A Genetic Algorithm Approach to Finding an Optimal Strategy for a Folk Dice Game

David Ebert

Tarleton State University

January 3, 2017

How to Play Fargo

A player begins her turn by rolling 10 dice. Dice are scored as follows:

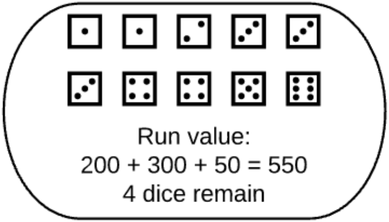
1. Three of a kind n is worth $100n$ points, except three 1's count as 1000 points.

How to Play Fargo

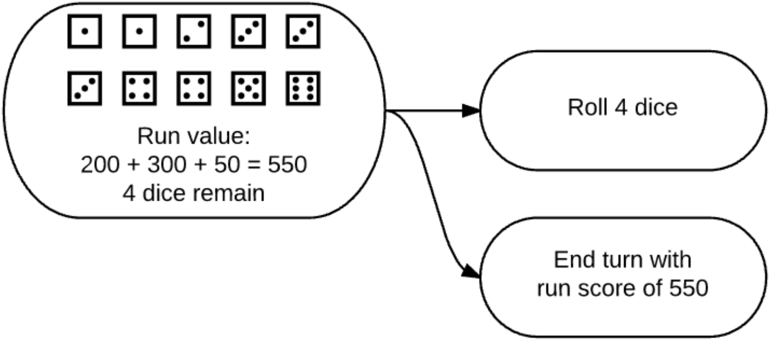
If a run is continued indefinitely, it will end in one of two ways:

1. If 0 points are added to a run's score after a re-roll, then the entire run is worth 0 and the turn is ended.
2. If a run ends by running out of dice, then the run's score is added to that player's score and the player begins a new run of 10 dice.

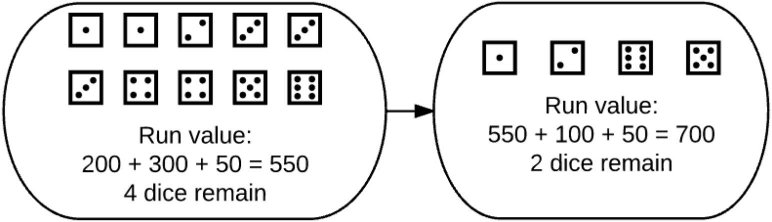
Example Turn



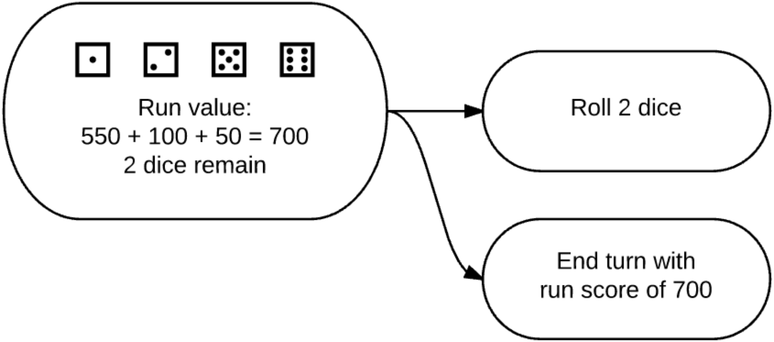
Example Turn



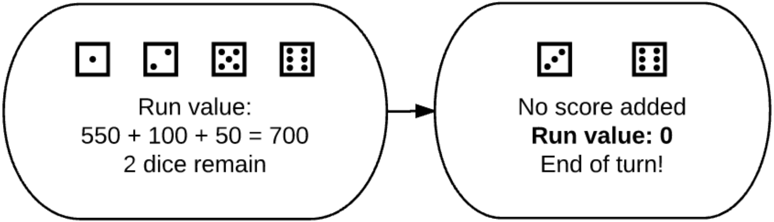
Example Turn



Example Turn



Example Turn



How (not) to Play Fargo

Endgame: After a player ends his or her turn with 10,000 or more points, all other players may take one more turn, then the player with the highest score wins.

Final Scores	
Adam	10,750
Mary	8,300
Mikaela	9,250
Parker	1,750
Steph	6,100
Me	**0**

Questions

- How many strategies are there?
- What's the expected value of a strategy?
- Which strategy is the best?

