Optimal Play of the All Yellow Zombie Dice Game

Todd W. Neller(✉), John C. Llano, Minh Q. Vu Dinh, and Clifton G. M. Presser

Department of Computer Science, Gettysburg College, Gettysburg, USA
tneller@gettysburg.edu

Abstract. In this paper, we solve and visualize optimal play for All Yellow Zombie Dice, a simplification of the Zombie Dice jeopardy dice game by Steve Jackson [1] where we assume that all dice have the same outcome distribution. We present a spectrum of All Yellow Zombie Dice human-playable strategies that trade off greater play complexity for better performance, and collectively clarify key considerations for excellent play.

1 Introduction

Zombie Dice is a dice game first published in 2010 by Steve Jackson [1]. It is a jeopardy dice game [4, Ch. 6] in the Ten Thousand dice game family [2]. In this paper, we analyze a simplified variant, All Yellow Zombie Dice, computing optimal play as well as providing additional insights to gameplay.

We begin by describing the rules of Zombie Dice, and a variant thereof, All Yellow Zombie Dice. We define 2-player optimality equations for All Yellow Zombie Dice and our method for solving them. After visualizing the optimal play policy, we share observations on the optimal roll/hold boundary. We then present an array of human-playable policies we have devised along with their performances against the optimal policy. The policies demonstrate different design trade-offs of greater complexity for greater win rates, and highlight key play policy considerations. Finally, we discuss future work and summarize our conclusions.

2 Rules

Zombie Dice (ZD) is a dice game for two or more players using 13 nonstandard six-sided dice described in Fig. 1.

A *turn* consists of a sequence of player dice rolls where rolled brains and shotguns are set aside. The turn ends when, after rolling, the player either decides to *hold* (i.e. stop rolling) and score the total number of brains rolled, or has rolled three or more shotguns, ending the turn and scoring 0 points.

At the beginning of the turn, three of the supply of 13 dice are drawn at random and rolled. Any rolled brains and shotguns are then set aside. If 3 or

M. Hartisch et al. (Eds.): CG 2024, LNCS 15550, pp. 197–206, 2025.
https://doi.org/10.1007/978-3-031-86585-5_16

Color	Number of Dice	Brain Sides	Shotgun Sides	Footprint Sides
Green	6	3	1	2
Yellow	4	2	2	2
Red	3	1	3	2

Fig. 1. Zombie Dice outcome distributions

more shotguns are set aside, the turn ends scoring 0 points. Otherwise, the player chooses either to hold, scoring the number of brains and ending their turn, or to roll again. To roll again, dice are drawn at random and added to any rolled footprints until there are three dice to roll. (If there are no dice to draw, keep track of the number of brains set aside, and add all rolled brain dice back into the dice supply, and continue drawing dice at random.) Then those three dice are rolled, any brains and shotguns are again set aside, and we repeat the process described above.

A *round* consists of each player taking one turn in sequence. When a round ends with any player having 13 or more points, a player having the most points wins. If two or more players are tied with the most points (13 or more), another round is played between those players only. In this paper, we will focus on the two-player ZD game, so we can say that a 2-player game ends when a round concludes with a single winner having the most points with 13 or more points.

We will denote red, yellow, and green dice as R, Y, and G, respectively, with brain, shotgun, and footprint rolls denoted as B, S, and F, respectively. We denote a brain roll of a green die as BG. Consider this example round of two-player play:

- Player 1 draws G, G, and Y dice and rolls FG, FG, and SY. SY is set aside. We have zero (fewer than three) S set aside, so the player can hold (scoring zero B) or roll. The player chooses to roll, drawing a third random die to join the two FG dice. A G is drawn and the three G are rolled as BG, SG, and FG. BG and SG are set aside. We have two S set aside, so the player can hold (scoring one B) or roll. The player chooses to roll, drawing two R dice at random, and rerolling FG with these to get BG, SR and SR. All three B/S dice are set aside, totaling four S. With three or more S, the turn ends with no score change. (All dice are returned to the supply at the end of a turn.)
- Player 2 draws three G dice and rolls BG, BG, and FG. The two BG dice are set aside. Player 2 with no S set aside chooses to roll, draws a G and Y, and rerolls the FG with these to get BG, BY, and FG. The two B dice are set aside for a total of four B dice. Player 2 with no S chooses to roll, draws Y and R dice, and rerolls the FG with these to get BY, SG, and SR. These are all set aside for a total of five B and two S dice set aside. Although player 2 could roll again with less than three S set aside, player 2 decides to hold, scoring five points, one for each B set aside, ending the round.

