Math 381

Six-Sided Fair Die Game

Nov 19, 2021

Chenhao Lin

Suppose there is a game designed in the following way.

Let two players alternately roll a six-sided fair die and follow the following rules:

1. A player could either add or not add the roll to his current score.
2. If a player's score is $\geq$ M, then the game ends and that player wins the game.
3. If the game is not over yet and the player's current score is an even number, the player can choose to gamble or not. If the player decides to gamble, the player needs to roll the dice again. If the roll is greater than 3, add 10 to his current score. Otherwise subtract 10 from his current score. If the score ever becomes negative, replace the score with zero.
    - Since there is equal probability to add or subtract 10, the expected gain from this gamble is 0, so the gamble is fair.

To win the game, the player might use the following strategies:

A. Add the roll to the player's score only if the roll is an even number, so his score will always satisfy the requirement for rule 3 if the game isn't over. He will always gamble.
B. Always add the roll to the player's score. When the score is an even number that is smaller than 10, the player will choose to gamble. Otherwise, the player will not gamble.
C. Add the roll to the player's score. When the player's score is an even number and smaller than the other player's score, the player will choose to gamble. Otherwise, the player will not gamble.

D. Add the roll to the player's score. When the player's score is an even number and "the opponent score - the player's his score ≤ 6", the player will choose to gamble. Otherwise, the player will not gamble.

E. Always add the roll to the player's current score and never gamble.

F. Always add the roll to the player's current score and always gamble if possible.

To find out the performance of these strategies, we choose three goal scores: 10, 25, and 100. We played 6 games for each strategy pair and took turns to start the game first. The following charts are the results (player 1 chooses the strategies from the leftmost column, player 2 chooses the strategies from the top row, and data below represent the number of games player 1 wins when number of games player 2 wins could be calculated as "6 - number of games player 1 wins".

Chart for goal score(M) = 10:

	A	B	C	D	E	F
A	--	4	5	4	4	4
B	--	--	4	3	5	3
C	--	--	--	2	4	2
D	--	--	--	--	3	3
E	--	--	--	--	--	2
F	--	--	--	--	--	--

Chart for goal score(M) = 25:

	A	B	C	D	E	F
A	--	3	4	3	4	3
B	--	--	3	2	2	4
C	--	--	--	2	3	2
D	--	--	--	--	3	3
E	--	--	--	--	--	3
F	--	--	--	--	--	--

Chart for goal score(M) = 100:

	A	B	C	D	E	F
A	--	2	2	1	2	1
B	--	--	2	2	4	3
C	--	--	--	1	3	2
D	--	--	--	--	4	3
E	--	--	--	--	--	2
F	--	--	--	--	--	--

According to the above game results, we observed that whatever the goal score is, strategy D and strategy F have about the same probability to win. To find out which strategy is better, we wrote the code to simulate the game in which players use these two strategies and observe which strategy has higher probability to win when the goal score is 100.

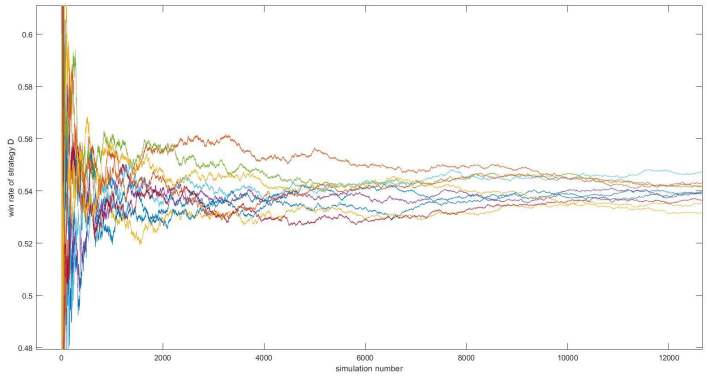
Coding Part:

Below is the Java code we write to check for the convergence of average winning probability by simulating 100000 games for 10 times with players using strategy D and strategy F when the goal score is 100:

```
import java.util.Arrays;
public class HW6_1 {
	public static int games=100000; // Equals to the number of simulation
	public static int goal=100; // Equals to the goal number
	// Strategy D and F
	public static void main(String[] args) {
		int scoreA=0;
		int scoreB=0;
		int steps=0;
		int winA=0;
		int winB=0;
		// To check the convergence, do 100000 simulations for 10 times
		for (int j=0;j<10;j++) {
			float allA[]=new float[games];
			float allB[]=new float[games];
			winA=0;
```

```
                winB=0;
                for (int i=0;i<games;i++) {
                        scoreA=0;
                        scoreB=0;
                        steps=0;
                        while (scoreA<goal && scoreB<goal) {
                                int rand=(int)(1+Math.random()*6);
                                steps+=1;
                                if (steps%2==i%2) { // Let player 1 and player 2 take turns to start first
                                        // Let player 1 use strategy D
                                        scoreA+=rand;
                                        if (scoreA-scoreB<=6 && scoreA%2==0 && scoreA<goal) {
                                                scoreA=gamble(scoreA);
                                        }
                                } else {
                                        // Let player 2 use strategy F
                                        scoreB+=rand;
                                        if (scoreB%2==0 && scoreB<goal) {
                                                scoreB=gamble(scoreB);
                                        }
                                }
                        }
                        if (scoreA>=goal) {
                                winA+=1; // Count the number of times player 1 wins
                        } else {
                                winB+=1; // Count the number of times player 2 wins
                        }
                        // Calculate the current win rate of strategy D
                        allA[i]=(float) (winA*1.0/(i+1));
                        // Calculate the current win rate of strategy F
                        allB[i]=(float) (winB*1.0/(i+1));
                }
                System.out.println(Arrays.toString(allA));
        }
    }
    // The function for gamble
    public static int gamble(int score) {
        int rand=(int)(1+Math.random()*6);
        if (rand>3) {
                score+=10;
        } else {
                if (score-10<0) {
                        score=0;
                } else {
                        score-=10;
                }
        }
        return score;
    }
}
```

Here is the plot for 10 runs of the above game simulation and 100000 game simulations per run (this is the zoomed-in part when simulation number goes from 0 to 12000):



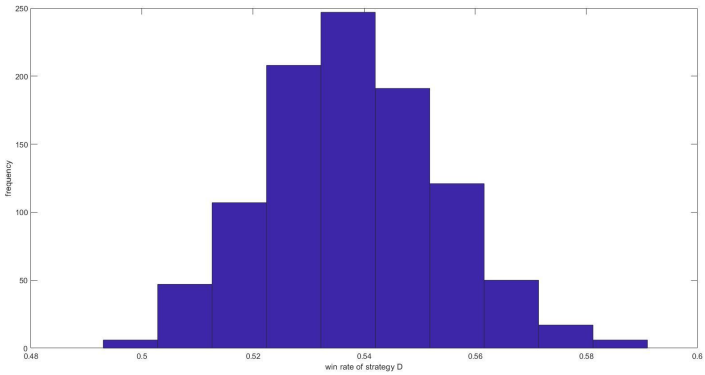
We could see that there is a convergence in the plot. When simulation number = 100000, the minimum winning rate for strategy D in the 10 runs is 0.53746 and the maximum winning rate for strategy D in the 10 runs is 0.54092. So, in these 10 runs, we conclude that as the simulation times approaches 100000, the winning rate for strategy D converges in [0.53746, 0.54092].

