tigers' heuristic vs. all defensive Goats' strategies, and a heuristic Goats' defence vs. all Tigers' attacks. The better the heuristic, the more likely these reduced searches succeed in verifying such an assertion. Thus, we see that insight into what makes a good move is essential for designing feasible searches. Given our lack of access to human expertise, we have developed player programs that learn from their experience of being pitted against each other.

This paper is organized as follows. Section 2 discusses the size and structure of the two relevant state spaces, corresponding to the placement phase and the sliding phase. We compute their sizes using Polya's (1937) theory of counting. Sections 3 and 4 describe the sliding-phase databases and the placement-phase forward searches, respectively, with statistical summaries. Section 5 summarizes implementation issues and the many experiments that were run over a period of several months. Section 6 concludes with an interpretation of the results obtained, and two open questions.

2. SIZE AND STRUCTURE OF THE STATE SPACE

The most important factor that determines the complexity of a search problem is the size and structure of the state space consisting of all the positions that must be considered. For *Tigers and Goats* it is convenient to partition this space into 6 subspaces.

S_0: all the positions that can occur during the placement phase, including 4 tigers and 1 to 20 goats.
S_k for k = 1 … 5: all the positions that can occur during the sliding phase, with 4 tigers, 21 - *k* goats, and *k* empty spots on the board.

Notice that any position in any S_1 to S_5 visually looks exactly like some position in S_0, yet the two are different positions: in S_0, the legal moves by Goats are to drop a goat onto an empty spot, whereas in S_1 to S_5, the legal moves are to slide one of the goats already on the board. For each subspace, a position is determined by the placement of pieces on the board, which we call the board image, and by the player whose turn it is to move. Thus, the number of positions is twice the number of distinct board images.

Since the game board has all the symmetries of a square that can be rotated and flipped, many board positions have symmetric *siblings* that behave identically for all game purposes. Thus, all the spaces S_0 to S_5 can be reduced in size by roughly a factor of 8, so as to contain only positions that are pairwise inequivalent. Using Polya's (1937) counting theory, paper and pencil, and the mathematics software MAPLE, we have computed the exact size of the symmetry-reduced state spaces (see Table 1).

	# of board images	**# of positions**
S_0	3,316,529,500	6,633,059,000
S_1	33,481	66,962
S_2	333,175	666,350
S_3	2,105,695	4,211,390
S_4	9,469,965	18,939,930
S_5	32,188,170	64,376,340

Table 1: Number of distinct board images and positions for corresponding subspaces.

As an example, Appendix B shows the computation of $| S_1|$; it is easy to perform by hand. One additional number is relevant for the definition of the indexing function that assigns an integer to each board position and helps to identify symmetric board positions: there are 1,666 inequivalent ways of placing 4 tigers on the game board.

S_0 is very much larger than all of S_1 to S_5 together, and has a more complex structure. Due to captures during the placement phase, play in S_0 can proceed back and forth between more or fewer goats on the board, whereas play in the sliding phase proceeds monotonically from S_k to S_{k+1}. These two facts suggest that the subspaces are analysed differently: S_1 to S_5 are analysed exhaustively using retrograde analysis, whereas S_0 is probed selectively using forward search.

3. DATABASES AND THEIR STATISTICS

Table 2 shows the distribution of won, drawn, and lost positions among the 88,260,972 positions in the spaces S_1 to S_5, i.e., during the sliding phase. A win by Tigers is defined as the capture of 5 goats, a loss for Tigers as the inability to move, and a draw (by repetition) is defined as a position where no opponent can force a win, and each can avoid a loss. In Table 2, GTM means Goats to move, and TTM means Tigers to move.

