Team 105 Facebook Farkle Program

School of Dreams Academy

Team Members:

Jonathan Daniels
Joseph Orona
Albert Reed

Executive Summary

Our team's project was to test whether a Facebook Farkle player would have a better chance of success if they followed mathematical law instead of “gut instinct.” We began this project by working with our school's mathematician learning Matlab and the probability that would be involved in this project. As the end of the year approached, we were swamped with other competitions. These challenges include FIRST robotics, BEST robotics, and Botball. When our final competition came to a close in the middle of March our focus shifted back over to supercomputing challenge. Because our school’s mathematician left, we no longer had an expert in Matlab; consequently, we switched over to the programming language NetLogo. As a first step, we have been attempting to create a Farkle game in NetLogo. Once the game is created, we can start programming in players that follow an optimal Farkle strategy following the laws of probability and players that imitate how most people would play and see which

players does better scores better or wins more often. Another factor we would like to include into next year's program is the built in feature to Facebook's Farkle game, power dice. These dice help the player in various ways and are used at the player's discretion. A problem to solve next year is whether or not using these power dice following the laws of probability will lead to more success over following "gut instinct."

Problem Statement

Farkle is a simple dice game that relies heavily on probability, strategy, and luck. A player will roll six dice, remove scoring dice, and re-roll the remaining dice, if they decide to. If a player rolls no scoring dice, they “Farkle Out”, meaning that their turn is over and they do not receive any points for that turn. The Facebook version of Farkle throws in “Power Dice” which give the game a whole new dimension. The main problem that we were trying to solve was how to build the optimal Facebook Farkle strategy based on your current score, the score of your opponents, the number of remaining dice you have to roll that turn, and so on. As the year progressed, our team changed our focus to other engineering competitions we were competing in, which meant we didn’t spend as much time working on this project as we needed to. Consequentially, we had to revise our project.

Instead of building a program that would develop optimal Facebook Farkle strategy, our problem turned to building a NetLogo program that would act as a Farkle game following standard Farkle rules. Once we got to this point, we could incorporate a computer player that would follow optimal Farkle strategy from the beginning in order to compare success rate to a human player who was playing without any prior knowledge of said strategy.

The main goal of this project is to learn about the statistics and probability involved in this game, while gaining an understanding on how it can be applied to real world problems. Although we did not achieve our original end product, we still feel this was a successful year. This is our first year working on the Supercomputing Challenge, so we are also using this project to learn about the challenge and to gain the skills necessary to be competitive in the challenge in future years.

Method

We began our project by learning how to build simple functions in Matlab. The functions that we put together were all exercises based off of statistics that our mentor gave us. The most complete one, as seem in appendix B.1, would take user input of a number of dice between one and two, and, based on standard Farkle rules, give back the odds of scoring certain amounts with those two dice. This program was helpful to us because we gained a better understanding of Farkle rules, Matlab programming, and probability.

The second step we took was to find all possible combinations with six dice. This seemed daunting at first, but after some research we found plenty of online resources that had such information available already. There are 46656 possible combinations, starting from 1,1,1,1,1,1 and ending with 6,6,6,6,6,6. From here we could begin looking at the probability of scoring certain dice combinations. These calculations can be found in Appendix A.1.

From this point, our goal was to develop a Matlab program that would take user input based on current score, turn number, opponent score, roll number, and number of available dice. After taking input, the program would output what move would be best on the human player's part, such as "roll again" or "end your turn." The program would be built on later to be more interactive and descriptive, but this was our first step.

This Matlab program was never complete, though. This is due to several factors, such as our mentor leaving the school and our team’s participation in other engineering competitions such as the FIRST Robotics competition. Because we couldn’t achieve what we planned, we narrowed the scope of our project.

Our new project idea was to develop a NetLogo Farkle game. After the game was complete, we could add a computer player that would play the game using optimal strategy and pit the computer against a human player in order to see the rates of success in both players. Thus far, we have the beginning of the game. Code for the game can be found in appendix C.

Results and Conclusions

Seeing as how our project was not fully complete, we have little results and conclusions. Most of what we have taken from this project is more knowledge on statistics, probability, and programming in Matlab and NetLogo. We now feel that we can successfully write a NetLogo program in order to model something, and we are also a bit more confident in our Matlab skills. In the future, we hope to apply what we learned to a more in depth project.

Credits and References

"Official Rules of Farkle." *Official Rules of Farkle*. N.p., n.d. Web. 03 Apr. 2013.

