FairKalah: Towards Fair Mancala Play

Todd W. Neller(✉) and Taylor C. Neller

Gettysburg College, Gettysburg, PA 17325, USA
tneller@gettysburg.edu
http://cs.gettysburg.edu/~tneller/

Abstract. Kalah (a.k.a. Mancala) is a two-player game of perfect information that has been a popular game for over half a century despite a strong first player advantage. In this paper, we present initial game states that are fair, as well as optimal play insights from analysis of optimal and suboptimal states.

1 Introduction

Kalah (a.k.a. Kalaha), usually called "Mancala" in the United States, is a Mancala game variant invented in 1940 by William Julius Champion and patented in 1955 as US Patent 2,720,362 [3]. In the Mancala games family, it is most closely related to Dakon [10, p. 75] from Java. Despite being a perfect information game with a strong first player advantage [5], Kalah has enjoyed popularity in the United States and Germany through its history. In this paper, we share initial game states that are fair[1] and provide insights to game play from analysis of optimal and suboptimal states.

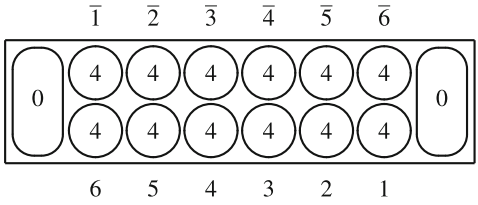


Fig. 1. Kalah initial board state with 4 pieces per play pit

Kalah (a.k.a. Mancala) is played on a rectangular board as depicted in Fig. 1 with 6 play pits for each player along the long side and 1 score pit ("Kalah") for each player on the player's right-hand end of the board. In the standard initial board state, each of the play pits starts with 4 pieces (a.k.a. seeds, counters) per pit. In constrast to prior rules, we label play pits by the number of pits they are clockwise from that player's score

[1] In this paper, we use "fair" in the sense that two perfect players would draw the game. There is no game-theoretic advantage for either player.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
C. Browne et al. (Eds.): CG 2022, LNCS 13865, pp. 3–13, 2023.
https://doi.org/10.1007/978-3-031-34017-8_1

pit. Pits of the second player (a.k.a. "north player") are notated with overbars. Numbers depicted within pits indicate the number of pieces in that pit[2].

On a player's turn, the player selects one of their play pits containing pieces, removes all pieces from that play pit, and redistributes ("sows") them counterclockwise, one piece per play/score pit, and skipping their opponent's score pit. If the last piece redistributed lands in their score pit, the player takes another turn, i.e. gets a free move; otherwise, it becomes the opponent's turn. If the last piece redistributed lands in an empty play pit of the player (including after having redistributed pieces to all opponent play pits), then that piece captures: both that piece and any pieces in the opponent's opposite play pit are removed from the pits and placed into the player's score pit[3].

The game ends when, at the end of a turn, there are no pieces left in one player's play pits. Their opponent scores any remaining pieces in play pits. The player with more pieces in their score pit wins. If the number of pieces in the score pits are the same, the players draw (i.e. tie) the game.

While boards and pieces are readily available commercially, this game has often been played with egg cartons and bowls as play and score pits, respectively. Glass beads, beans, stones, and shells are just some of the materials used throughout the world to play Mancala games.

Featuring simple rules and accessibility of materials, Kalah has been successfully popular, but with a little play, one can easily discern that opening with plays 4 and 1 put the second player at a significant advantage. Irving, Donkers, and Uiterwijk computed that optimal play for the standard initial board results in a first player win by 10 points (39–19). In our notation, one line of optimal play of Fig. 1 is as follows: 41, $\overline{56}$, 3, $\overline{3}$, 1513, $\overline{5}$, 2, $\overline{1}$, 621, $\overline{2}$, 131214, $\overline{13}$, 2, $\overline{5}$, 3, $\overline{4}$, 6, $\overline{12}$, 5, $\overline{3}$. Using this notation has the convenience that when there are free moves on a turn (e.g. 131214), the number of pieces moved is the same as the pit label for all except the final move.