# of Goats Captured	GTM Wins	GTM Draws	GTM Losses	Total	TTM Wins	TTM Draws	TTM Losses	Total
4	913,153 (2.8%)	8,045,787 (25.0%)	23,229,230 (72.2%)	32,188,170 (100%)	30,469,634 (94.7%)	1,569,409 (4.9%)	149,127 (0.5%)	32,188,170 (100%)
3	1,315,111 (13.9%)	6,226,358 (65.7%)	1,928,496 (20.4%)	9,469,965 (100%)	6,260,219 (66.1%)	2,918,104 (30.8%)	291,642 (3.1%)	9,469,965 (100%)
2	882,523 (41.9%)	1,199,231 (57.0%)	23,941 (1.1%)	2,105,695 (100%)	465,721 (22.1%)	1,353,969 (64.3%)	286,005 (13.6%)	2,105,695 (100%)
1	252,381 (75.8%)	80,706 (24.2%)	88 (0.03%)	333,175 (100%)	6,452 (1.9%)	197,537 (59.3%)	129,186 (38.8%)	333,175 (100%)
0	30,609 (91.4%)	2,812 (8.4%)	60 (0.2%)	33,481 (100%)	146 (0.4%)	9468 (28.3%)	23,867 (71.3%)	33,481 (100%)

Table 2: Endgame database statistics.

Table 2 confirms the game player's experience that Goats stands a good chance of winning as long as he emerges from the placement phase with the loss of at most one goat. With two goats lost, Goats still has the upper hand, but the most likely outcome is a draw. Once three or more goats are lost, Goats' chances decline rapidly. Whereas statistical data by itself says nothing about optimal play, it gives some information about winning chances when play is affected by randomness, as is inevitably the case in human play.

An attempt to interpret win/loss statistics from a player's point of view shows that *Tigers and Goats* is a more complex game than the simplistic rule of thumb, "goats hug the border, tigers wait for a chance to pounce", suggests. If the quality of a position was determined mainly by the general location of pieces (e.g., goats bunched together safely near the borders, tigers spread far apart), the turn to move should not make much of a difference - but it does! For example, with one goat captured, Goats to move wins in 76 per cent of all positions; but if it is Tigers' turn to move, Goats wins only 39 per cent of the positions. In this instance, Goats' turn to move doubles his winning chances. Thus, we see that *Tigers and Goats* is a highly tactical game, sensitive to small changes due to a single move. We cannot expect to formulate robust positional evaluation functions based on general concepts analogous to "good shape" in Go, or a "solid pawn structure" in chess.

4. TREE COMPLEXITY OF THE PLACEMENT PHASE

The search space S_0, with approximately 6.6 billion positions, is too large for a static data structure that stores each position exactly once. Hence it is generated on the fly, with portions of it stored in hash tables. As a consequence, the same position may be generated and analysed repeatedly. A worst-case measure of the work thus generated is called game-tree complexity. The size of the full search tree can be estimated by a Monte-Carlo technique as described by Knuth (1975). The idea is to sample the width of the tree for a randomly chosen path for a number of times and use the samples to calculate an estimate of the game-tree complexity. For each of a number of random paths from the root to a leaf, we evaluate the quantity $F = f_1 + f_1 \times f_2 + f_1 \times f_2 \times f_3 + \ldots$, where f_j is the fan-out, i.e., the number of children, of the node at level j encountered along this path. The average of these values F, taken over the random paths sampled, is the expected number of nodes (other than the root) in the full search tree. Table 3 lists the estimated game-tree complexity (after the removal of symmetric positions) of five different "games", defined by their goal of capturing 1 to 5 goats during the placement phase. These estimates are based on 100,000 path samples.

Criterion	Estimated Tree Complexity
Kill 1 Goat	1.28×10^{24}
Kill 2 Goats	4.23×10^{36}
Kill 3 Goats	8.92×10^{38}
Kill 4 Goats	3.09×10^{40}
Kill 5 Goats	4.88×10^{41}

Table 3: Estimated Tree Complexity for winning criteria of capturing *k* goats during the placement phase.

5. IMPLEMENTATION, HEURISTICS, AND EXPERIMENTS

In order to verify their correctness, the endgame databases S_1 to S_5 of the sliding phase were computed twice, independently, using different software and hardware. The first computation, based on Gasser's (1996) algorithm, took approximately a week on a Pentium-4 personal computer. The second computation, based on the retrograde algorithm (Wu and Beal, 2002), took 4 days on a PowerPC G4 Apple computer.