Strategy Space

Two Observations:

- If a reasonable strategy continues rolling n dice and p points, then it should also continue rolling with n dice and $< p$ points.
- A player should always roll 9 or 10 dice, since it's impossible to lose.

Strategy Space

Dice Remaining	Minimum Score Possible	Maximum Score Possible
1	450	3000
2	400	2200
3	350	2100
4	300	2000
5	250	1200
6	200	1100
7	150	1000
8	100	200

Strategy Space

Conclusion: All reasonable Fargo strategies can be written as a list of $x_1, x_2, \ldots, x_8$, where x_i indicates that with i dice remaining a player should continue rolling unless their score is at least x_i.

Strategy Space

Example:
Consider the strategy vector

$$[450, 600, 700, 650, 500, 1000, 1000, 250]$$

With one die, always stop rolling
With 2 dice, keep rolling unless the run is worth at least 600
With 3 dice, keep rolling unless the run is worth at least 700
⋮
With 7 dice, keep rolling unless the run is worth at least 1000
With 8 dice, always keep rolling

Strategy Space

i	$\min(x_i)$	$\max(x_i)$	$\text{count}(x_i)$
1	450	3050	52
2	400	2250	37
3	350	2150	36
4	300	2050	35
5	250	1250	20
6	200	1150	19
7	150	1050	18
8	100	250	4

Total number of *reasonable* strategy vectors:

$$\prod_{i=1}^{8} \text{count}(x_i s) = 66327206400 > 66 \text{ billion}$$

Questions

- ☐ What's the expected value of a strategy?
- ☐ Which strategy is the best?

Expected Value

Algorithm 1 Pseudocode for expected Value of Fargo given a strategy vector

```
function FINDEV(strategyVector)
    expectedValue ← 0
    probRepeat ← 0
    MANAGER(ndice = 10, strategyVector)
    return expectedValue/(1 − probRepeat)
end function

function MANAGER(ndice, strategyVector, prob = 1, soft = 0)
    if continueRolling is True then
        for result in resultDict do
            ROLL(ndice, result, prob, soft)
        end for
    else
        expectedValue ← expectedValue + soft ∗ prob
    end if
end function

function ROLL(ndice, result, prob = 1, soft = 0)
    ndice, prob, soft ← result
    if ndice == 0 then
        expectedValue ← expectedValue + soft ∗ prob
        probRepeat ← probRepeat + prob
    else
        MANAGER(ndice, strategyVector, prob, soft)
    end if
end function
```

Expected Value

$$\begin{aligned}\mathrm{EV}(\mathrm{Turn}|\mathrm{Strategy}) = {} & \Sigma \mathrm{Outcome\ Score} \times \mathrm{P}(\mathrm{Outcome}) \\ & + \mathrm{EV}(\mathrm{Turn}|\mathrm{Strategy}) \times \mathrm{P}(\mathrm{Repeated\ Run})\end{aligned}$$

$$\mathrm{EV}(\mathrm{Turn}|\mathrm{Strategy}) = \frac{\Sigma \mathrm{Outcome\ Score} \times \mathrm{P}(\mathrm{Outcome})}{1 - \mathrm{P}(\mathrm{Repeated\ Run})}$$

Using a recursive function, it is easy to find a strategy vector's expected value.

Questions

- How many strategies are there?
- What's the expected value of a strategy?
- Which strategy is the best?

Genetic Algorithm

- Introduced by John Holland in the 1970's to explore large solution spaces by mimicking the process of natural selection.
- Applications include the vehicle routing problem, 3D simulated muscles, wind turbine placement, machine learning, & spacecraft antennae.

Genetic Algorithm

3 ingredients:

1. Problem encoding (strategy vectors)
2. Evaluation function (expected value)
3. Rules for genetic succession

Genetic Algorithm

Generation 1:

Select 1000 random strategy vectors (starting population)

Generation 2:

1. Sort 1000 strategies from generation 1 by EV
2. Keep the best 250 strategies (survival of fittest)
3. Add 20 random strategies (2% genetic diversity)
4. Randomly combine strategies until there are 1000 strategies (genetic recombination)
5. Randomly change 10 entries (1% mutation)