In this paper, we present initial states that are *fair*, i.e. that result in a draw between two optimal players. *FairKalah* is the name we use for Kalah played from a fair initial state. Additionally, we analyze optimal play data in order to yield optimal play insights.

2 Algorithmic Approach

Our approach to computing fair initial states for FairKalah is based on Java code we ported from Irving's original C code used for the results of [5]. We computed a 24-piece endgame database (1.16 GB), so search could terminate when half of the pieces had been scored. The standard initial board has 48 pieces with 4 pieces per play pit. At the top level of our algorithm, we enumerated all possible ways that one could move one or two of these pieces to different pits, including score pits[4] (e.g. Sect. 7 boards).

[2] Thus, a potential free move is easily seen where the number of pieces matches our numeric label for that pit.

[3] There is some controversy over whether "empty captures" were intended, as they were disallowed in Dakon. However, a strict reading of the patent makes no requirement for the number of opponent pieces opposite. Our interpretation is consistent with [5] and most printed rules.

[4] When we have a fair game with piece(s) in the score pit, we have essentially created a fair game with fewer piece(s) and a perfect komi compensation for fairness.

For each candidate initial game state, we evaluate the game value using the MTD(f) algorithm [4,9]. "Game value" here has more information than win/lose/draw; it is the score difference at game end with optimal play. Search was highly optimized with the MTD(f) algorithm, our large endgame database, and Irving's node ordering: transposition table move, free moves right-to-left, captures right-to-left, and remaining moves right-to-left. For MTD(f), we used an initial game value guess of 0, with Plaat's "next" choice being that of Van Horssen's Algorithm 2, and using Van Horssen's "SAFE" choice for storing the new best move in the transposition table.

After computing all 3 fair boards needing one piece movement and all 251 fair boards needing two piece movements (Sect. 7 Appendix), we computed and stored all non-terminal states along all possible optimal lines of play for these 254 fair initial positions.

3 Analysis of Fair Optimal Game Trees

A *fair optimal game tree* (FOGT) is a subtree of a game tree where the root game state of the subtree has game value 0 (i.e. results in a draw with perfect play), and each state node retains only successor-state children with game value 0. That is, a fair optimal play tree contains best plays of a drawn (sub)game.

There are pros and cons of analyzing data for optimal decisions from all game tree nodes, which we do later in the paper. On the positive side, one gains a complete picture of optimal play for any game states. On the negative side, the distribution of states includes mainly states that optimal players would not allow. That is, the distribution of states is skewed towards irrelevance in optimal game play.

In contrast, FOGTs provide the opportunity for focused insight on optimal play. We can ask questions of a state distribution where *both* players are playing optimally, gaining insight to what the highest level of Kalah play looks like. In the following subsections, we share raw and derived features of our collected game states as well as insights from the data.

3.1 Features

Of special interest are statistical and machine learning insights on the choice of optimal plays themselves. For this exploration, we define features that we conjecture may be relevant to optimal play decision-making and/or node game value prediction. These include both raw state variables and derived variables:

depth the number of plays made so far in the game (not included in model input)

player the current player p, either 0 or 1 for first/"s"/south player or second/"n"/north player, respectively (not included in model input)

pit_6, . . ., pit_1, score, pit_6o, . . ., pit_1o, score_o the number of pieces in each pit starting with the current player's leftmost play pit and proceeding counter clockwise around the board

counters, counters_o the sum of pieces in the current player's and opponent's play pits, respectively

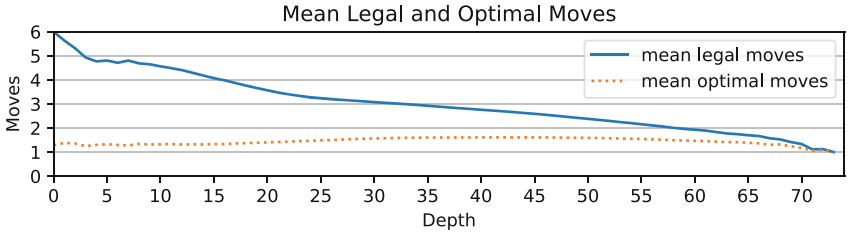


Fig. 2. Branching factor and optimal moves by depth.