The game thus consists of roll/hold risk assessments in a race to achieve a unique top score of 13 or more points, playing additional tie-breaker rounds with tied leaders as necessary. Given the player scores, the numbers of colored B and S dice set aside, and the current locations of non-S dice of different colors, should the current player roll or hold so as to maximize the probability of winning?

2.1 All Yellow Zombie Dice (AYZD)

It can often be insightful to first analyze a simplification of a game. In this case, we create a 2-player variant we call All Yellow Zombie Dice (AYZD), where all 13 dice are yellow with two B, two S, and two F sides. The three roll outcomes are equiprobable for all dice and we need not include dice colors nor their locations as part of our state description for this simplified game.

3 AYZD Optimality Equations and Solution Method

We here define optimality equations for the AYZD two-player game.

Nonterminal states are described as the 5-tuple (p, i, j, b, s), where p is the current player number (1 or 2), i is the current player score, j is the opponent score, b is the turn total (number of brains set aside), and s is the number of rolled shotguns set aside. $P(p, i, j, b, s)$ will denote the probability of player p winning in state (p, i, j, b, s) under the assumption of optimal play, i.e. each player plays so as to maximize one's own expected win probability.

Terminal states consist of a player 1 win or player 2 win. (Draws are not allowed, as tie-breaker rounds are mandated.) Player 1 wins at the beginning of their turn ($p = 1, b = 0$) when player 1 has achieved the goal score ($i \geq g$ where $g = 13$) and player 2 ended their turn with a lesser score ($j < i$). Player 2 wins on their turn ($p = 2$) when player 2's score plus their turn total exceed both the goal score and player 1's score ($i \geq 13$ and $i > j$), at which point player 2 can and should hold, winning the game.

Let $P_{\text{roll}}(b, s)$ be the probability of rolling b brains and s shotguns (and thus $3 - b - s$ footprints) on a roll of 3 dice:

$$P_{\text{roll}}(b, s) = \frac{\binom{3}{b}\binom{3-b}{s}}{3^3} = \frac{2}{9b!s!(3-b-s)!}$$

The probability of winning with a roll $P_{\text{roll}}(p, i, j, b, s)$ under the assumption of optimal play thereafter is:

$$\begin{aligned} P_{\text{roll}}(p, i, j, b, s) = & \textstyle\sum_{s^+=0}^{2-s} \sum_{b^+=0}^{3-s^+} (P_{\text{roll}}(b^+, s^+) P(p, i, j, b + b^+, s + s^+)) \\ & + \textstyle\sum_{s^+=3-s}^{3} \sum_{b^+=0}^{3-s^+} (P_{\text{roll}}(b^+, s^+)(1 - P(3 - p, j, i, 0, 0))) \end{aligned}$$

where b^+ and s^+ denote the number of additional brains and shotguns rolled.

The probability of winning with a hold $P_{\text{hold}}(p, i, j, b, s)$ under the assumption of optimal play thereafter is:

$$P_{\text{hold}}(p, i, j, b, s) = (1 - P(3 - p, j, i + b, 0,\ 0))$$

Then the probability of winning $P(p, i, j, b, s)$ under the assumption of optimal play is:

$$P(p, i, j, b, s) = \max(P_{\text{roll}}(p, i, j, b, s), P_{\text{hold}}(p, i, j, b,\ \text{s}))$$

Players can tie at or above the goal, requiring a tiebreaker round. Players can tie within a tiebreaker round as well, requiring another tiebreaker round. Since there is no limit to tiebreaker rounds and no limit to the turn total, we must create an artificial upper limit for computational purposes. We have chosen twice the goal score ($M = 2g = 26$) as this bound, and observe that computational results do not change for $M = 3g$, $4g$, or $5g$, assuring us that we capture optimal play behavior within such bounds.