We implemented standard alpha-beta search with hash-tables to perform the forward searches for the placement phase, each attempting to prove a specific hypothesis. The proof that Tigers can kill one goat took less than a day on a Pentium-4 PC. This was also the case for two goats. However, the attempted proof that Tigers can kill three goats, ran for a month without results. Next, we implemented a parallel alpha-beta search that distributes disjoint subtrees among the processes, to run on a Unix-based cluster of 48 Dual Pentium 3 computers. A root process expands the root of the search to a fixed depth of 5, and distributes the offspring among the processors without dynamic load balancing. Symmetric positions are reduced to a single canonical representation, and each process manages its own hash table of 8,000,000 entries. On this configuration, the proof that Tigers can kill three goats took about a week using 8 processors. Table 4 shows the number of nodes searched to prove each assertion.

Hypothesis	Number of nodes searched
Kill 1 Goat	2.8×10^{6}
Kill 2 Goats	1.1×10^{9}
Kill 3 Goats	8.5×10^{10}

Table 4: Number of nodes searched to prove various assertions.

Since Goats usually has more legal moves than Tigers, initial attempts to prove specific assertions based on the winning criterion of killing 5 goats focused on developing good heuristic Goats players. Recently, Chellapilla and Fogel (2001) implemented a co-evolutionary system with neural networks as evaluation functions that taught itself to play checkers at expert level. Co-evolving neural networks have also been shown to work well in other games such as Kalah (Oon and Lim, 2003). Co-evolution can be a flexible method for learning strategies to complex games where there is little available information about the domain. In the absence of expert knowledge for Tigers and Goats, we used co-evolutionary computing to learn a heuristic Goats player.

The setup of our co-evolutionary process follows the one used by Chellapilla and Fogel (1999), but instead of restricting the evaluation function inputs to the raw board, we selected 26 features of the board position as inputs. Examples of the selected features include the number of goats captured so far, the number of tigers which are currently trapped, the number of legal tiger moves, the number of goat moves which avoid immediate capture, and the total Manhattan distance of each pair of tigers. An initial population of neural networks is generated with random weights. Each neural network competes against a fixed number of randomly chosen opponents and its fitness score is determined by the results of these games. After every neural network has played its games, the “fittest” or highest-scoring networks are retained, while the remaining networks are removed. Each of the retained neural networks creates an offspring by mutating its weights. The process is then repeated until the desired number of iterations is reached.

The architecture of each neural network is illustrated in Figure 3: (1) the inputs have 24 nodes representing the current board position; moreover, there are 2 inputs directly linked to the output node; (2) there are 2 hidden layers and they consist of 12 and 5 nodes, respectively; (3) the single output node gives the evaluation of the board. The transfer function of each node in the neural network is the hyperbolic tangent (*tanh* bounded by ±1). Hence, an output value closer to 1.0 denotes a better position for the player to move, and a value closer to -1.0 denotes a worse position.

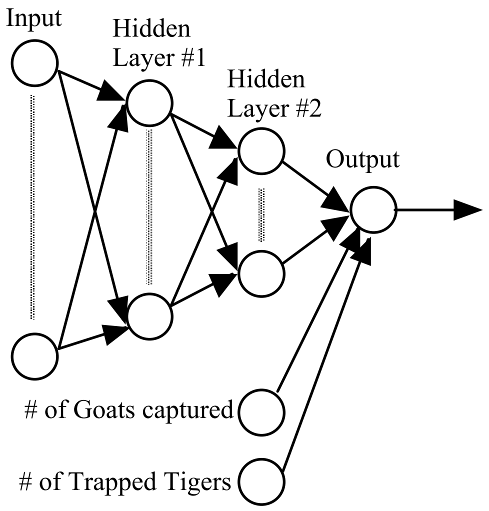

Figure 3: Neural network architecture of the evaluation function.

Instead of a single population of 30 neural networks that are co-evolved as in Chellapilla and Fogel (1999), we maintain two separate populations of 20 neural networks each. One population is designated as the evaluation functions for Goats and the other for Tigers. This setup allows us to learn good heuristics for each single player rather than to develop straightforwardly one overall position evaluation function.

Each neural network plays against four randomly chosen neural networks of the opposite player. The score awarded to a win is +1, to a draw is 0, and to a loss is -2. The fitness of each neural network is the sum of the scores achieved after playing the four opponents. Similar to Chellapilla and Fogel (1999), the top 10 neural networks of each population are retained for the next generation. Furthermore, each retained neural network generates one offspring by mutating its weights. This is the end of one generation and the process is repeated until 1000 generations have been produced.