free_moves, free_moves_o the number of current player and opponent play pits, respectively with pit index n containing n pieces
is_free_move_n whether or not pit n has n pieces
is_closest_free_move_n whether or not a pit n is the closest free move to the score pit
capture_move_n number of pieces captured by playing pit n
move_count number of non-empty pits on the current player's side of the board
move_count_o number of non-empty pits on the opponent's side of the board
relative_mobility move_count minus move_count_o
is_closest_move_n whether or not a pit n move is the closest move to the score pit
score_gain_n the number of pieces scored after pit n is moved
is_largest_capture_move_n whether or not a pit n move captures the largest number of pieces among all capture moves
largest_capture the largest number of pieces that the current player could capture
score_diff the current player's scored counters minus their opponent's scored counters
counter_diff the current player's in-play counters minus their opponent's in-play counters
is_exposed_n whether or not pit n has no pieces across from it
piece_threat_n number of pieces in pit n under immediate threat of capture
game_val The current player score minus the opponent score if the game was played to completion with perfect play (0 for fair game states)
optimal_n whether or not pit n of the current player is an optimal move

3.2 Optimal Play Branching Factors and Tension

From the 9,991,466 unique nonterminal optimal play states of these MTD(f) searches (including optimal endgame database states), there are 28,366,064 total plays and 15,828,178 optimal plays, yielding average play and optimal play branching factors of approximately 2.839 and 1.584, respectively. The number of states with 1–6 optimal moves are, respectively: 5,619,419, 3,188,759, 933,219, 219,859, 29,112, and 1,098. Thus, 88.16% of states have 1 or 2 optimal plays. Most states with 6 optimal plays avoid playing into the opponent play pits with the opponent having a forced emptying of their pits on the next turn (e.g. 1 piece adjacent to their score pit).

For FOGT states, there is a decreasing trend in both branching factor and number of optimal moves with increasing depth, as shown in Fig. 2. Whereas the branching factor is always 6 at the root, the mean branching factor non-monotonically decreases to 1 as

the number of pieces on the board decreases with depth. However, the mean number of optimal moves remains relatively steady, between 1 and 2 throughout play, slightly increasing for moderate depths.

As a result FOGT game tension, defined as the fraction of game-losing moves, is shown in Fig. 3. This derives from Cameron Browne's measure of puzzle tension [2], i.e. the fraction of losing plays. Here, we observe that tension is close to $\frac{5}{6}$ for root nodes, indicating that most but not all FairKalah roots have a unique optimal first play. This then non-monotonically decreases with depth down to 0.

A significant implication of this is that real-time management of play should favor computation distributed more towards early decisions. We would expect a precomputed opening book with a time-management policy such as OPEN [1] to be advisable for blunder avoidance when using limited memory and time, respectively.

3.3 Strategic Insights

Recall that a player gets a free move, i.e. another turn, when playing n pieces from pit n and having the last piece redistributed to the player's score pit. Of the 9,991,466 unique nonterminal optimal play states, 5,558,640 (55.63%) offer free moves. Of these, 5,326,495 (95.82%) have the closest free move to one's score pit as an optimal move. Thus, if one has one or more free moves, one should most likely play the closest free move to one's score pit.

In the remaining 232,145 (4.18%) states, the strategic considerations are diverse. In some states, taking the free move would force one to then play a closer pit that would spill one more pieces onto the opponent's side. In others, it is preferable to accumulate pieces in low index pits for endgame scoring. And, of course, some free moves spoil capture opportunities.

Captures are another important strategic consideration. Given that taking the closest free move is a frequent optimal play, we will now restrict our attention to states without free moves. There are 4,432,826 (44.37%) non-free-move states. Of these, 3,777,047 (85.21%) offer capture move(s). Of these non-free-move states with potential capture(s), 3,212,373 (85.05%) have an optimal capture move. Thus, if there is no free move, there is a 72.47% chance that a capture move is optimal.

