A Sound Strategy for Farkle Dice

Ryan McKenna

1 Introduction

1.1 Background

Farkle is a game of chance played between at least 2 players. On each turn, a player rolls 6 dice and scores points based on the dice that they roll. After each roll, the player can choose to either bank their current amount of points, or risk the points and roll the non-scoring dice. There are a variety of different scoring variations that people play by, and I will address that in the assumptions section of this paper. The first player to reach (or surpass) 10,000 points wins the game.

1.2 Mathematics

Farkle dice is a game of perfect information, that is, both players have access to all the information about the current state of the game. Farkle is a non-deterministic game, as the outcome relies on a random event. Unlike in deterministic games, it is impossible to come up with a strategy that would result in a win every game. Instead, the goal is to come up with a strategy that would maximize the probability of winning any given game.

2 Assumptions

There are a lot of scoring variations on farkle. In this paper, I will address one specific variation. Here are the rules:

- No threshold score required to enter game
- 1's are worth 100
- 5's are worth 50
- 3 of a kind is worth [1000,200,300,400,500,600]
- 4 of a kind is worth [2000,400,600,800,1000,1200]
- 5 of a kind is worth [3000,600,900,1200,1500,1800]
- 6 of a kind is worth [4000,800,1200,1600,2000,2400]

- A straight 1-6 is worth 1500
- A 3-pair is worth 750
- Everything else is a Farkle, which means you lose all the points you've accumulated on the current roll

3 Solution

3.1 Approach

Although the ultimate goal is to maximize the probability of winning, an easier problem to solve is to maximize the expected number of points you score on any single roll. While these might seem equivalent, they are not. For example, if you are losing 5000 to 9000, you should probably take more risk to try to catch up.

The first step to solving the game was to come up with a probability distribution for all the possible rolls. Once that was finished, I used those probabilities and a tree recursive function to evaluate all possible outcomes and weight them accordingly.

The benefit of doing it this way, as opposed to using a simulation, is that the results obtained are exact and reliable. Due to the computational complexity of the algorithm, I used memoization to cache results so they could be looked up at a later time (rather than being recomputed). This improved the running time of the algorithm from exponential to linear.

3.2 Representation of the Game State

Since I am only concerned with maximizing the EV on any single roll, the only attributes of my game state are number of points and number of dice to roll. This will make memoization use very little memory – to be exact a 6x200 array where the first dimension is the number of dice and the second dimension is the number of points accumulated[1]. The value is the maximum EV from that position assuming optimal play.

If I were to maximize the probability of winning, however, I would also have to consider my overall score as well as the scores of all my opponents. For simplicity and compuational limitations, I do not consider these variables.

[1]Since you can only score in multiples of 50, the array only has to be 10000 / 50 = 200 in length

3.3 Table of Probabilities

I calculated all the probabilities by hand using counting principles. For consiseness, I will only show the probability distribution for a 3 dice situation.

Points	Example	Combinations	Rolling
1000	111	1	6
200	222	1	6
300	333	1	6
400	444	1	6
500	555	1	6
600	666	1	6
250	115	3	6
200	155	3	6
200	112	12	1
150	152	24	1
100	552	12	1
100	122	48	2
50	522	48	2
0	223	60	-

In some situations, you have multiple choices. For example, in the 150 point example (152), you could either bank the 1 and the 5 and stop, bank the 1 and the 5 and roll 1 die, or bank the 1 and roll 2[1]. This is where the AI comes into play, as it will evaluate all possible plays and return the play that ends up resulting in the highest expected value.

[1]Technically, you could also bank the 5 and roll 2 as well, but that strategy is dominated by banking the 1 and rolling 2, which is why I left it out.