Since one population of the neural networks is designated as evaluation functions for Goats, we are able to bias the strength of Tigers during play by setting the search depth of Tigers to 4 and the search depth of Goats to 2. This resulted in Tigers initially winning more often during the evolutionary process. However, Goats soon learned to draw the game despite having the handicap of a lower search depth. The handicap during the evolutionary process prepares the neural network for Goats to handle opponents that search more deeply than Goats does, and yet still be able to achieve a draw. This is vital as the top neural network for Goats after 1000 generations is used to re-order and prune the moves to be considered during the search. The fact that Goats were able to draw the game despite having the handicap mentioned leads to the conjecture that *Tigers and Goats* (with the winning criterion of 5 goats killed) is a draw or win for Goats.

The *Tigers and Goats* player that resulted from the neural network can be tested on the web (Lim, 2004). The lack of opponents, both human and computer, makes it difficult to assess the quality of the evaluation function learned. The authors, who do not claim to be expert players, have been unable to defeat the Goats player using a search depth of 4 ply. Additionally, the neural network has not lost to any program available online (T&G Players, 2004).

A proof of the assertion "Goats can at least draw the game", pitting a heuristic Goats player against all attacks by Tigers, breaks into six cases as shown in Figure 4.

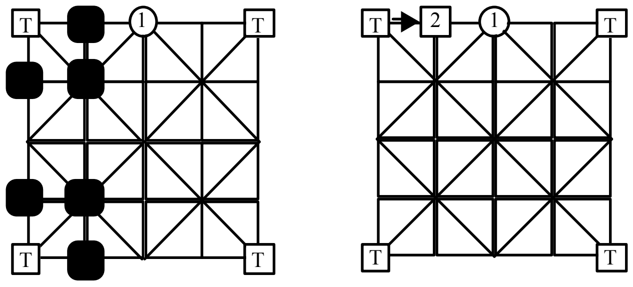

Figure 4: (Left) After Goats' only plausible first move, Tigers has 6 distinct replies (highlighted). One of these leads to the position shown at right that has been proven to be at least a draw for Goats.

Starting with the position shown at right in Figure 4, the top neural network obtained after 1000 generations succeeded in proving that Goats can at least draw the game for this particular opening. In each Goat node searched up to ply 29, the moves are re-ordered according to a 2-ply shallow search using the neural network as the evaluation function. Only the top-3 Goats moves are then tried. Using 10 Pentium 4 Xeons, each process managing its own hash table of 8,000,000 entries, a search that took 2 months proved this position to be at least a draw for Goats. Due to crashes and restarts, no reliable number of nodes searched is available.

The result that Goats can at least draw the game (with the winning criterion of 5 goats killed) after the most plausible first moves by both players leads us to the conjecture that the same assertion holds in general. Future work should focus on using more specialized techniques such as proof-number search (Allis, Van der Meulen, and Van den Herik, 1994), which are more suited to the uneven distribution of nodes in search trees of tactical games.

6. INSIGHT AND OPEN QUESTIONS

The insight obtained from the experiments reported is of two kinds: technical and "gamesmanship". From a purely technical point of view, we understand the combinatorial complexity of *Tigers and Goats*. We know that Tigers can force the capture of up to three goats, yet we have evidence for the conjecture that Goats has the upper hand, in the sense of being able to force at least a draw.

The key to successful forward searches through the state space S_0 of the placement phase is to replace a 39-ply search with a number of carefully designed searches that are effectively only 20 plies deep. This is achieved by (1) formulating hypotheses/assertions of the type "player X can achieve result Y"; (2) programming a competent and efficient heuristic player X that generates only one or a few candidate moves in each position; and (3) confronting the selective player X with his exhaustive opponent who tries all his legal moves. If this search that alternates selective and exhaustive move generation succeeds, the assertion Y is proven. If not, one may try to develop a stronger heuristic player X, or weaken the hypothesis, e.g., from "X wins" to "X can get at least a draw". We conjecture that continued computation of a similar nature as done so far, but with more powerful resources, will settle the currently open question whether Tigers can capture 5 goats and win the game.

