Constructing Pin Endgame Databases for the Backgammon Variant Plakoto

Nikolaos Papahristou and Ioannis Refanidis

Dept. of Applied Informatics, University of Macedonia
Egnatia 156, Thessaloniki, 54006, Greece

nikpapa@gmail.com and yrefanid@uom.gr

Abstract. Palamedes is an ongoing project for building expert playing bots that can play backgammon variants. Until recently the position evaluation relied only on self-trained neural networks. This paper describes the first attempt to augment Palamedes by constructing databases for certain endgame positions for the backgammon variant of Plakoto. The result is 5 databases containing 12,480,720 records in total that can calculate accurately the best move for roughly 3.4x10^{15} positions. To the best of our knowledge, this is the first time that an endgame database is created for this game.

Keywords: Retrograde analysis, endgame databases, Plakoto

1 Introduction

Computer game programs have been using endgame databases to great effect, especially in board games. Examples of complex games benefiting from such databases are chess [6], Chinese chess [4], checkers [10], awari [8], Kriegspiel [3], and nine-men morris [5], to name a few. Moreover endgame databases are catalytic in every attempt to solve a game, as can be seen in solved games like checkers [11], nine-men morris [5] and more recently heads-up limit texas holdem poker [2].

An endgame database usually contains precomputed game-theoretical values (or near perfect heuristics) for each position record. The game playing program can use this database by searching the records when an endgame position contained in the database is reached by the AI search. The benefits for the program are multiple: firstly, the value retrieved from the database is more accurate than the program's evaluation function; secondly, the retrieval of the database value is typically faster than the evaluation function execution speed; thirdly, there is no need to search any further down the tree.

The endgame databases can also provide a powerful analytical tool for game professional and for understanding the game in general. A prominent example is chess, where positions which humans had analyzed as draws were proven winnable and vice-versa. Also the database constructed when heads-up limit texas holdem poker was solved [2] offered insights that contradicted some human beliefs about the best play in this game.

Backgammon programs also make use of endgame databases. These usually cover the positions where both players have their checkers in the bearoff quadrant (also known as *bearoff* databases). In the two-sided version, these databases offer the game-theoretic value of the position whereas in the one-sided version, the goal being to minimize the average number of rolls to bearoff, a distribution of the expected number of rolls to bearoff. The one-sided version is much smaller than the two-sided one but it is not as accurate with respect to finding the best move.

Our main focus is several backgammon variants that are not yet sufficiently examined by computer games research. The aforementioned bearoff endgame databases can be used in many of the variants that we are interested in (Portes, Plakoto, Fevga, Narde), since the bearoff positions of backgammon can occur in all of these games as well. For this purpose, Palamedes already contains a two-sided bearoff database that was constructed using similar techniques used by other programs. This database gives the game theoretical value of all bearoff positions when the doubling cube is not used and is 5.48GB in size.

This paper describes our efforts of our first attempt to construct endgame databases for positions only seen in the Plakoto variant. To the best of our knowledge, this is the first time this kind of endgame database is constructed. We believe this is the first step towards constructing bigger and better endgame databases in the future. Section 2 presents the rules of the Plakoto variant and introduces the position types that are stored in the databases. Section 3 describes the algorithm used to compute the databases. Section 4 discusses several issues including perspectives on building larger databases. Finally Section 5 concludes and gives avenues for future work.

2 The rules of the Plakoto variant

This section presents the rules of the Plakoto variant and analyzes the positions that we are interested in storing in the databases.

2.1 The Plakoto variant

The general rules of the game are the same as the regular backgammon apart from the procedure of hitting. Players start the game with fifteen checkers placed in opposing corners (Fig 1.a.) and move around the board in opposite directions until they reach the home board which is located opposite from the starting area. Players can win a single or double game as usual but there is no triple win as in backgammon. Finally, the doubling cube is not typically used.

The key feature of the Plakoto variant is the ability to pin hostile checkers, so as to prevent their movement. When a checker of a player is alone in a point, the opponent can move a checker of his own in this point thus pinning (or trapping) the opponent's checker. This point counts then as a "*made point*" as in regular backgammon, which means that the pinning player can move checkers in this point while the pinned player cannot. The pinned checker is allowed to move normally only when all opponent pinning checkers have left the point (unpinning). More detailed explanation of the Plakoto rules can be found in http://ai.uom.gr/nikpapa/Palamedes/manual/#Plakoto.