Genetic Algorithm

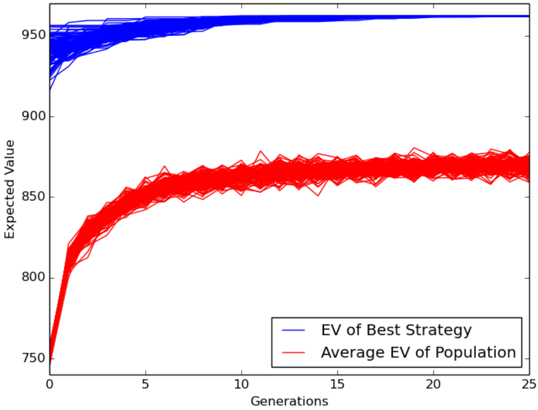
Generation 1:

Select 1000 random strategy vectors (starting population)

Generation $n > 1$:

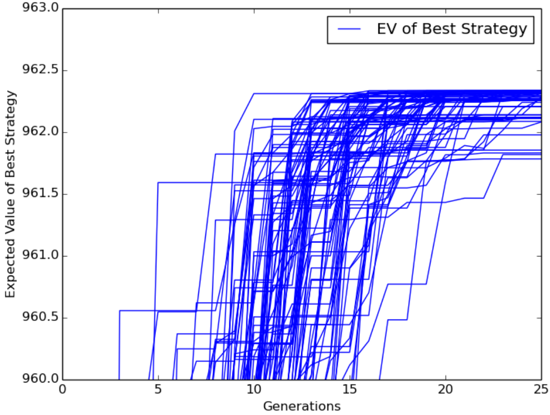
1. Sort 1000 strategies from generation $n - 1$ by EV
2. Keep the best 250 strategies (survival of fittest)
3. Add 20 random strategies (2% genetic diversity)
4. Randomly combine strategies until there are 1000 strategies (genetic recombination)
5. Randomly change 10 entries (1% mutation)

Results



Results from 100 trials with a population of 1000 strategies over 25 generations.

Results



Results from 100 trials with a population of 1000 strategies over 25 generations.

Results

- Maximum EV of 962.3343332342337 attained by 8/100 trials with the following strategy vector:

  [550, 400, 550, 1150, 1250, 1150, 1050, 250]

- Among the highest-EV vectors from each trial, the mean vector is 2.42 *steps* away from the best strategy, and the farthest vector was 7 *steps* away.

Results

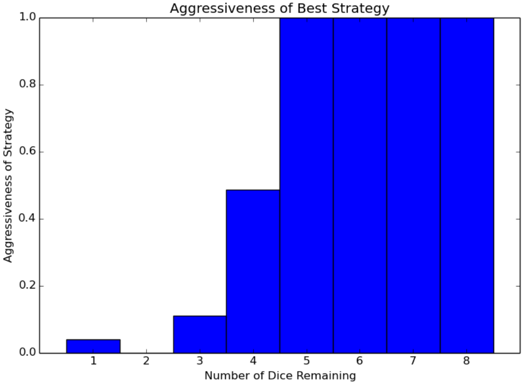
Aggressiveness:
For each strategy vector $x_1, \ldots, x_8$, define the *aggressiveness* of the vector as $a_1, \ldots, a_8$, where each a_i denotes the fraction of possible values for x_i that are smaller than x_i.

Strategy Space

i	$\min(x_i)$	$\max(x_i)$		x_i from Best Strategy	Aggressivenes
1	450	3050		550	$2/52$
2	400	2250		400	0
3	350	2150		550	$4/36$
4	300	2050		1150	$17/35$
5	250	1250		1250	1
6	200	1150		1150	1
7	150	1050		1050	1
8	100	250		250	1

Aggressiveness of [550, 400, 550, 1150, 1250, 1150, 1050, 250] strategy vector, which yielded the highest EV of 962.3343332342337.

Results



Aggressiveness of [550, 400, 550, 1150, 1250, 1150, 1050, 250] strategy vector, which yielded the highest EV of 962.3343332342337.

Questions

- How many strategies are there?
- What's the expected value of a strategy?
- Which strategy is (probably) the best?

Conclusions and Future Work

- The genetic algorithm efficiently and consistently yields a viable strategy.
- More work is necessary to confirm the optimal strategy and make the genetic algorithm more efficient.
- The genetic algorithm and EV algorithm are likely to extend to further analyses of Fargo, including the multi-player game and end-game strategies.

Questions?

- Python code repository: `https://github.com/dpebert7/fargo`
- `david.ebert@go.tarleton.edu`
- Special thanks to Dr. Jesse Crawford for his insight and inspiration