The experiments also provide insight into the nature of *Tigers and Goats* from a game player's point of view. Humans, as poor calculators, are generally fond of games that admit a vague feeling of what features are correlated with good and bad positions. This allows them to replace concrete computation of variations by an intuitive assessment of what are likely to be strong or weak moves. For example, the vocabularies of chess and Go are full of terms that are effective and meaningful but defy an exact definition, such as "good or bad Bishop", or "territorial framework". For *Tigers and Goats* we have been unable to identify any such positional features beyond the obvious "stray goats are in danger". Lacking such positional guidelines, the course of a game appears to be a random walk in the dark, armed only with a short-range flashlight, trying to spot "combinations", i.e., forcing sequences that lead to capture. Perhaps the lack of positional concepts is merely due to insufficient experience with this game. Computers have established themselves as the only tool that allows us to get exact numerical answers to precise questions such as "who wins, how soon?" Will computers also help in discovering and formulating concepts that help humans develop an intuitive understanding of the nature of a position?

7. REFERENCES

Allis, L.V., Meulen, M. van der, and Herik, H.J. van den (1994). Proof-Number Search. *Artificial Intelligence*, Vol. 66, pp. 91-124. ISSN 0004-3702.

Berlekamp, E.R., Conway, J.H., and Guy, R.K. (2001). *Winning Ways for Your Mathematical Plays*. Vol. 1-4, 2nd Edition. A. K. Peters Ltd., Wellesley, MA. ISBN 1 5688-1130-6 (Vol. 1), 1-5688-1142-X (Vol. 2), 1-5688-1143-8 (Vol. 3), 1-5688-1144-6 (Vol. 4).

Chellapilla, K, and Fogel, D. (1999). Evolving Neural Networks to Play Checkers without Expert Knowledge. *IEEE Transactions on Neural Networks*, Vol. 10, No. 6, pp. 1382-1391. ISSN 1045-9227.

Chellapilla, K, and Fogel, D. (2001). Evolving an Expert Checkers Playing Program without Using Human Expertise. *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 4, pp. 422-428. ISSN 1089-778X.

Gasser, R. (1996). Solving Nine Men's Morris. *Games of No Chance* (ed. R. Nowakowski), pp. 101-113. MSRI Publications, Vol. 29, Cambridge University Press, Cambridge, UK. ISBN 0-521-57411-0.

Knuth, D.E. (1975). Estimating the Efficiency of Backtrack Programs. *Mathematics of Computation*, Vol. 29, pp. 121-136. ISSN 0025-5718.

Lim, YJ. (2004). An online version of our Tigers and Goats player can be tested at: www.educeth.ch/informatik/interaktiv/tigersandgoats/

Murray, H.J.R. (1952). *A History of Board Games other than Chess*. Oxford University Press, Oxford, UK.

Oon, W.C. and Lim, Y.J. (2003). An Investigation on Piece Differential Information in Co-Evolution on Games Using Kalah. *Proceedings of the Congress on Evolutionary Computation (CEC2003),* Vol. 3, pp. 1632-1638. ISBN 0-7803-7804-0.

Polya, G. (1937). Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und Chemische Verbindungen. *Acta Mathematica*, Vol. 68, pp. 145-253.

Romein, J.W. and Bal, H.E. (2003). Solving Awari with Parallel Retrograde Analysis. *IEEE Computer*, Vol. 36, No. 10, pp. 26-33. ISSN 0018-9162.

T&G Players (2004): www.cin.es/juegostablero/NepalI.html ; www.nepalhomepage.com/dir/entertainment/software/arun.html. xavier.bangor.ac.uk/xavier/SWGal/BagaChal/

Wu, R and Beal, D.F. (2002). A Memory Efficient Retrograde Algorithm and its Application to Chinese Chess Endgames. *More Games of No Chance* (ed. R.J. Nowakovski), pp. 213-227. MSRI Publications, Vol. 42, Cambridge University Press, Cambridge, UK. ISBN 0-521-80832-4.