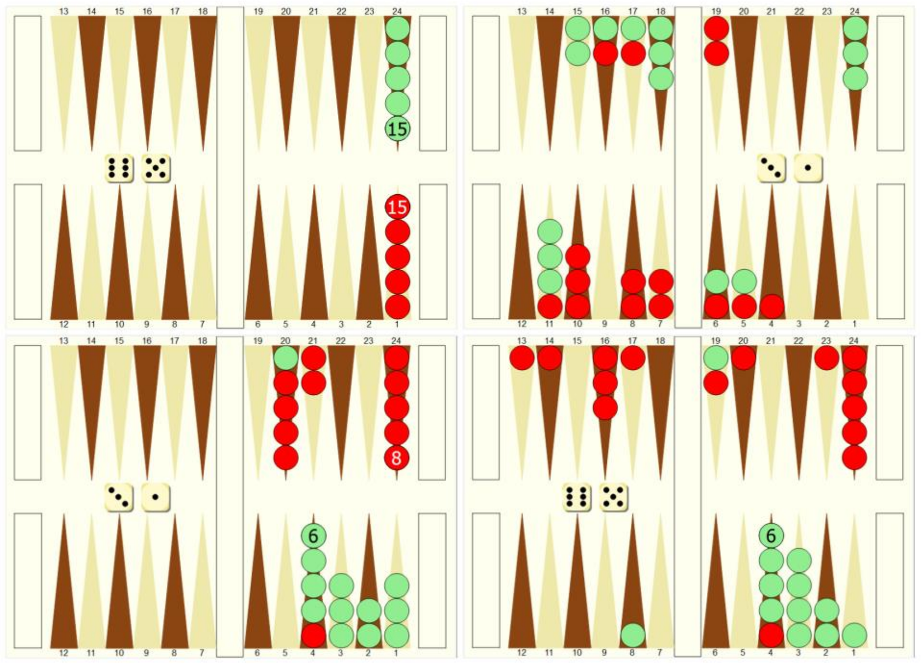

Fig. 1. Various Plakoto positions a) Upper left: Starting position. Red player starts at point 1 and bears off at point 24, while green player starts at point 24 and bears off at point 1, b) Upper right: Typical middle-game position c) Lower Left: Endgame position where both players have pins in their bearoff quadrant d) Lower right: Both players have pins in their bearoff quadrants and some checkers in the previous quadrant.

2.2 Endgames with pins

Strategically thinking, pinning is the most important characteristic of the Plakoto variant for the following reasons:

1. A "made point" can be constructed with only one checker instead of the usual two which makes primes and other formations easier.
2. Players can nullify bad luck when they roll small rolls and/or the opponent rolls big rolls. This is true because running to the bearoff phase is unimportant when one or more checkers are trapped.
3. The side that has pinned without getting pinned usually gets a few rolls ahead in the bearoff race. The further ahead the pin is, the bigger the advantage.

A typical occurrence in a Plakoto game is for both players to have pinned each other. Then the best strategy usually is to try to maintain the pin(s) for as long as possible trying to make the opponent unpin his own pins. This is especially true in race situations (like Fig.1.c, Fig.1.d.), where no more pins are possible. For this initial exploration on Plakoto pin databases, we are interested in positions with the following characteristics:

- The side to move has pinned the opponent exactly once inside her bearoff quadrant (points 2-6)[1].
- The opponent has pinned the moving player exactly once.
- No more further pins are possible. In this paper, these no-contact positions are also called *race* positions.

These endgames eventually resolve by one player unpinning his pin, followed by the other player moving his newly freed checker to begin the bearoff. One reason that we are interested in these endgames is that they take place frequently in practice. In an initial 100,000-game self-play experiment with Palamedes best neural network Pla-koto-5 [7] at the highest settings, we found out that these endgames occur in 14% of games played.

2.3 Number of endgame positions

The number of positions (R) of C checkers residing inside P points can be calculated by the following formula [9]:

$$R = \binom{C+P-1}{C} = \frac{(P+C-1)!}{C!\,(P-1)!} \quad (1)$$

The number of checkers for the positions of interest is 13 (one checker is pinned by the opponent, and one checker must always be at the pinning point to maintain the pin). Depending on the memory needs of the game playing program a different number of points (P) can be used. For example for P = 6 (all the non-pinned checkers are under the 6-point, i.e. inside the bearoff quadrant) the total number of positions is 8568 per pin placement. Such a position is shown in Fig.1.c. For the remainder of this paper all discussion takes place under the assumption that all unpinned checkers of the player to move is under the 12 point (P = 12, R = 2496144). A sample position can be seen in Fig.1.d. Note that in both Fig.1.c and Fig.1.d, the position is valid for database retrieval for either player to move.

