Mastering the Royal Game of Ur

Newton Kwan and Alexander Hartemink

***Abstract*— Achieving above human level performance on a task once considered only possible by humans is a hallmark of artificial intelligence research. There are still many human tasks that are too complex for machines to learn, so we look to simpler tasks like games – activities with heuristics that often capture an important part of human ingenuity – to understand learning. The Royal Game of Ur is a game that involves both elements of strategy and chance. To understand learning in an environment with random elements, we compare performance and learning strategies between humans, agents playing randomly, agents using a lookup table, and agents trained to play using reinforcement learning.**

I. INTRODUCTION

The Royal Game of Ur is a simple model for how we make decisions in the world. If we have good intuition, there is often a clear decision as to which move to make. Other times, we have a choice between two or three fairly similar moves. Sometimes, you are forced to make an undesirable move. And every so often, you cannot make a move at all. The current situation or gamestate that one finds themselves in is determined by a history of boardstates, dice rolls, and moves. We explore the process of learning to make optimal choices given a current gamestate with the ultimate goal of winning the game.

If we had perfect knowledge of our opponent's moves and the outcome of each die roll, we could chose the move that brings us closer to winning the game. But, perfect knowledge is often not possible. However, even with optimal play, a few unlucky rolls can dramatically reduce the probability of winning, even for the most skilled player. To counteract this, strategies are developed to reduce the negative side effects of chance. Understanding how an agent reacts to environments with randomness is crucial to understanding how we weigh our own decisions.

In this paper, we explore human strategies and agent strategies for the Royal Game of Ur. We will develop an agent that plays the game randomly, an agent that plays the game using a look-up table, and an agent that plays the game using a strategy learned through reinforcement learning. By comparing win percentages and strategies, we can better understand decision making and how an agent learns to win.

II. GOALS

We hope to accomplish the following things

- Develop human strategies
- Create a lookup table for optimal play
- Understand how closely human play compares to optimal play
- Use open source material to train a reinforcement learning agent that performs as well as the optimal strategy without any prior knowledge of strategies

III. THE ROYAL GAME OF UR

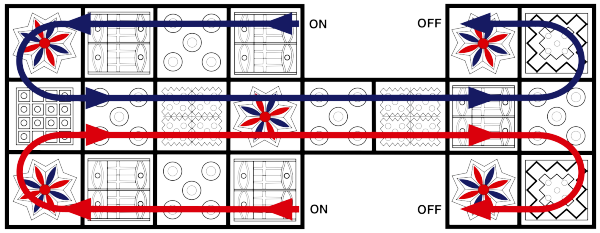

Fig. 1. Game board and the path to follow. Image by J.P Smith used with permission.

A. The Rules

- The game is played with two players.
- Dice rolls are from 0-4.
- The goal is to advance all of your pieces to the end.
- Two of the same player's pieces may not occupy the same location.
- When the opponent lands on the opponents piece, the piece previously occupying the spot is sent home, except when the piece is on a flower
- A flower is a safe spot and allows the player who lands on it an extra roll the same turn that the player lands on the flower. These are the blue and red shapes in Fig. 1
- You must follow the designated direction (you cannot move backwards)
- You must move a piece if you are capable of moving (no passes)

For a full list of rules, see the Appendix on board representation.

B. Gameplay

Each player starts with six pieces of his respective color off the board. There is no set rule for who goes first, so one can either flip a coin or simply decide. Once the player who will go first is decided, the player will roll the dice and advance a token to a legal position. Then, the players take turns rolling the die and moving pieces to whichever legal move they choose following the rules. The game is over when one player successfully moves all six of his or her colored pieces off the board. That player is declared the winner.

IV. AGENTS

We will have three agents with three different play styles: random, optimal, and trained through reinforcement learning

A. Random

This agent generates a list of all legal moves from its current gamestate and chooses its next move based uniformly from this list of possible moves.

B. Optimal

This agent uses a look-up table to determine the next best move. The next best move in essence is determined by choosing the move that maximizes the likelihood of transitioning into any of the possible winning boardstates. The process of creating the look-up table can be found in the Appendix.

C. Reinforcement Learning

We train our last agent using reinforcement learning, a paradigm of machine learning. Reinforcement learning can be thought of as balancing exploration and exploitation. The basic intuition is this: Try something new. Add randomness to your actions and compare your results to your expectations. If your result exceeds your expectation, then change your parameters to take those actions in the future. Try out the new parameters. If you like it, do more of it in the future.

In order to train our agent using reinforcement learning, we will use Simple AlphaGo Zero (Thakoor, Nair, Jhunjhunwala 2017), a simplified version of the the algorithm used by AlphaGo Zero (Silver 2017 et. al). Simple Alpha Go is an algorithm that can be used for other two person adversarial games, like the Royal Game of Ur. More details about the AlphaGo Zero algorithm are found in the Appendix.