Having bounded our nonterminal state space representation such that $p \in \{1, 2\}, 0 \leq i, j, b \leq M, s \in \{0, 1, 2\}$, we apply value iteration as in [5] until the maximum probability change of an iteration is less than $\epsilon = 10^{-14}$.

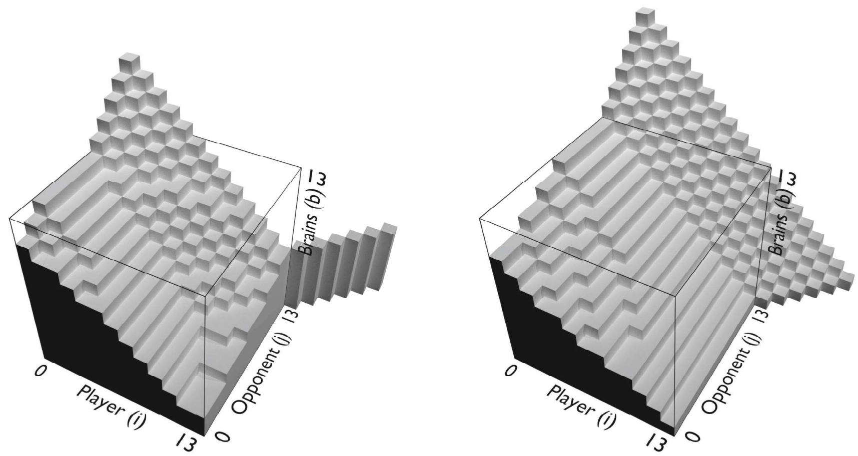
4 AYZD Optimal Policy

The optimal roll/hold boundaries of AYZD are shown in Fig. 2. Each subfigure depicts a 3-dimensional (i, j, b) roll/hold boundary for each possible pair of player p and rolled s. Axes are player score i, opponent score j, and turn total b. Given a current state inside or outside of the appropriate solid, an optimal player should roll or hold, respectively.

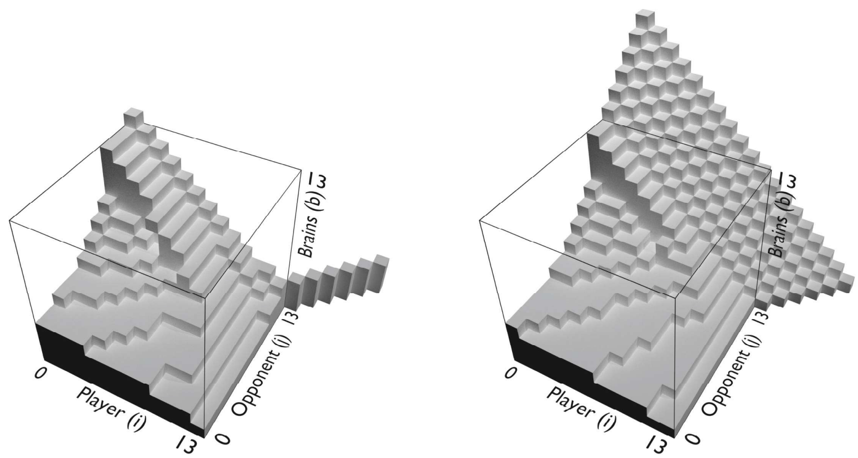
We first observe a few expected similarities between these roll/hold solids. First, the $i+b = 13$ diagonal plane indicating a rolling for the goal score appears in situations where player(s) are close to the end of the game or have little to risk with many dice to roll. Player 2 should exceed ($s < 2$) or meet ($s = 2$) player 1's score when player 1 has reached the goal score, so the planes $i + b = j$ or $i + b = j + 1$ are also a prominent hold planes. As a player has fewer dice to roll, play becomes more conservative.

There are some interesting differences and subtleties to observe as well. Player 1 plays more aggressively than player 2 with higher minimum hold values with all other state variables being equal. Also, there are interesting nonlinearities when player 1 seeks to not just reach 13 points, but to far enough exceed 13 so as to make it unlikely that player 2 will exceed their final score. Player 2, having the opportunity to exceed player 1's final score, has an advantage and generally plays so as to keep within striking distance of player 1's score.