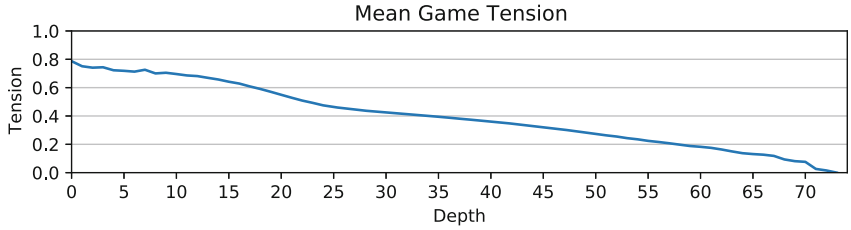


Fig. 3. Game tension by depth.

This brings us to consideration of the frequencies with which play pits are found to be optimal:

Pit	States Optimal	Percent
1	3568534	35.72%
2	2945674	29.48%
3	2640664	26.43%
4	2527830	25.30%
5	2333925	23.36%
6	1811551	18.13%

As we can see, pits closer to a player's score pit are indeed more likely to be optimal plays. The probability of optimality of the closest pit is highest and that probability decreases monotonically with each farther pit to slightly over half of the probability in the farthest pit. This is to be expected as close free moves are the most frequent and easy to set up, and one often clears the farthest pits in response to capture threats.

Note that our FOGT data fully supports the wisdom of the node-ordering heuristic of [5]: After a stored transposition table move (if available), consider free moves, then captures, then other moves, preferring moves closer to the score pit within each category. This should, in turn, guide our strategic prioritization of move considerations in play.

4 Modeling Optimal Play for Diverse States

We are interested in using such data to build predictors of expected game value as well as optimal move prediction by pit. While examining optimal game states of Kalah led to some interesting results, predictive power of models trained on only FOGT data did not generalize well to non-fair states. Thus, to better model optimal play, we generated a different training data set that included unfair (non-0 value) Kalah states.

In order to efficiently generate diverse state data, we randomly generated distinct starting nodes by randomly scoring 5 pieces from different pits. We then analyzed optimal play for each node using a breadth-first-traversal limited to 100 nodes per depth, similar to a beam search. This process was repeated for different randomized initial states until 9,449,283 states were generated. After removing depth and player number features and adding the aforementioned engineered features from the point of view of the current player, the number of unique states was 6,151,746.

4.1 Model Performance Summary

For all prediction, we used a 50–50 train-test split. For prediction of the negamax state game value `game_val`, we trained a number of different models: scikit-learn's `LinearRegression`, `DecisionTreeRegressor`, and `RandomForestRegressor`, as well as the CatBoost `CatBoostRegressor`. `DecisionTreeRegressor` had a `min_samples_split=500000`, `RandomForestRegressor` had hyperparameters `n_estimators=100`, and `CatBoostRegressor` had

Model	MSE	R^2
Linear regression	23.69	0.81
Decision tree	43.84	0.63
Random forest	5.70	0.95
CatBoost	4.37	0.96

Fig. 4. Regression model performance summary

`iterations=30`, `learning_rate=0.9`, and `depth=16`. A performance summary is given in Fig. 4.

Similarly, we built classification models for each play pit predicting `optimal_`n, i.e. whether the pit is an optimal play using scikit-learn's `LogisticRegression` and `RandomForestClassifier`, as well as the `CatBoostClassifier`.

Since empty play pits are illegal for play and trivial to predict as not optimal, our performance measurements per pit are for only states where those pits are not empty. Accuracy and log loss performance results are given per pit number in Figs. 5 and 6, respectively. We also include a "Base predictor" row for a probabilistic prediction based on the frequency of non-empty pit states for which that pit is optimal.