We will compare how well the RL agent plays against random agent, the optimal agent, and the human. If it fairs well against the optimal agent, we can make guesses about performance.

V. PREDICTED OUTCOMES

The table below shows our hypothesis of the matches. We will simulate 10,000 games where the first player to go is decided using a coin flip.

Win %	Random	Optimal	RL
Random	50%	$\ll$ 50%	$\ll$ 50%
Optimal	$\gg$ 50%	50%	50%
RL	$\gg$ 50%	50%	50%

TABLE I

LEFT COLUMN AGENT'S PREDICTED WIN % VS. THE TOP ROW AGENTS

We expect each agent to have a 50% win rate against itself. If agent that learned through reinforcement learning is as good as the agent playing optimally, we expect 50% win rate between the optimal agent and the RL agent. Both the optimal play agent and the reinforcement learning agent are expected to win $\gg$ 50% of the time against the random agent. It is not feasible for a human to play 10,000 games against these agents, so we will play a few dozen games against each agent to get a rough estimate of human performance against these agents. We predict that the human will win $\gg$ 50 % of the time against the random agent, and do not have clear estimates on how well a human will perform against the optimal or RL agent. At best, the human will win 50 % of the time against the optimal or the RL agent.

Win %	Random	Optimal	RL
Human	$\gg$ 50 %	$\leq$ 50 %	$\leq$ 50 %

TABLE II

PREDICTED HUMAN WIN % VS. THE TOP ROW AGENTS

VI. WINNING STRATEGIES

This section will describe human strategies, optimal agent strategies, and compare the two.

A. Human intuition

It is clear after even a few games that there are strategies a player can use to increase the likelihood of winning. A few of the key human strategies are outlined below.

1) Center flower control: Center flower control involves landing on the center flower tile and staying on it a long as possible. This prevents your piece from being taken, eliminates the possibility of your opponent landing on the center flower, and gives you the opportunity to take any piece in front of you if appropriate. In other games, a piece like the center flower may colloquially be referred to as a *power position*. In other words, a power position is boardstate that gives the player more control of the game by allowing you to make better moves and/or decreasing the number of good moves for your opponent.

2) Loading the bases: Like in baseball, loading the bases allows a team to build up small progress and capitalize on one great hit. In the Royal Game of Ur, loading the bases involves setting up pieces in the first four spots of the respective player's territory, keeping the player's pieces safe and waiting for your opponent to make the first move into the shared territory. This maximizes the likelihood that the player whose bases are loaded will be able to land on an opponent pieces in the shared territory and send their piece home.

3) Running for the hills: The saying goes that when you're in the danger zone, you should run for the hills. Whenever a player has pieces in the shared territory, it is advantageous to get to the end of the shared territory as quickly as possible. If a piece is in the shared territory (unless the player's piece is on the center flower tile), the piece is susceptible to being sent home. The closer a piece is to the beginning of the shared territory, the more dangerous the tile.

4) Offloading territory flower tiles: The benefits of landing on a territory flower tile only include the extra roll because the territory flower tiles are in the players' respective safe zones and can only be used by that player. In order to maximize the benefits of the territory flower tiles, a player should try to land on the a territory flower tile (for the extra

roll) and then move off quickly so that their other pieces can utilize the territory flower tile.

B. Optimal agent

We will simulate games with the optimal agent playing against itself and record metrics such as how often the agent chooses to do certain moves and how long the agent stays on certain tiles. For example, comparing the number of turns the agent will stay on the center flower tile after landing on it can tell us how desirable the tile position is, or how quickly the agent decides to go through the shared territory tells us something about how undesirable it is to stay in the the shared tile territory. It will be exciting to see the agent learn human strategies, but arguably more exciting to see strategies that are better than the ones we learned through human play.

VII. CONCLUSIONS

Decision making and learning in a landscape of uncertainty both in machines and in humans will continue to be an area of interest as progress is made towards more powerful AI systems. Although some of the results and goals of the paper are still in progress, mastering the Royal Game of Ur continues to be a useful and instructive means to understanding machine learning. Finding a way to unpackage the idea of human intuition and how to train agents will continue to be of interest into the future.

APPENDIX

A. Boardstate representation

We represent boardstates in a 3×8 matrix from the perspective of Fig. 1 above, following the same move path. The opponent's viewpoint will require only two rotations, one about the axis parallel to the long middle strip dividing the top and bottom of the board and one about the axis dividing the left and right side of the board.