Web references
We are not aware of publications on Tigers and Goats. Searching the web for Tigers and Goats, or Bagha Chal in various spellings, leads to some web sites that describe the game, e.g,,
- www.drizzle.com/~aganse/baghchal/baghchal.html
- www.thetrueillumanati.com/fur/games/bagha%20chal/
- http://rip.physics.unk.edu/nepal/games/bagh_chal.html
- www.barcelona2004.org/eng/eventos/juegos/baghchal.htm
- www006.upp.so-net.ne.jp/wkenji/

8. ACKNOWLEDGEMENTS

Elwyn Berlekamp pointed out *Tigers and Goats* and got us interested in trying to solve this game - like most exhaustive search problems, a task harder than it looks. Tony Tan showed us the simplest way of computing the number of distinct board positions. We are grateful to Thomas Lincke, Elwyn Berlekamp, and to anonymous referees for helpful comments that improved this paper.

9. APPENDICES

9.1 APPENDIX A: ADDITIONAL DATABASE STATISTICS

Tables 5 to 9 contain the statistics of the databases S_5 (4 goats captured) to S_1 (0 goats captured).

	Tigers to Move				
	Outcome	Tigers Wins	Draw	Goats Wins	Total
Goats to Move	Tigers Wins	23,217,329	11,249	652	23,229,230
	Draw	6,584,426	1,459,852	1,509	8,045,787
	Goats Wins	667,879	98,308	146,966	913,153
	Total	30,469,634	159,409	149,127	

Table 5: Statistics of database S_5 (4 goats captured).

	Tigers to Move				
	Outcome	Tigers Wins	Draw	Goats Wins	Total
Goats to Move	Tigers Wins	1,923,410	4,954	132	1,928,496
	Draw	3,879,335	2,342,280	4,743	6,226,358
	Goats Wins	457,474	570,870	286,767	1,315,111
	Total	6,260,219	2,918,104	291,642	9,469,965

Table 6: Statistics of database S_4 (3 goats captured).

	Tigers to Move				
	Outcome	Tigers Wins	Draw	Goats Wins	Total
Goats to Move	Tigers Wins	23,355	575	11	23,941
	Draw	354,808	838,197	6,226	1,199,231
	Goats Wins	87,558	515,197	279,768	882,523
	Total	465,721	1,353,969	286,005	

Table 7: Statistics of database S_3 (2 goats captured).

	Tigers to Move				
Goats to Move	Outcome	Tigers Wins	Draw	Goats Wins	Total
	Tigers Wins	63	25	0	88
	Draw	3,541	73,551	3,614	80,706
	Goats Wins	2,848	123,961	125,572	252,381
	Total	6,452	197,537	129,186	

Table 8: Statistics of database S_2 (1 goat captured).

	Tigers to Move				
Goats to Move	Outcome	Tigers Wins	Draw	Goats Wins	Total
	Tigers Wins	35	21	4	60
	Draw	59	1,505	1,248	2,812
	Goats Wins	52	7,942	22,615	30,609
	Total	146	9,468	23,867	

Table 9: Statistics of database S_1 (0 goats captured).

9.2 APPENDIX B: THE MATHEMATICS OF COUNTING APPLIED TO TIGERS AND GOATS

Consider the 5×5 *Tigers and Goats* board, with grid points numbered 1 to 25, and the 8 symmetry transformations that permute the set D = { 1 … 25 } as shown in Figure 5. We identify a board position with a function f : D → { Tiger, Goat, Empty } = { T, G, E } that assigns to each grid point its status, subject to the constraint that there are exactly 4 tigers and a number of goats that varies from 16 to 20. We show how to compute the number of distinct board positions, modulo symmetry, for the case of 20 goats, which we abbreviate as 4T1E, i.e., 4 tigers and 1 empty spot.