Model	6	5	4	3	2	1
Base predictor	.68	.68	.67	.65	.57	.58
Logistic regression	.80	.80	.78	.77	.77	.86
Random forest	.87	.86	.86	.85	.86	.89
CatBoost	.89	.88	.88	.87	.87	.92

Fig. 5. Classification accuracies per non-empty pit model

Model	6	5	4	3	2	1
Base predictor	.63	.63	.63	.65	.68	.68
Logistic regression	.43	.44	.47	.48	.46	.31
Random forest	.31	.32	.33	.33	.32	.23
CatBoost	.27	.28	.28	.29	.29	.21

Fig. 6. Classification log loss per non-empty pit model

4.2 SHAP Beeswarm Figure Interpretation

Using the `shap` package, we generated beeswarm plots in Figs. 7 and 8, to visualize SHapley Additive exPlanation (SHAP) values [7] for our CatBoost `game_val` and `optimal_1` prediction models, respectively. SHAP values "assign each feature an importance value for a particular prediction." [6] Each instance in our dataset represents one point for each horizontally visualized feature. The subset of features included are the most significant to the model prediction according to SHAP values, with higher features being most significant. The horizontal position of each point is determined by

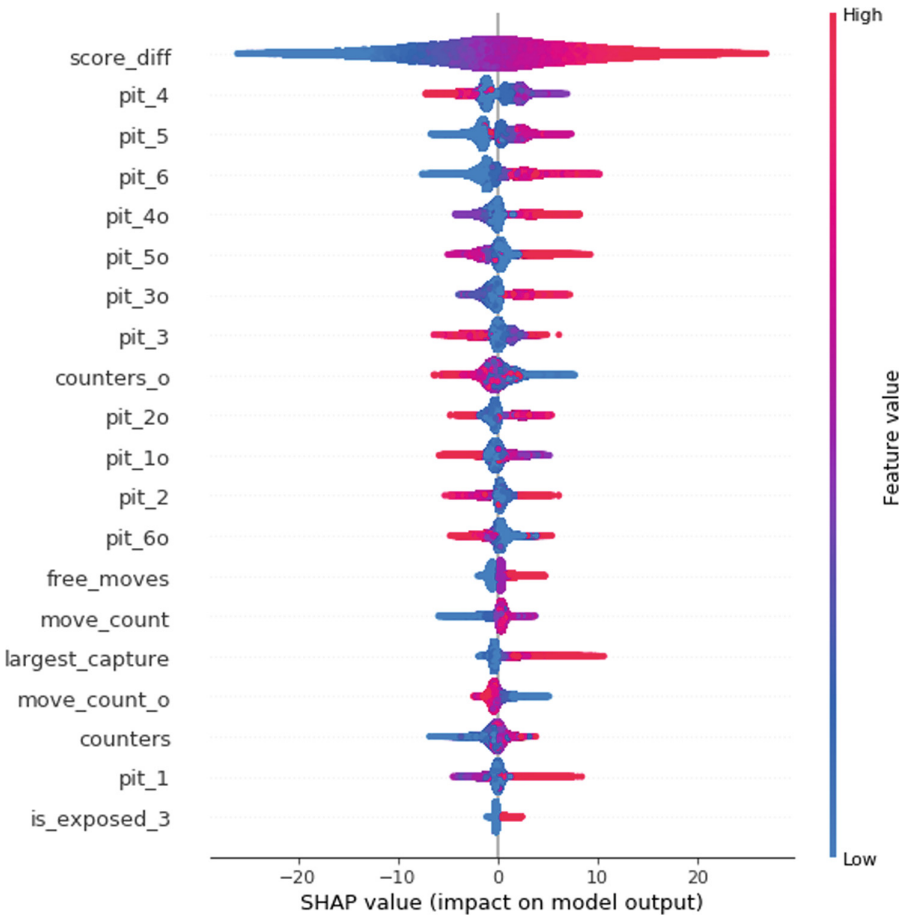


Fig. 7. SHAP beeswarm plot for CatBoost model predicting `game_val`

its SHAP value for that feature, and points are bunched more thickly along each feature row to show density. Color of a dot represent the original feature value for that instance, with red/blue indicating high/low values, respectively.