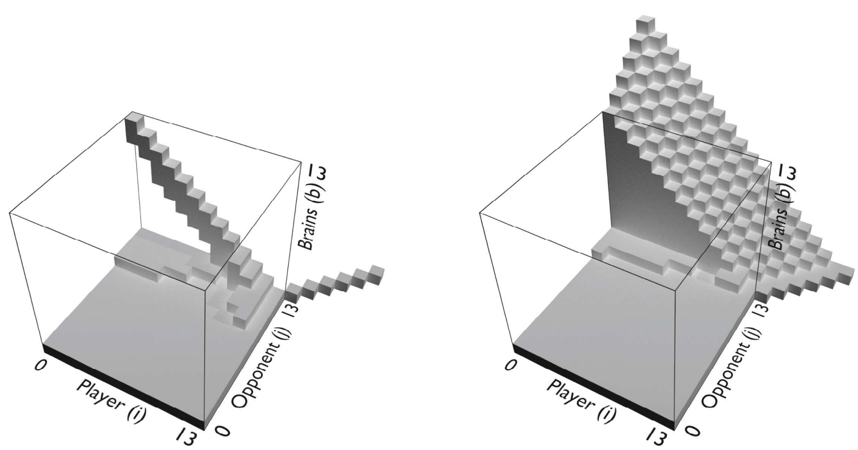
Most interesting and complex are the roll/hold boundaries when a player has rolled one shotgun. Here we observe nonlinearities in the roll/hold surface for both players. Whereas one might approximate player with no shotguns or two shotguns as "always roll" and "hold at 1", respectively, the roll/hold surface



(a) Player 1 with no shotguns rolled (b) Player 2 with no shotguns rolled



(c) Player 1 with one shotgun rolled (d) Player 2 with one shotgun rolled



(e) Player 1 with two shotguns rolled (f) Player 2 with two shotguns rolled

Fig. 2. AYZD optimal play visualization. A player in a state inside or outside the gray solid should roll or hold, respectively. Subfigures are by p, s cases, and axes follow i, j, b state variables.

shape is relatively complex when the current player has rolled one shotgun and player scores are not close to the goal.

We also observe that, during tiebreaker rounds, player 1 should hold at 6/3/1 when 0/1/2 shotgun(s) have been rolled, respectively.

5 Human-Playable Policies

In this section, we present a range of human-playable policies mapping states to roll/hold actions that trade off greater complexity for greater win rate. By *human-playable*, we mean that all roll/hold decisions may be made through simple mental math. As we will see, these policies range from extremely simple rules to very-complex sub-cases requiring memorization of several constants in order to approximate roll/hold surfaces.

Each policy is evaluated against the optimal policy with each having equal probability of playing first. Policy evaluation follows the same value-iteration-style algorithm of [6]. The performance of each is summarized in Fig. 3.

Policy	**Difference**
Fixed Hold-At	-0.0274
Minh Cases	-0.0133
Llano Cases	-0.0118
Neller Cases	-0.0100

Fig. 3. Differences between human-playable and optimal policy win rates

We present each policy as a method that returns whether or not to roll in the given state.

5.1 Fixed Hold-At Policy

First, we consider a Fixed Hold-At Policy (Algorithm 1) where we need only remember a few turn total thresholds.

Requiring memorization of only a few cases and two hold-at constants (4 and 1), Algorithm 1 reduces the optimal play gap to ∼2.74%.

5.2 Minh Cases Policy

The Minh Cases Policy (Algorithm 2) elaborates the Fixed Hold-At Policy (Algorithm 1) for situations where a player has rolled 1 shotgun.

In this situation, if the opponent's score is greater than or equal to 8, the current player will aim for the winning score of 13 or try to exceed the opponent's score by 3 brains if the opponent has reached 10 or more. This policy results in an optimal play gap of ∼1.33%.

The policy relies on four non-goal score constants to keep in mind (8, 3, 4, and 1) which is relatively simple but still provides a close approximation to optimal play.