We represent a boardstate $\mathbf{b}$ in the following way

$$\mathbf{b} = \begin{bmatrix} \mathbf{b_{00}} & b_{01} & b_{02} & b_{03} & \mathbf{b_{04}} & \mathbf{b_{05}} & \mathbf{b_{06}} & b_{07} \\ b_{10} & b_{11} & b_{12} & \mathbf{b_{13}} & b_{14} & b_{15} & b_{16} & b_{17} \\ \mathbf{b_{20}} & b_{21} & b_{22} & b_{23} & \mathbf{b_{24}} & \mathbf{b_{25}} & \mathbf{b_{26}} & b_{27} \end{bmatrix}$$

where $\mathbf{b} \in \mathbb{R}^{3\times 8}$. All elements of $\mathbf{b}$ are specific real numbers that each represent different tilestates of the board. The bold elements are still real numbers, but are bold to show that they have special properties. Following the specific rules of the Royal Game of Ur, B is the set of all possible board states.

$$B = \{\mathbf{b^{(1)}}, ..., \mathbf{b^{(n)}}\}$$

where n is the number of legal boardstates. For example, the tile in the 1st row and 2nd column of the 146th boardstate would be represented as $b_{01}^{(146)}$. The numbering scheme and a full set of rules are in the tables below:

Number	Representation
0	Empty tile
1-6	One-six black pieces
7-12	One-six white pieces

TABLE III

REAL NUMBER REPRESENTATIONS OF TILESTATES

Element	Name	Rule
b_{24}	Black start	Can hold 0-6
b_{04}	White start	Can hold 0, 7-12
b_{25}	Black end	Can hold 0-6
b_{05}	White end	Can hold 0, 7-12
b_{20}, b_{26}	Black Flower	Roll again
b_{00}, b_{06}	White Flower	Roll again
b_{13}	Center Flower	Roll again and safe

TABLE IV

PROPERTIES AND RULES OF SPECIAL TILES

1) Shared tiles: Shared tiles are in the middle row and can be occupied by one white piece or one black piece at a time. With the exception of b_{13}, shared tiles are also *dangerous tiles*. For example, if a white piece is occupying a shared tile and black decides to legally move a piece to the same tile, the white piece is sent to the start tile and black will occupy the tile, vice versa. These tiles are *dangerous tiles*.

2) White / Black tiles: The top row of tiles in the White territory are considered White tiles and the bottom row of tiles in the Black territory are considered Black tiles because only white and black pieces can occupy these tiles respectively. This property is a natural corollary from the way that a player can move pieces on the board, and in some sense, these tiles are considered *safe tiles*.

3) Flower tiles: If a player lands on a Flower tile, the player receives one additional die roll. Flower tiles are also *safe tiles* because an opponent cannot land on a Flower tile occupied by the other player. Specifically, the only Flower tile where this rule applies is tile $\mathbf{b_{13}}$, which is also called the Center Flower tile. Although it may not be clear yet, utilizing this center tile is part of a winning strategy. Black and White Flower tiles are Flower tiles found in the Black and White territory, respectively.

4) Start tiles: The Start tiles, $\mathbf{b_{24}}$ and $\mathbf{b_{04}}$, begin with six black pieces and six white pieces at the start of the game. Start tiles can hold from zero to six pieces and is where pieces that are sent home go.

5) End tiles: The End tiles, $\mathbf{b_{25}}$ and $\mathbf{b_{05}}$, can hold zero to six black and white pieces respectively. The first person to have six pieces of their respective color in their respective end tile is the winner of the game.

B. Creating the Lookup Table

Each player has a set of winning boardstates W,

$$W = \{\mathbf{w^{(1)}}, ..., \mathbf{w^{(k)}}\}$$

where k is the number of possible winning boardstates. The transition probabilities $\mathbf{p}$ for each winning boardstate $\mathbf{w^{(i)}}$ conditioned on $\mathbf{s}$ is given by

$$\mathbf{p} = \begin{bmatrix} P(\mathbf{w^{(1)}}|\mathbf{s}) \\ \vdots \\ P(\mathbf{w^{(k)}}|\mathbf{s}) \end{bmatrix}$$

where $\mathbf{s} = (\mathbf{b}, d, C)$ is called the gamestate where $\mathbf{b}$ is the boardstate, $d \in D$ is the die roll, and C is the color to play which has a constant value of either 1 if black is to play or 0 if white is to play. D is the set of possible rolls

$$D = \{0, 1, 2, 3, 4\}$$

with probability distribution $\mathbf{p_D}$

$$\mathbf{p_D} = \begin{bmatrix} P(d=0) \\ P(d=1) \\ P(d=2) \\ P(d=3) \\ P(d=4) \end{bmatrix} = \begin{bmatrix} 1/16 \\ 4/16 \\ 6/16 \\ 4/16 \\ 1/16 \end{bmatrix}$$