"The Solarium." *The Solarium*. N.p., 30 Aug. 2009. Web. 03 Apr. 2013.

"Farkle." *Wikipedia*. Wikimedia Foundation, 29 Mar. 2013. Web. 03 Apr. 2013.

Acknowledgements

Mentors

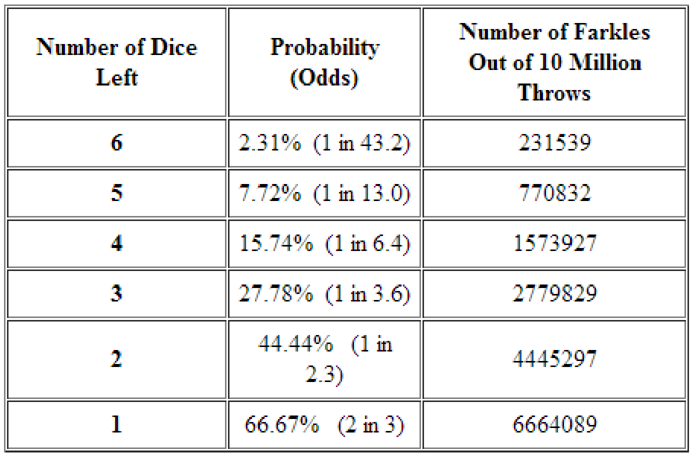
- Elizabeth Finley
- Creighton Edington

Appendix A.1

The total number of possibilities was found by multiplying the number of dice by the number of sides- 6^6

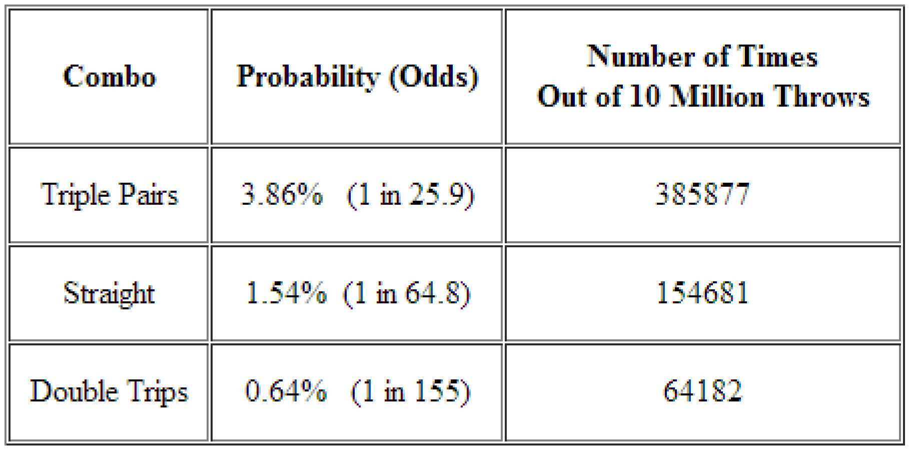
This is a chart of the probability of Farkling based on the number of dice you have.

(The Solarium)



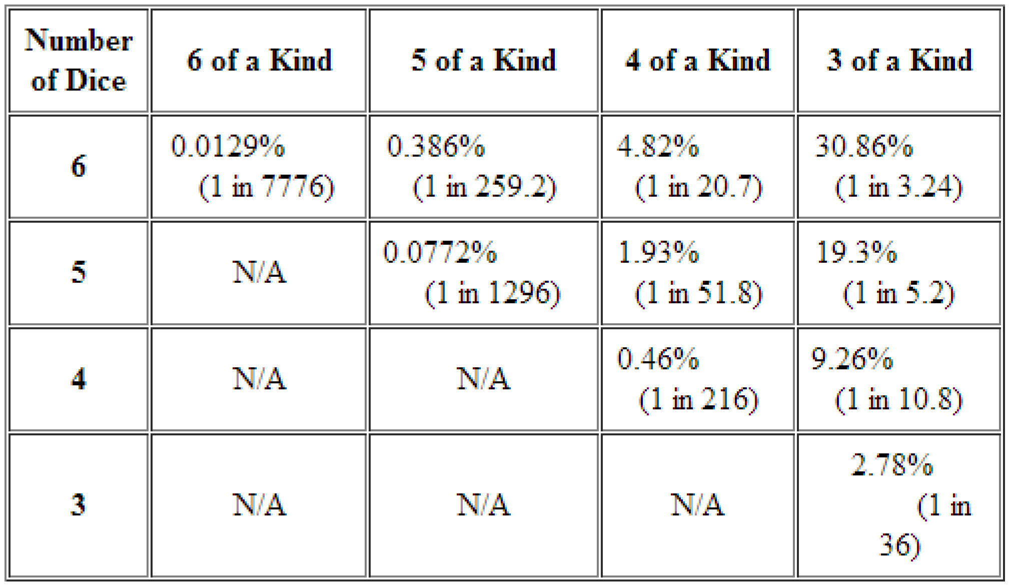
Number of Dice Left	Probability (Odds)	Number of Farkles Out of 10 Million Throws
6	2.31% (1 in 43.2)	231539
5	7.72% (1 in 13.0)	770832
4	15.74% (1 in 6.4)	1573927
3	27.78% (1 in 3.6)	2779829
2	44.44% (1 in 2.3)	4445297
1	66.67% (2 in 3)	6664089