Below is the Java code we write to find out the distribution of average win rate of strategy D by simulating 1000 games for 1000 times with players using strategy D and strategy F when the goal score is 100:

```
import java.util.Arrays;
public class HW6_2 {
	public static int games=1000; // Equals to the number of simulation
	public static int goal=100; // Equals to the goal number
	// Strategy D and F
	public static void main(String[] args) {
		int scoreA=0;
		int scoreB=0;
		int steps=0;
		int winA=0;
		int winB=0;
		float allA[]=new float[games];
		float allB[]=new float[games];
		// To find out the distribution, do 1000 simulations for 1000 times
		for (int j=0;j<games;j++) {
			winA=0;
			winB=0;
			for (int i=0;i<games;i++) {
				scoreA=0;
				scoreB=0;
				steps=0;
				while (scoreA<goal && scoreB<goal) {
					int rand=(int)(1+Math.random()*6);
					steps+=1;
					if (steps%2==i%2) { // Let player 1 and player 2 take turns to start first
						// Let player 1 use strategy D
```

```
                                        scoreA+=rand;
                                        if (scoreA-scoreB<=6 && scoreA%2==0 && scoreA<goal) {
                                                scoreA=gamble(scoreA);
                                        }
                                } else {
                                        // Let player 2 use strategy F
                                        scoreB+=rand;
                                        if (scoreB%2==0 && scoreB<goal) {
                                                scoreB=gamble(scoreB);
                                        }
                                }
                        }
                        if (scoreA>=goal) {
                                winA+=1; // Count the number of times player 1 wins
                        } else {
                                winB+=1; // Count the number of times player 2 wins
                        }
                }
                // Calculate the average win rate of strategy D of 1000 simulations
                allA[j]=(float) (winA*1.0/games);
                // Calculate the average win rate of strategy F of 1000 simulations
                allB[j]=(float) (winB*1.0/games);
        }
        System.out.println(Arrays.toString(allA));
    }
    // The function for gamble
    public static int gamble(int score) {
        int rand=(int)(1+Math.random()*6);
        if (rand>3) {
                score+=10;
        } else {
                if (score-10<0) {
                        score=0;
                } else {
                        score-=10;
                }
        }
        return score;
    }
}
```

Here is the distribution of average winning probability with 1000 runs of the above game simulation and 1000 game simulations per run:



According to the Central Limit Theorem, the distribution of strategy D's average win probability is nearly normal. Since the probability that all 10 of our estimates are on one side of the true probability is $2/2^{10} \approx 0.001953125$, the probability that the true average win probability of strategy D is $1 - 0.001953125 \approx 99.8\%$. Therefore, there is a 99.8% probability that the average win probability of strategy D is in [0.53746, 0.54092].

We say the true average win probability of strategy D is in the interval [0.53746, 0.54092] at 99.8% level of confidence.

Now, we move to the final step for this paper: analyze this game.

Analyze this game:

We try to find out the best strategies from the 6 strategies above.

Coding part:

Below is the code for simulation:

```
import java.util.Arrays;
public class HW6_3 {
	public static int games=100000; // Equals to the number of simulation
	public static int goal=100; // Equals to the goal number
	public static int num=6; // Equals to the gap mentioned in strategy D
	public static void main(String[] args) {
		int scoreA=0;
		int scoreB=0;
		int steps=0;
		int winA=0;
		int winB=0;
		float allA[]=new float[10];
		float allB[]=new float[10];
		// To check the convergence, do 100000 simulations for 10 times
		for (int j=0;j<10;j++) {
			winA=0;
			winB=0;
			for (int i=0;i<games;i++) {
				scoreA=0;
				scoreB=0;
				steps=0;
				while (scoreA<goal && scoreB<goal) {
					int rand=(int)(1+Math.random()*6);
					steps+=1;
					if (steps%2==i%2) { // Let player 1 and player 2 take turns to start first
						// Strategy A
						/*
						if (rand%2==0) {
							scoreA+=rand;
						}
```

```
                if (scoreA%2==0 && scoreA<goal) {
                        scoreA=gamble(scoreA);
                }
                */
                // Strategy B
                /*
                scoreA+=rand;
                if (scoreA<10 && scoreA%2==0 && scoreA<goal) {
                        scoreA=gamble(scoreA);
                }
                */
                // Strategy C
                /*
                scoreA+=rand;
                if (scoreB>scoreA && scoreA%2==0 && scoreA<goal) {
                        scoreA=gamble(scoreA);
                }
                */
                // Strategy D

                scoreA+=rand;
                if (scoreA-scoreB<=num && scoreA%2==0 && scoreA<goal) {
                        scoreA=gamble(scoreA);
                }

                // Strategy E
                /*
                scoreA+=rand;
                */
                // Strategy F
                /*
                scoreA+=rand;
                if (scoreA%2==0 && scoreA<goal) {
                        scoreA=gamble(scoreA);
                }
                */
        } else {
                // Strategy A
                /*
                if (rand%2==0) {
                        scoreB+=rand;
                }
                if (scoreB%2==0 && scoreB<goal) {
                        scoreB=gamble(scoreB);
                }
                */
                // Strategy B
                /*
                scoreB+=rand;
                if (scoreB<10 && scoreB%2==0 && scoreB<goal) {
                        scoreB=gamble(scoreB);
                }
                */
                // Strategy C
                /*
                scoreB+=rand;
                if (scoreA>scoreB && scoreB%2==0 && scoreB<goal) {
                        scoreB=gamble(scoreB);
                }
                */
                // Strategy D
                /*
```

```
                                        scoreB+=rand;
                                        if (scoreB-scoreA<=num && scoreB%2==0 && scoreB<goal) {
                                                scoreB=gamble(scoreB);
                                        }
                                        */
                                        // Strategy E
                                        /*
                                        scoreB+=rand;
                                        */
                                        // Strategy F

                                        scoreB+=rand;
                                        if (scoreB%2==0 && scoreB<goal) {
                                                scoreB=gamble(scoreB);
                                        }

                                }
                        }
                        if (scoreA>=goal) {
                                winA+=1; // Count the number of times player 1 wins
                        } else {
                                winB+=1; // Count the number of times player 2 wins
                        }
                }
                // Calculate the average win rate of strategy of 100000 simulations
                allA[j]=(float) (winA*1.0/games);
                // Calculate the average win rate of strategy of 100000 simulations
                allB[j]=(float) (winB*1.0/games);
        }
        Arrays.sort(allA);
        // Print the confidence interval
        System.out.print("["+allA[0]+", "+allA[9]+"]");
    }
    // The function for gamble
    public static int gamble(int score) {
        int rand=(int)(1+Math.random()*6);
        if (rand>3) {
                score+=10;
        } else {
                if (score-10<0) {
                        score=0;
                } else {
                        score-=10;
                }
        }
        return score;
    }
}
```