At a high level, the look-up table will input a gamestate and output the optimal move. As the game progresses, each player should pick the moves that maximize the probability of transitioning into a winning boardstate. Specifically for black, black wants to maximize the likelihood of having six black pieces in the black end tile.

$$\max(P(\mathbf{b_{25}} = 6))$$

Since black wins in any of the winning boardstates, we want to maximize the sum of the transition probabilities into each the winning boardstates. In order to generate the lookup table, we use a function f that maps a gamestate $\mathbf{s}$ to a boardstate $\mathbf{b} \in M$ (the set of next legal boardstates) that maximizes the likelihood that the current player will transition into any of the winning boardstates. For each $\mathbf{b} \in M$, calculate

$$f(\mathbf{s}, M) = \underset{\mathbf{b}}{\operatorname{argmax}} \sum_{i=1}^{k} P(w^{(i)}|\mathbf{s}, \mathbf{b}) \tag{1}$$

and choose the boardstate that maximizes the probability of transitioning into a winning boardstate.

As an example, imagine that black has just won a game. Tracing back the steps from the winning boardstate to the initial gamestate, we can create a set of gamestates that occurred one after the other, starting from the initial gamestate to the winning boardstate. We call this particular set the history of the game

$$H = \{(\mathbf{s_1}), ..., (\mathbf{s_{t-1}}), (\mathbf{b_t} = \mathbf{w^{(i)}}, C)\}$$

where t is the number of turns in the game and j is the specific winning boardstate number. Note that the last element of the history only contains the winning boardstate and the player whose turn it is at time step t.

Backtracking one time step from a winning boardstate, we can calculate the probability of transitioning into the winning boardstate before and after the die roll. During each time step, the probability of winning is updated twice, once before and once after the die roll. The probability of transitioning into the winning boardstate given the die roll is

$$P(\mathbf{b_t}|\mathbf{s_{t-1}}) = \begin{cases} 1, \text{if } f(\mathbf{s}, M) = \mathbf{b_t} \\ 0, \text{otherwise} \end{cases} \tag{2}$$

If we want the win probability before rolling the die, the probability of transitioning into the winning boardstate is

$$P(\mathbf{b_t}|\mathbf{b_{t-1}}, C) = P(\mathbf{b_t}|\mathbf{s_{t-1}})P(d_{t-1} = a) \tag{3}$$

where a is the number of tiles away the black's final piece was from the black end tile. In the case where we are in the penultimate gamestate, we always want to pick the legal move that brings us to the winning boardstate.

In general, we can calculate the transition probabilities from any gamestate to any boardstate (after die roll) or any boardstate to any boardstate (before die roll). From these calculations, a look-up table can be generated using dynamic programming and utilizing equations (1), (2), and (3) as the updates.

C. Simple AlphaGo Zero

A neural network f_θ takes input gamestate $\mathbf{s}$ and outputs two values: a continuous real value of the boardstate $v_\theta(\mathbf{s}) \in [-1, 1]$ and a policy $\mathbf{p}_\theta(\mathbf{s})$ that is the probability vector over all possible actions.

At the end of each iteration of self play, the neural network is given training data in the form $(\mathbf{s_t}, \pi_\mathbf{t}, z_t)$, where $\pi_\mathbf{t}$ is an estimate of the policy at state $\mathbf{s_t}$ (this is found through Monte Carlo Tree Search (MCTS)) and $z_t \in \{-1, 1\}$ is the final outcome of the game from the perspective of the player at $\mathbf{s_t}$ (+1 if the player wins and -1 if the player loses). The neural network is then trained to minimize the loss function:

$$l = \sum_t (v_\theta(\mathbf{s}) - z_t)^2 - \pi_\mathbf{t} \cdot \log(\mathbf{p}(\mathbf{s_t}))$$

The idea is that the neural network will learn what gamestates $\mathbf{s_t}$ lead to wins, and learning the policy will give a good estimate of which the best move to take for a given gamestate $\mathbf{s_t}$.

ACKNOWLEDGMENT

I'd like to thank Alex for his help in making this project possible. Without his insightful suggestions and encouraging conversations, this project would not be where it is today. I'd like to thank the team at DeepMind for their inspiring work on AlphaGo Zero, the team of graduate students at Stanford and their implementation of AlphaGo Zero, and the many friends and colleagues with whom I played the Royal Game of Ur.

REFERENCES

[1] Silver, D. et al. Mastering the game of Go without human knowledge, Nature, 550, 2017, pp. 354-359.

[2] Thakoor S., Nair S., Jhunjhunwala M. Learning to Play Othello Without Human Knowledge, 2017.