Algorithm 1: Fixed Hold-At Policy

Input : player p, player score i, opponent score j, turn total b, shotguns rolled s

Output: whether or not to roll

```
if p = 2 ∧ j ≥ 13 ∧ i + b < j then        // When player 2 with j ≥ goal ...
    return true                            // and holding would lose, roll.
else if s = 0 then                         // Keep rolling with 0 shotguns.
    return true
else if s = 1 then                         // Hold at 4 with 1 shotgun.
    return b < 4
else                                       // Hold at 1 with 2 shotguns.
    return b < 1
end if
```

Algorithm 2: Minh Cases Policy

Input : player p, player score i, opponent score j, turn total b, shotguns rolled s

Output: whether or not to roll

```
if p = 2 ∧ j ≥ 13 ∧ i + b < j then        // When player 2 with j ≥ goal ...
    return true                            // and holding would lose, roll.
else if s = 0 then                         // Keep rolling with 0 shotguns.
    return true
else if s = 1 then                         // With 1 shotgun,
    if j ≥ 8 then                          // if opponent's score j ≥ 8,
        return i + b < max(13, j + 3)      // win (with lead of 3 if j ≥ 10),
    else                                   // else hold at 4.
        return b < 4
    end if
else                                       // Hold at 1 with 2 shotguns.
    return b < 1
end if
```

5.3 Llano Cases Policy

Algorithm 3, "Llano Cases Policy", breaks down cases by player p and number of shotguns rolled s. Policies for each player are near identical, with a minor difference for zero shotguns rolled.

Though not as computationally simple as the previous policies, this policy requires the memorization of few constants when playing below the goal score and allows the player to play nearly the same each game, regardless of whether they went first or second. The main differences arise from playing beyond the goal score with zero shotguns rolled as player 1, and the added condition of either player being at or above 10 in order to go for the goal with 1 shotgun rolled. Also, specific strategy has been determined for play when either player is

Algorithm 3: Llano Cases Policy

```
Input  : player p, player score i, opponent score j, turn total b, shotguns rolled s
Output: whether or not to roll
i' ← i + b                                          // i': score after holding
h = {6, 3, 1}     // h: tiebreaker hold values indexed by shotguns rolled
if i ≥ 13 ∨ j ≥ 13 then       // If either player reached/exceeded goal...
    if i = j then                                 // if the scores are even...
        if p = 1 then   // player 1 holds at the appropriate turn score.
            return b < h[s]
        else                               // Player 2 holds when b reaches 1.
            return b < 1
        end if
    else if p = 2 ∧ i < j then                   // If player 2 is trailing...
        return (s < 2 ∧ i' ≤ j) ∨ (s = 2 ∧ i' < j) // match the opponent with
         2 shotguns, exceed by 1 otherwise.
    end if
    return false                                          // Otherwise hold.
else                             // If both players are below goal score...
    if s = 0 then                                  // if 0 shotguns rolled...
        if p = 1 then   // player 1 goes for the higher of goal or j + 9.
            return i' < max(13, j + 9)
        else                                   // Player 2 goes for the goal.
            return i' < 13
        end if
    else if s = 1 then                              // If 1 shotgun rolled...
        if i ≥ 10 ∧ j ≥ 10 then    // and either player has at least 10...
            return i' < 13                                 // go for the goal.
        else
            return b < 4                              // Otherwise, hold at 4.
        end if
    else                                           // If 2 shotguns rolled...
        return b < 1                                            // Hold at 1.
    end if
end if
```

above the goal score, involving the memorization of a few constants for player 1, and playing to catch or slightly exceed their opponent in the case of player 2.

5.4 Neller Cases Policy

Algorithm 4, "Neller Cases Policy", also breaks down cases by player p and number of shotguns rolled s.

This policy is similar in complexity and decisions to Algorithm 3, organizing and handling endgame and 0 shotgun cases differently. It also has player 1 and 2 treating 8 and 10 as different end-game score thresholds, respectively. However,