We will make 10 runs for every pair of different strategies and calculate the confidence interval by using the same way that is mentioned earlier.

The charts below will include the confidence interval for average winning rate of row strategies v.s. column strategies.

Now, let’s take a look at the charts for more information:

Chart for goal score(M) = 10:

	A	B	C	D	E	F
A	--	[0.66358, 0.66785]	[0.81852, 0.82246]	[0.66653, 0.67123]	[0.8192, 0.82353]	[0.66293, 0.66705]
B	--	--	[0.59887, 0.60315]	[0.49714, 0.50148]	[0.63712, 0.64263]	[0.49821, 0.50397]
C	--	--	--	[0.39443, 0.40012]	[0.58738, 0.59367]	[0.39598, 0.40154]
D	--	--	--	--	[0.63761, 0.64201]	[0.49812, 0.5034]
E	--	--	--	--	--	[0.35741, 0.36331]
F	--	--	--	--	--	--

Chart for goal score(M) = 25:

	A	B	C	D	E	F
A	--	[0.55282, 0.55901]	[0.5881, 0.59512]	[0.49997, 0.50591]	[0.64754, 0.65265]	[0.51663, 0.52024]
B	--	--	[0.47528, 0.48157]	[0.41604, 0.42073]	[0.66921, 0.67228]	[0.45675, 0.4618]
C	--	--	--	[0.40689, 0.41274]	[0.67078, 0.67839]	[0.42377, 0.42985]
D	--	--	--	--	[0.72448, 0.72925]	[0.52104, 0.52519]
E	--	--	--	--	--	[0.33386, 0.3389]
F	--	--	--	--	--	--

Chart for goal score(M) = 100:

	A	B	C	D	E	F
A	--	[0.35814, 0.3637]	[0.34127, 0.34795]	[0.31391, 0.31842]	[0.37881, 0.38421]	[0.3392, 0.34604]
B	--	--	[0.32503, 0.32979]	[0.30175, 0.30543]	[0.59649, 0.6023]	[0.43539, 0.44108]
C	--	--	--	[0.45648, 0.4621]	[0.72239, 0.72737]	[0.49018, 0.4967]
D	--	--	--	--	[0.75403, 0.75612]	[0.53746, 0.54092]
E	--	--	--	--	--	[0.39743, 0.40151]
F	--	--	--	--	--	--

By the three grids for confidence intervals above, we find the following interesting things:

- When the goal score equals to 10, strategy A is the best strategy among these 6 strategies, *since the lower bound for the confidence interval of average winning rate is always larger than 0.5 no matter the opponent uses which strategy and there is no overlapping between the confidence intervals*.
- When the goal score equals to 25, strategy A and strategy D have better performance than the other 4 strategies because of the same reason mentioned above. Since the confidence interval of average winning rate of strategy A and strategy D contain 0.5 and have overlap, these two strategies have a similar winning rate.
- When the goal score equals to 100, strategy D has the highest winning rate because of the same reason mentioned above.