We have constructed a different database for each possible pin point of the moving player (2-6), so we have 5 databases and 12,480,720 positions in total for the 12-point version. This database is one-sided and corresponds to half the board. If we assume that the opponent has a similar position to the other half, the total possible 2-sided “true” positions that these databases can apply is $12{,}480{,}720^2 = 155{,}768{,}371{,}718{,}400$. If we further assume that the opponent is pinning at the full half of his board (points 13-23) then the total applicable positions are 12,480,720 x 2,496,144 x 11 =

[1] No-contact positions where a player has pinned the 1-point (also known as “mana” point) are proven double wins for the pinning player except for the rare cases when the opponent has also pinned the 1-point (tie).

342,690,417,780,480. This number is the lower bound because the endgame characteristics set in the previous section can be met in positions where the opponent player has checkers below the 12 point.

3 Algorithm

The goal of the players in the endgame positions that we are interested in is to maintain his pin as long as possible. Essentially, the player is playing a mini-game where he tries to maximize the number of moves keeping the pin. Since the game has a chance layer, this goal becomes the maximization of the average distance to unpin. Due to the fact that there is no contact, this metric can be computed using a one-sided database.

3.1 Plakoto endgame pin database algorithm

The procedure we use is inspired by retrograde analysis [12] where the algorithm starts from a terminal position and works backwards. In our case we do not have terminal positions, but we are starting at a position where all checkers have been moved the furthest. This is the position where all 13 checkers are placed at the last point (point 1). The procedure then works backwards as usual.

Algorithm1. Plakoto endgame pin database creation

```
pinDatabase(p, pinPlacement)
position ← createStartPosition(pinPlacement)
endPos ← createEndPosition(pinPlacement, p)
while position is not endPos do
  saveInDB(hash(position), findDistance(position))
  increment(position)
end while

function findDistance(position)
  avgDistance ← 0
  for every roll d of the 21 possible rolls do
    afterStates ← findMoves(position, d)
    distances ← readDistancesFromDB(afterStates)
    distance ← max(distances)
    if d is double roll
      avgDistance += distance
    else
      avgDistance += 2 * distance
    end if
  end for
  return avgDistance / 36
```

The database creation algorithm is shown above (Algorithm 1). For every position encountered and all 21 rolls, we find all the legal afterstates, retrieve the distances and return the max distance. The distance of the current position is then calculated as the weighted average of all rolls and stored in the database. The algorithm increments the position and begins the next iteration until all positions are exhausted. The position is incremented in such a way that the resulting afterstates will always have a distance in the database. The only exception is when the roll has no moves, but we can find the distance of this case with a simple recursive operation.

During actual play the database is activated when the position before the roll has the characteristics described in section 2.2. We retrieve the distances of all the afterstates and we select the move which results in the largest distance.

3.2 Storage and Hashing

Important properties for many endgame databases are the storage and the compression mechanisms used. We use a modified version of the hashing function used in [1] to encode the board position to a 32-bit integer. This function is fast, gives a perfect hash, and can be easily decoded for the reversed procedure (int to position). Since the number of records is relatively small we have not made any attempts to compress the database. For the same reason we store the distance value as a double for maximum precision, although it may not be needed. The minimum amount of precision that is acceptable for best play is left for future work. The final database size is 19MB for the 12-point, and 67Kb for the 6-point version per pin placement.

4 Discussion

In this section we discuss potential problems with the one-sided databases and conduct two experiments to evaluate our existing AI in positions from the database.

4.1 Potential problems with one-sided databases

One problem with one-sided databases is that it may give errors in actual play when we take into account the opponent. This is already documented for the one-sided bearoff databases used in backgammon [9]. We identified one possible problem case in our databases in a very rare situation where the player to move has a high average distance to unpin for all available moves and the opponent is almost ready to unpin. In this case, because the unpinning of the opponent is almost certain, it may be best for the moving player to prepare for a better placement in the bearoff quadrant instead of continuing to maximize his distance to unpin. However, rollout experiments in 5 samples of such cases have not given evidence that one strategy is better than the other. We believe the problem exists in the bearoff databases because the problematic bearoff positions are near the end of the game while our “problematic” positions being much further away from terminal, allow the luck factor to “wash out” any small errors.