For example, Fig. 7 shows us first and foremost that `score_diff`, i.e. the current score difference is the strongest positively correlating predictor for the final score different, i.e. the game value. Looking at the next five most significant features, we see that number of pieces in pits 4, 5, 6, $\overline{4}$, $\overline{5}$, $\overline{3}$, and 3 are all strong predictors of game value, with more pieces in those pits tending to positively correlate with higher game values with the exceptions of pits 4 and 3. More pieces in opponent pits (`counters_o`) negatively correlate with game value.

Turning our attention to Fig. 8, we see that the single most significant predictor for pit 1 being an optimal play is the fact that it is the closest free move, i.e. a free move. The number of pieces in pit 1 is the next most significant feature, with more pieces negatively correlating with optimality. Higher numbers of free moves positively correlate, as taking a farther free move first would add a piece to pit 1, ruining the free

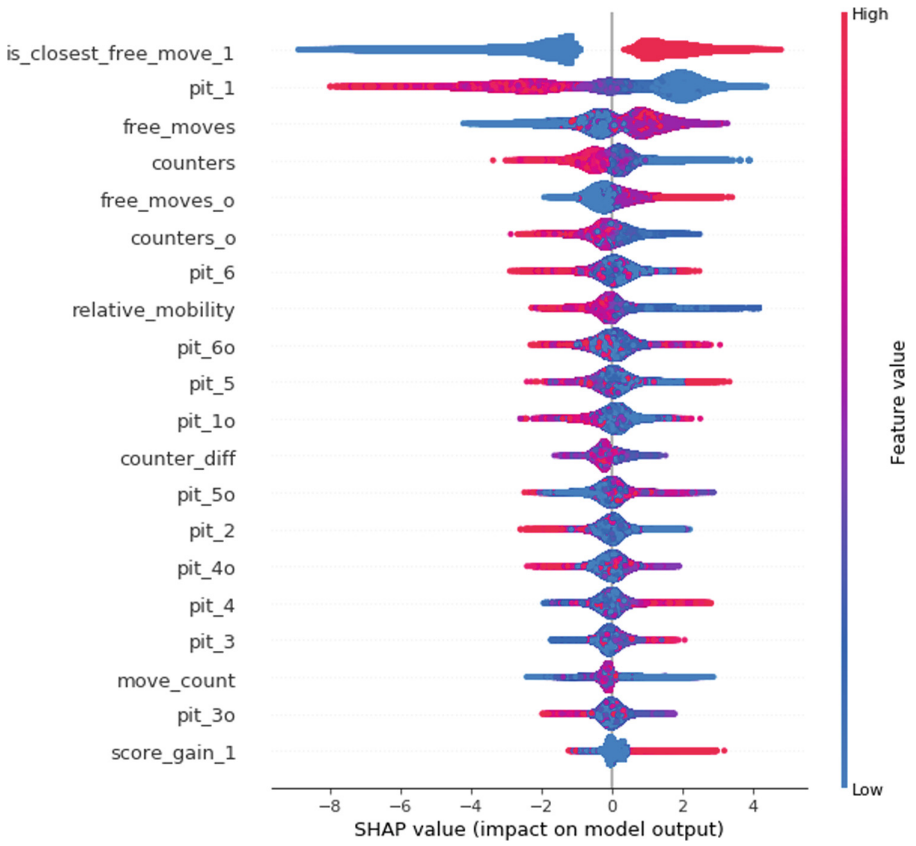


Fig. 8. SHAP beeswarm plot for CatBoost model predicting `optimal_1`

move option. A greater number of counters for either player negatively contributes to prediction of optimality, but the number of opponent free moves positively contributes.

We lack space here to share all of the insights that can be drawn from such plots, but such explanatory visualizations can be helpful for forming hypotheses of strategic significance in play.

5 Future Work

We believe there are interesting future work possibilities in seeking to approximate optimal play under computational constraints. Whereas one can easily compute any optimal play on current retail computers within half a minute using our 1.16 GB endgame database and search optimization, we are interested in what is possible for web and mobile apps with limited computational resources.