Now, we try to improve strategy D when the goal score is 100. We will change the number "6" in strategy D to other numbers and see how itse winning rate will change. Define strategy D-x as same as original strategy D except we change the number from "6" to "x". We will pick goal score (M) = 100 and make a chart for x = 5, 6, 7, with one x smaller than the original value and one x larger than the original value. The chart with confidence intervals is below:

	D-5	D-6	D-7
D-5	--	[0.48873, 0.49517]	[0.48971, 0.4961]
D-6	--	--	[0.49528, 0.49953]
D-7	--	--	--

By the confidence intervals above, we could increase strategy D's winning rate by changing the number "6" to a larger number (by the same reason we mentioned at the top of this page). Now we try to optimize strategy D by making x in [7, 11]:

	D-7	D-8	D-9	D-10	D-11
D-7	--	[0.49714, 0.50014]	[0.49593, 0.4995]	[0.49478, 0.49901]	[0.49607, 0.50102]
D-8	--	--	[0.49865, 0.50212]	[0.49611, 0.49978]	[0.49734, 0.50211]
D-9	--	--	--	[0.49755, 0.49996]	[0.49812, 0.50259]
D-10	--	--	--	--	[0.50002, 0.50273]
D-11	--	--	--	--	--

We see that strategy D-9 is better than strategy D-8, strategy D-10 is better than strategy D-9, but strategy D-11 is a little bit worse than strategy D-10 through confidence intervals, so we could end with the conclusion that strategy D-x's winning rate reaches the peak when x = 10. Thus, if we replace strategy D by strategy D-10, strategy D would have the highest possible winning rate.

Now, I will compare strategy D-10 against strategy A through F when M = 100:

Chart for goal score(M) = 100:

	A	B	C	D	E	F
D-10	[0.68684, 0.69054]	[0.66498, 0.67186]	[0.54508, 0.54891]	[0.50427, 0.51032]	[0.72171, 0.72503]	[0.54228, 0.54645]

We could see that when goal score = 100, strategy D-10 is better than the other 6 strategies because of the reason mentioned at the beginning of page 11.

Try to make strategy D-10 even better:

Strategy D-10 only considers the situation about the gap between two players for gambling, but it doesn't put the opponent's score into the consideration. To find out the best time the player starts to use strategy D-10, we will let strategy D-10 make the player gamble when the player is not leading by more than 10 points and *the opponent's score is no smaller than N*.