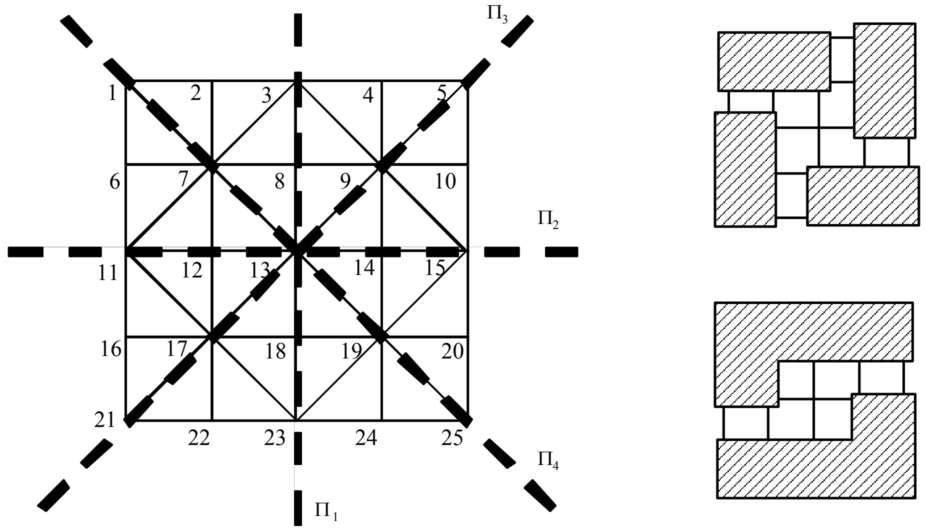

Figure 5: Any symmetry permutation of the square board can be obtained by some sequence of flips around the axes shown. The shaded areas at right are used in the analysis of rotational symmetries.

Let S = { f : D → { T, G, E } | f assumes 4 values T, 20 values G, 1 value E }
Observation: the group G of permutations of the domain D induces permutations on the set S of functions.
Definition: two functions f and g are equivalent iff there is a permutation P in G such that f = P(g).
Definition: a function f is invariant under permutation P iff f = P(f).
Definition: let I(P) be the number of functions f in S that are invariant under P.

Burnside's lemma: The number of equivalence classes of S under G is
1/ |G| • Σ *I* (*P*), where the sum is taken over all permutations P in G.

In order to evaluate Burnside's formula we investigate, for each of the 8 permutations of G, how many board positions remain invariant.

Identity permutation, '1'. Any board position remains invariant. There are 25 possibilities to place the empty spot, multiplied by 24-choose-4 possibilities to place the 4 tigers, resulting in 265,650 board positions invariant under the identity permutation.

Rotation by 90^o or by 270^o. The empty spot must be on the centre point 13, and each of the 4 tigers must be placed in "his own" 2 x 3 area (shaded in Figure 5, top right), such that the 4 chosen spots are symmetric under rotation (e.g., on spots 2, 10, 24, 16). Since the location of 1 tiger determines the place of the other 3 as well, there are 6 board positions invariant under rotations by $\pm 90^o$.

Rotation by 180^o. The empty spot must be on the centre point 13. Two tigers can be placed anywhere in the upper angular area consisting of 12 spots, shaded in Figure 5, bottom right. The location of the other two tigers is then determined by symmetry, resulting in 12-choose-2 = 66 invariant board positions.

Flipping around the vertical or horizontal axis through the centre. The empty spot can be anywhere along the axis, i.e., in 5 different places. After the empty spot has been placed, three cases of tiger placements must be distinguished:
a) four tigers located on the flipping axis: there is only 1 way to place them
b) two tigers located on the flipping axis (4-choose-2 ways to place them), one tiger in each half of the remaining board (10 ways to place the first tiger, the other is determined by symmetry)
c) two tigers in each half of the remaining board (10-choose-2 ways to place the first 2, the other 2 are determined by symmetry).
In total we have 5 × (1 + 10 × 4-choose-2 + 10-choose-2) = 530 invariant board positions.

Flipping around either diagonal axis through the centre. This turns out to be the same as flipping around the vertical or horizontal axis, resulting in 530 invariant board positions.

Summing up all these invariants and dividing by |G| yields:

1 Identity	25 × 24-choose-4 =	265,650
2 Rotations $\pm 90^o$		6
1 Rotations 180^o	12-choose-2 =	66
4 Flips		530
8 Permutations		267,848

Number of equivalence classes =
number of inequivalent board positions for 4 tigers and 20 goats = 267,848 / 8 = 33,481.

The calculations for 16 to 19 Goats are similar but more complicated, resulting in the following numbers of inequivalent board positions:

4T 20G 1E:	33,481
4T 19G 2E:	333,175
4T 18G 3E:	2,105,695
4T 17G 4E:	9,469,965
4T 16G 5E:	32,188,170

In each case, the number of inequivalent board positions is roughly 1/8 of the number of distinct positions if symmetry is ignored.