Our best predictive models of optimal play are memory intensive, yet we believe the necessity of strict memory and time limitations can spur further insight to Kalah AI strategy. It was surprising what Óscar Toledo Gutiérrez was able to accomplish in 1 KB

of JavaScript when creating his winning chess engine entry for the JS1K contest [11]. We believe it would be interesting to see what might be possible with various code and memory constraints. Limitations breed creative problem-solving.

Also recall that we expect that time-management emphasis on early game play should be rewarded with greater blunder avoidance. A study of time-management systems for this game could test that conjecture.

Finally, given that many play Kalah without zero-captures, one might repeat this research with that rule variation. One would expect different fair initial states, but how characteristics of optimal gameplay might change is an interesting open question.

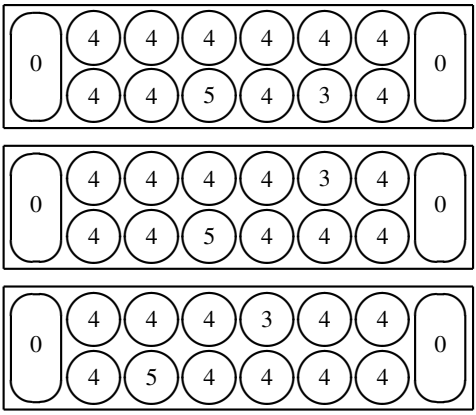
6 Conclusions

In this article, we have offered three primary contributions. First, we have computed fair starting states for Kalah (a.k.a. Mancala) that should improve the game experience for future generations. Second, we have analyzed the fair optimal game trees to observe characteristics of perfect play. This affirmed the node-ordering heuristic of [5], prioritizing free moves, then capture moves, with a preference towards moves closer to the score pit. Finally, we collected optimal play data from a diverse set of optimal and suboptimal game states, yielding models with strength for predicting game values and optimal moves. These yielded further insights into important features for AI gameplay heuristics.

7 Appendix: FairKalah Boards

7.1 Moving 1 Piece

These are all existing FairKalah boards where one moves exactly one piece from 4-pieces-per-pit initial conditions.



7.2 Moving 2 Pieces

All 251 existing FairKalah boards where one moves exactly two pieces from 4-pieces-per-pit initial conditions are available in SVG image form and as CSV data from the FairKalah website [8]. In the CSV data, comma-separated integers on each line indicate the number of pieces in each pit (including scoring pits) starting from the first (i.e. "south") player's leftmost play pit and proceeding counterclockwise.

References

1. Baier, H., Winands, M.H.M.: Time management for Monte-Carlo tree search in go. In: van den Herik, H.J., Plaat, A. (eds.) ACG 2011. LNCS, vol. 7168, pp. 39–51. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31866-5_4
2. Browne, C.: Tension in puzzles. Game Puzzle Design J. **3**(2), 71–78 (2017)
3. Champion, W.J.: Game counter (US Patent US2720362A, 1955–10-11). https://patents.google.com/patent/US2720362A/en
4. van Horssen, J.J.: Move selection in MTD(f). ICGA J. **41**, 1–9 (2019). https://doi.org/10.3233/ICG-190096
5. Irving, G., Donkers, J., Uiterwijk, J.: Solving Kalah. ICGA J. **23**(3), 139–147 (2000). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.7870
6. Lundberg, S.: beeswarm plot - SHAP latest documentation (2022). https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/beeswarm.html. Accessed: 2022-09-03
7. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017). https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
8. Neller, T.W., Neller, T.C.: FairKalah: Fair Mancala. http://cs.gettysburg.edu/~tneller/games/fairkalah/. Accessed: 2022-08-29
9. Plaat, A., Schaeffer, J., Pijls, W., de Bruin, A.: Best-first fixed-depth minimax algorithms. Artif. Intell. **87**(1), 255–293 (1996). https://doi.org/10.1016/0004-3702(95)00126-3
10. Russ, L.: The Complete Mancala Games Book: How To Play the World's Oldest Board Games. Marlowe & Company (2000). https://books.google.com/books?id=BA9yHQAACAAJ
11. Toledo Gutiérrez, O.: Chess programs (2022). https://nanochess.org/chess.html. Accessed: 2022-09-03