This is a chart of the odds of scoring certain combinations. (The Solaruim)



Combo	Probability (Odds)	Number of Times Out of 10 Million Throws
Triple Pairs	3.86% (1 in 25.9)	385877
Straight	1.54% (1 in 64.8)	154681
Double Trips	0.64% (1 in 155)	64182

This is a chart of the odds of scoring x of a kind, depending on how many dice you have left.



Number of Dice	6 of a Kind	5 of a Kind	4 of a Kind	3 of a Kind
6	0.0129% (1 in 7776)	0.386% (1 in 259.2)	4.82% (1 in 20.7)	30.86% (1 in 3.24)
5	N/A	0.0772% (1 in 1296)	1.93% (1 in 51.8)	19.3% (1 in 5.2)
4	N/A	N/A	0.46% (1 in 216)	9.26% (1 in 10.8)
3	N/A	N/A	N/A	2.78% (1 in 36)

Appendix B.1

Below is our Matlab program that we built.

```
function [text] = dicefunc(d)
%Function should take input for d, with d being number of dice between one and two, and
give
%probability for scoring following standard farkle rules%


if (d == 1)
   text = fprintf('1/6 chance of 100 points; 1/6 chance of 50 points; 1/3 chance of scoring');
end;


if (d == 2)
   text = fprintf('8/36 chance of 50 points; 9/36 chance of 100 points; 2/26 chance of 150
points; 1/36 chance of 200 points; 20/36 chance of scoring');
end;


if (d < 1)
   text = fprintf('ERROR'); %I couldn't figure out how to do this in any other way, but it
works.%
end;


if (d > 2)
   text = fprintf('ERROR');
end;
```

Appendix C

Net Logo Code

```
globals [ farkle? total-score bank-score ]


breed [dice a-dice]


dice-own [
  spots
]


to setup
  clear-all
  ;; setup-globals
  setup-dice
  reset-ticks
  ;;setup-plot
end




to setup-dice
  set-default-shape dice "die-1"
  create-dice 6
  [
    set xcor who
    set color white
    roll-one
  ]
end

to roll-one
  set spots (1 + random 6)
  set shape (word "die-" spots)
end

to go
  roll
  farkle
  while [ bank-score = 0 ]
  [
   score-points
  ]

  tick
end



to roll
```

```
  ask dice [
    roll-one
  ]

end

to farkle
 ifelse ((any? dice with [ spots = 1 or spots = 5])
     or ( count dice with [ spots = 2 ] >= 3 )
     or ( count dice with [ spots = 3 ] >= 3 )
     or ( count dice with [ spots = 4 ] >= 3 )
     or ( count dice with [ spots = 6 ] >= 3 ))

 [
 ;; do nothing
 ]
 [
 set farkle? true
 print "farkle"
 ]
end


to score-points
```

```
 grab-die

end



to grab-die
   if mouse-down? [
   let candidate min-one-of turtles [distancexy mouse-xcor mouse-ycor]
   if [distancexy mouse-xcor mouse-ycor] of candidate < 1 [
     ;; The WATCH primitive puts a "halo" around the watched turtle.
     watch candidate
     while [mouse-down?] [
       ;; If we don't force the view to update, the user won't
       ;; be able to see the turtle moving around.
       display
       ;; The SUBJECT primitive reports the turtle being watched.
       ask subject [ setxy mouse-xcor mouse-ycor ]
     ]
     ;; Undoes the effects of WATCH.  Can be abbreviated RP.
     reset-perspective
   ]
```

```
  ]

end
```