Below is the code for simulation for strategy D-10 v.s. other 6 strategies when goal score = 100 and we let D-10 take the consideration of opponent's score:

```
import java.util.Arrays;
public class HW6_4 {
	public static int games=100000; // Equals to the number of simulation
	public static int goal=100; // Equals to the goal number
	public static int num=10; // Equals to the gap mentioned in strategy D
	public static int opponent=30; // Equals to when the player starts to use strategy D-10
	public static void main(String[] args) {
		int scoreA=0;
		int scoreB=0;
		int steps=0;
		int winA=0;
		int winB=0;
		float allA[]=new float[10];
		float allB[]=new float[10];
		// To check the convergence, do 100000 simulations for 10 times
		for (int j=0;j<10;j++) {
			winA=0;
			winB=0;
			for (int i=0;i<games;i++) {
				scoreA=0;
				scoreB=0;
				steps=0;
				while (scoreA<goal && scoreB<goal) {
					int rand=(int)(1+Math.random()*6);
					steps+=1;
					if (steps%2==i%2) { // Let player 1 and player 2 take turns to start first
						// Strategy D
						scoreA+=rand;
						if (scoreA-scoreB<=num && scoreA%2==0 && scoreA<goal &&
								scoreB>=opponent) {
							scoreA=gamble(scoreA);
						}
					} else {
						// Strategy A
						/*
						if (rand%2==0) {
							scoreB+=rand;
						}
						if (scoreB%2==0 && scoreB<goal) {
							scoreB=gamble(scoreB);
						}
						*/
						// Strategy B
						/*
						scoreB+=rand;
						if (scoreB<10 && scoreB%2==0 && scoreB<goal) {
							scoreB=gamble(scoreB);
						}
						*/
						// Strategy C
						/*
						scoreB+=rand;
						if (scoreA>scoreB && scoreB%2==0 && scoreB<goal) {
							scoreB=gamble(scoreB);
						}
						*/
						// Strategy E

						scoreB+=rand;

						// Strategy F
						/*
```

```
                                        scoreB+=rand;
                                        if (scoreB%2==0 && scoreB<goal) {
                                                scoreB=gamble(scoreB);
                                        }
                                        */
                                }
                        }
                        if (scoreA>=goal) {
                                winA+=1; // Count the number of times player 1 wins
                        } else {
                                winB+=1; // Count the number of times player 2 wins
                        }
                }
                // Calculate the average win rate of strategy of 100000 simulations
                allA[j]=(float) (winA*1.0/games);
                // Calculate the average win rate of strategy of 100000 simulations
                allB[j]=(float) (winB*1.0/games);
        }
        Arrays.sort(allA);
        // Print the confidence interval
        System.out.print("["+allA[0]+", "+allA[9]+"]");
    }
    // The function for gamble
    public static int gamble(int score) {
        int rand=(int)(1+Math.random()*6);
        if (rand>3) {
                score+=10;
        } else {
                if (score-10<0) {
                        score=0;
                } else {
                        score-=10;
                }
        }
        return score;
    }
}
```

The chart below represents the confidence interval of average winning rate when using strategy D-10 against the strategies at the left column when goal score(M) = 100 and the top row represents the value of N, which is mentioned at the end of page 12:

	0	30	60	90
A	[0.68684, 0.69054]	[0.66277, 0.66879]	[0.65422, 0.65603]	[0.62947, 0.63346]
B	[0.66498, 0.67186]	[0.60181, 0.60869]	[0.57095, 0.57495]	[0.50664, 0.51185]
C	[0.54508, 0.54891]	[0.50469, 0.50864]	[0.48833, 0.49258]	[0.42284, 0.42617]
D	[0.50427, 0.51032]	[0.4531, 0.45862]	[0.42606, 0.43028]	[0.42521, 0.43004]
E	[0.72171, 0.72503]	[0.65845, 0.66234]	[0.63675, 0.64074]	[0.59863, 0.60244]
F	[0.54228, 0.54645]	[0.49965, 0.50449]	[0.47709, 0.48416]	[0.42631, 0.43109]

Now we could have a hypothesis that the winning rate of strategy D-10 is decreasing when N is increasing. To make the hypothesis be more reasonable, we change the value of N to {0, 5, 10, 15} and get the chart below (the goal score is still 100):

	0	5	10	15
A	[0.68684, 0.69054]	[0.67516, 0.67887]	[0.67234, 0.67742]	[0.66758, 0.67055]
B	[0.66498, 0.67186]	[0.64011, 0.64303]	[0.63321, 0.63674]	[0.62446, 0.63171]
C	[0.54508, 0.54891]	[0.52845, 0.53231]	[0.51905, 0.52216]	[0.51056, 0.51621]
D	[0.50427, 0.51032]	0.48241, 0.48645]	[0.47489, 0.47859]	[0.46923, 0.47262]
E	[0.72171, 0.72503]	[0.69622, 0.70026]	[0.67844, 0.68205]	[0.66859, 0.67479]
F	[0.54228, 0.54645]	[0.52399, 0.52757]	[0.51601, 0.52348]	[0.51154, 0.51722]

We now can conclude when the value of N increases, the winning rate of strategy D-10 against the other 6 strategies strictly decreases. Thus, we should make N = 0 (or to say, always gamble when the player is not leading by more than 10 points regardless of opponent's score if possible) to make strategy D-10 has a higher winning rate.