4.2 Using the databases to evaluate the neural networks

Another interesting use of endgame databases (or databases of solved games) is to evaluate existing AI implementations. We conducted experiments with Palamedes using the best neural network (NN) available for Plakoto: a) firstly for all database positions and all possible rolls we checked if the best move of the NN coincided with the best as seen in the databases, and b) secondly we played 100,000 self-play games with the NN and when a database position was encountered we again compared the move chosen by the NN to the database's optimal. For the first experiment we constructed the opponent position as a mirror of the player to move.

Table 1. Evaluation of Palamedes AI in Plakoto pin endgames

Comparison Method	Correct moves by the NN (%)
All positions	15%
Self-play positions	64%

As can be seen the NN is not selecting the best move 85% of the time in the first test, however it is doing noticeably better at positions found in practical play. We believe this is normal behavior for the NN to score so low in the first test because the self-play procedure used to train the network certainly could not generalize well to all possible cases most of which are corner cases rarely to be seen in actual play. The result of the second test shows the importance of such databases to enhance the move selection mechanism of the existing AI.

5 Conclusion and future work

We have presented an algorithm that created several one-sided endgame databases for the game of Plakoto. The databases are small but can be applied to a huge number of endgame positions. To the best of our knowledge this is the first time that endgame databases are created for the game of Plakoto. We have also shown that the usage of these databases greatly enhances our AI's move selection.

There are several avenues to build upon these results. An obvious one is to construct more databases with the same method. We have only built databases for 2-6 pinned points, pinned points 7-18 can be easily created. Also databases with more than one pin per side are possible. The conversion of our algorithm to race endgames where the opponent has pinned more than one checker is straitghtforward. A more difficult case is when the moving player has two or more pins.

With the presence of these databases our neural network evaluation function does not need to generalize in these types of positions. We could improve the representation power of our network by retraining the NN without taking into account these endgames.

Finally we would also like to explore compression techniques for storage. This will be essential for the creation of larger pin endgame databases.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful comments and suggestions that contributed to improving the final version of the paper.

References

1. Benjamin, A., Ross, Andrew, M.: Enumerating backgammon positions: the perfect hash. Interface: Undergraduate Research at Harvey Mudd College, 16 (1), 3–10 (1996)
2. Bowling, M., Burch, N., Johanson, M., Tammelin, O.: Heads-up limit hold'em poker is solved. Science 347(6218), 145–149 (2015)
3. Ciancarini, P., Favini, G.P.: Solving kriegspiel endings with brute force: the case of KR vs. K. Advances in Computer Games, pp. 136–145 (2010)
4. Fang, H.R., Hsu, T.-s., Hsu, S.C.: Construction of Chinese chess endgame databases by retrograde analysis. In: Marsland, T., Frank, I. (eds.) Computers and Games. Springer, Heidelberg, pp. 96–114 (2002)
5. Gasser, R.: Solving nine men's morris. Computational Intelligence, 12(1), 24–41 (1996)
6. Nalimov, E.V., Haworth, G. McC., Heinz, E.A.: Space-efficient indexing of chess endgame tables. ICGA Journal, 23(3), 148–162 (2000)
7. Papahristou, N., Refanidis, I.: On the Design and Training of Bots to Play Backgammon Variants. In: Iliadis L., Maglogiannis I., Papadopoulos H. (eds.) 8th IFIP WG 12.5 Artificial Intelligence Applications and Innovations (AIAI 2012), IFIP AICT, vol. 381, pp. 78–87, (2012)
8. Romein, J.W., Bal, H.E.: Solving awari with parallel retrograde analysis. Computer, 36(10), 26–33 (2003)
9. Ross, A.M., Benjamin, A.T., Munson, M.: Estimating winning probabilities in backgammon races. In Ethier, Stewart N. and Eadington, William R. (Eds.), Optimal Play: Mathematical Studies of Games and Gambling, pp. 269–291. Institute for the Study of Gambling and Commercial Gaming, University of Nevada, Reno (2007)
10. Schaeffer, J., Bjornsson, Y., Burch, N., Lake, R., Lu, P., Sutphen, S.: Building the checkers 10-piece endgame databases. In van den Herik, H. J., Iida, H., Heinz, E. A. (eds.), Advances in Computer Games: Many Games, Many Challenges, Kluwer Academic Publishers, pp. 193–210 (2003)
11. Schaeffer, J., Burch, N., Björnsson, Y., Kishimoto, A., Müller, M., Lake, R., Lu, P., Sutphen, S.: Checkers is solved. Science 317, 1518–1522 (2007)
12. Thompson, K.: Retrograde analysis of certain endgames. ICCA Journal, 9(3), 131–139 (1986)