3.4 Pseudocode

```
double expectedValue(int pts, int dice) {
     if(pts >= 10000) return pts; //base case
        double ev_roll; //EV if I continue to roll
        switch(dice) {
            case 1:
             ev_roll = (expectedValue(pts+50,6) +
                           expectedValue(pts+100,6)) / 6.0;
                ... (cache result) ...
                return max(ev_roll, pts);
            case 2:
                ... (calculate ev_roll) ...
                ... (cache result) ...
                return max(ev_roll,pts);
            case 3:
                ev_roll = (... +
                     24 * max(expectedValue(pts+100,2),
                              expectedValue(pts+150,1)) +
                     ...) / 216.0;
                ... (cache result) ...
                return max(ev_roll,pts);
            case 4:
                ... (calculate ev_roll) ...
                ... (cache result) ...
                return max(ev_roll,pts);
            case 5:
                ... (calculate ev_roll) ...
                ... (cache result) ...
                return max(ev_roll,pts);
            case 6:
                ... (calculate ev_roll) ...
                ... (cache result) ...
                return max(ev_roll,pts);
        }
}
```

3.5 Explanation of Code

As the code indicates in case 1 and case 3, ev-roll is calculated by a recursive formula. In fact, it is a tree recursive formula where the number of branches depends on the number of distinct outcomes for the given number of dice (3 for 1 die, 13 for 3 dice). This computationally intense problem is greatly simplified by caching the results to avoid recomputation.

The smart decision making occurs when there is a situation that the player has more than one choice. For example, consider a situation in which the player rolled 1,5,2. The player could bank the 1,5 and roll the 2, bank the 1,5 and stop, or bank the 1 and roll the 5 and 2. The AI checks all of these and returns the one that results in the highest EV.[1]

3.6 Results

	Number of Dice					
Points	1	2	3	4	5	6
0						548.858
50					363.834	
100				274.776	399.911	
150			227.968	307.673		
200		221.706	257.584	343.8		
250	274.669	250	293.4			
300	300	300	329.228			802
350	350	350			596.22	846.014
400	400	400	400.947	498.2	638.291	890.036
450	450	450	450	540.176	680.368	934.065
500	500	500	500	582.157	722.453	978.099
550	550	550	550	624.144	764.545	1022.666
600	600	600	600	666.139	806.642	1067.891
650	650	650	650	708.139	849.953	1113.281
700	700	700	700	750.142	893.894	1158.674
750	750	750	750	792.147	937.847	1204.068
800	800	800	800	834.154	981.801	1249.464
850	850	850	850	876.162	1025.756	1295.174
900	900	900	900	918.17	1069.711	1341.411
950	950	950	950	960.178	1114.429	1387.917
1000	1000	1000	1000	1002.188	1160.198	1434.468
1050	1050	1050	1050	1050	1206.254	1481.147
1100	1100	1100	1100	1100	1252.311	1528.007

Figure 1: A condensed table of results

[1]See case 3 to see how this is implemented.

Figure 1 shows a condensed table of results, where each entry in the table is the expected value in the given situation assuming optimal play. The cells are color coated by optimal play, green indicates that you should roll, red indicates that you should bank, and an uncolored cell indicates that the state is unreachable.

In situations where you have multiple choices, you should choose the choice in the table that has the highest value. For example, lets say I rolled 2,3,3,3,5,6, then I can bank the 350 (ev 350), roll 3 with 300 (ev 329.2), or roll 5 with 50 (ev 363.8). It's easy to see that rolling 5 with 50 is the optimal choice in this situation.

Since I do not show the entire table, I will give the transition values for 5 dice and 6 dice.

- For 5 dice, the it is optimal to roll up to and including 3,000 points. After that, it is best to bank.
- For 6 dice, it is optimal to roll up to and beyond 10,000 points. In other words, is is never optimal to bank with 6 dice.

4 Conclusion

I believe any player that uses the strategy derived in this paper will have an edge against all opponents that they face. That being said, the approach does not necessarily maximize your probability of winning. A true optimal strategy would be dynamic and it would change as the game progresses. The computational complexity of solving such a problem with a brute force approach is beyond my computers capabilites