Algorithm 4: Neller Cases Policy

Input : player p, player score i, opponent score j, turn total b, shotguns rolled s
Output: whether or not to roll

```
i' ← i + b                                      // i': score after holding
h = {6, 3, 1}      // h: tiebreaker hold values indexed by shotguns rolled
if p = 1 then                                          // If player 1, ...
    if i ≥ 13 ∧ i = j then         // if tied at/above 13, hold at h values.
        return b < h[s]
    else if s = 0 then     // If 0 shotguns, hold at ≥ 13 with ≥ 8 lead.
        return i' < max(13, j + 8)
    end if
    e ← 8                 // Set player 1 end-game score threshold e to 8.
else                                                // Else if player 2, ...
    if j ≥ 13 then         // if player 1 has achieved the goal score, ...
        return (s < 2 ∧ i' ≤ j) ∨ (s = 2 ∧ i' < j)  // exceed/meet player 1's
          score with under/exactly 2 shotguns, respectively.
    else if i ≥ 13 ∧ i > j then                   // If holding wins, hold.
        return false
    else if s = 0 then                     // Always roll with no shotguns.
        return true
    end if
    e ← 10               // Set player 2 end-game score threshold e to 10.
end if
if s = 1 then                                    // If 1 shotgun rolled, ...
    if i ≥ e ∨ j ≥ e then                   // hold at 13 if score(s) ≥ e.
        return i' < 13
    else                                            // Otherwise, hold at 4.
        return b < 4
    end if
else if s = 2 then                    // If 2 shotguns rolled, hold at 1.
    return b < 1
end if
```

this better approximates optimal play performance, closing the optimal play gap to $\sim$1.00%.

Through these algorithms, we see that key play considerations (beyond trivial decisions regarding immediate win/loss) could be approximately summarized as follows: With 0 shotguns, keep rolling. With 1 shotgun, get at least 4 points, but close to the game end, try to win with a lead as player 1, and just win as player 2. With 2 shotguns, get at least 1 point. For tiebreaker rounds, player 1 rolls for 6/3/1 points with 0/1/2 shotguns, whereas player 2 tries to exceed/match player 1's score with under/exactly 2 shotguns.

6 Future Work

Recall that our AYZD simplification to Zombie Dice treats all dice as having the average (i.e. yellow) distribution of brains, shotguns, and footprints. In the full game, we have the distribution of dice and dice outcomes shown in Sect. 2.

Our next step will be to compute optimal play for the full complexity of 2-player Zombie Dice. We will then compare performance of both the optimal and human-playable AYZD policies against optimal Zombie Dice play to see how much dice color distribution matters for play performance.

7 Conclusions

In this paper, we have computed and visualized optimal play for the 2-player case of the All Yellow Zombie Dice game, a simplification of the regular Zombie Dice game. Prior work has determined optimal turn scoring strategy [3], but this is the first step towards understanding optimal game winning strategy. To maximize expectation of score gain per turn is not to maximize expectation of winning probability. We have determined that player 2 has a slight informational advantage due to the fact that the game will always end with player 2's turn.

In addition, we presented a variety of human-playable strategies, trading off simplicity for performance against optimal play, with optimal play performance gaps ranging from ∼2.74% to ∼1.00%. For casual play, we recommend the Minh Cases Policy (Algorithm 2) with an optimal play performance gap of only ∼1.33%.

The All Yellow Zombie Dice game has a fairly complex optimal roll-hold policy boundary (Fig. 2), yet relatively simple human-playable policies offer decent performance against optimal play, revealing some of the key considerations for excellent play.

References

1. Zombie dice. https://boardgamegeek.com/boardgame/62871/zombie-dice Accessed 24 May 2023
2. Busche, M., Neller, T.W.: Optimal play of the Farkle dice game. In: Advances in Computer Games: 15th International Conferences, ACG 2017, Leiden, The Netherlands, July 3–5, 2017, Revised Selected Papers 15, pp. 63–72. Springer (2017). https://doi.org/10.1007/978-3-319-71649-7_6
3. Cook, H.L., Taylor, D.G.: Zombie dice: An optimal play strategy (2014). https://arxiv.org/abs/1406.0351
4. Knizia, R.: Dice Games Properly Explained. Elliot Right-Way Books, Brighton Road, Lower Kingswood, Tadworth, Surrey, KT20 6TD U.K. (1999)
5. Neller, T.W., Presser, C.G.: Optimal play of the dice game pig. UMAP J. **25**(1), 25–47 (2004)
6. Neller, T.W., Presser, C.G.: Practical play of the dice game pig. UMAP J. **31**(1), 5–19 (2010)