Breaking Even:

Applying Genetic Algorithms to Weekly Fantasy Football

Greg Filla

CSC 578

Topic Introduction:

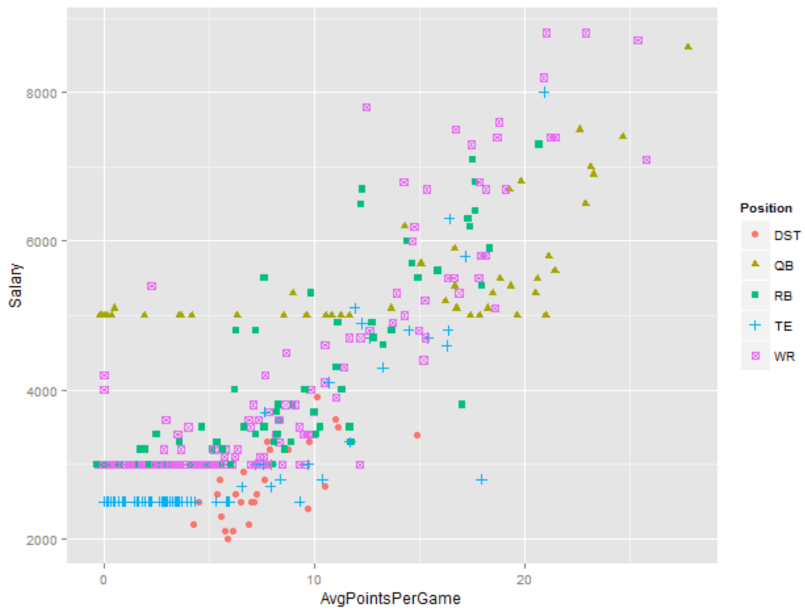
As long as predictive models and machine learning have been studied, there has been an interest in its application to sports prediction. While predicting winners for traditional sports gambling has proven to be a very difficult task, recent skill based "fantasy sports" games have potentially given an edge to players who can pick a team of high performing players on a given week. Weekly fantasy football is a relatively new game concept that has gained tremendous popularity since websites began offering the game format in 2014 (namely FanDuel and DraftKings).

Traditional fantasy football involves a user choosing a set of NFL players for their team that they will "own" for the entire season. Players can be added or dropped from the team through trades with other players in their league or through picking up players that were not already owned. The point of fantasy sports is to score the most fantasy points on a given week. Fantasy points are awarded to players when they complete some positive football action (e.g. throw for a touchdown, complete a reception). The advent of weekly fantasy football coupled with the legality of gambling on these "skill based games" has potentially created an opportunity to apply machine learning techniques with an immediate financial performance metric.

Executive Summary:

This report explains the methodology of applying a genetic algorithm to the problem of choosing the optimal fantasy football team lineup. The goal of this paper is to determine if genetic algorithms can capture the optimal team lineup, and what factors can be used to improve the performance of the selected team. The platform being used for validation of performance is DraftKings. On this site, a line-up consists of 9 players chosen from 5 categories of positions.

Each player is assigned a price on DraftKings and the salary cap for a line-up is $50,000. Players typically have a price ranging from $2,000 to $8,000. The plot below shows the positive correlation between average points per game and a player's salary.



It is obvious that the higher the average points, the higher the salary, but is it possible to capture the optimal selection of players for the budget? The selected lineups will be entered in two types of contests. In one, the top 50% of teams will win and will double the initial entry fee as a prize. The other competition has a different payout for different places with higher payouts for the better line up performance relative to other lineups.

Genetic Algorithms Explained:

Genetic algorithms (GAs) attempt to mimic natural selection and characteristics of a species' evolution to solve some problem. In contrast to other machine learning techniques where a hypothesis space is searched from simple to complex, GAs approach to searching the hypothesis space starts randomly and "evolves" by taking components of the best performing hypotheses from each "generation" or epoch of the algorithm. These algorithms have been applied to a number of optimization problems and also perform well for programming applications and for problems with a very large hypothesis space. The following explanation of genetic algorithms is referencing and supported by the chapter on Genetic Algorithms in Tom Mitchell's book "Machine Learning".

In order for a GA to determine the best hypothesis from each generation, a function is required to determine the fitness level. The fitness of a hypothesis can vary greatly depending on the domain of the problem being studied. To give a concrete example of a fitness function, Code 2 in the methodology section was the function used for this project. A fitness function should penalize a hypothesis proportionately to the error made for a given parameter. An example of this is the restriction for the lineup to have a max salary of $50,000. This fitness function returned a penalty value of the absolute difference between the salary of the lineup selected and this maximum value. The intention of using a fitness function is to find the best performing hypotheses so that they can be used for future generations.

In addition to the fitness function, a GA requires other parameters to be set before it can be used. The size of the population must be selected, which is number of chromosomes that will be evaluated during each generation or iteration. Another parameter is the fraction of the population that will be replaced at the Crossover step for each generation. Additionally a value is needed for the mutation level for any mutations of chromosomes between generations.

For each generation (similar to epochs in a neural network), a number of steps are taken to evaluate hypotheses and complete a thorough search of the hypothesis space. First, select members of the population of hypothesis to be considered for the current generation. This is done probabilistically using Formula 1, provided by Mitchell in “Machine Learning”.

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

Formula 1

This formula shows that the probability of picking a given hypothesis (hi) is directly proportional to the fitness value of that hypothesis compared to the sum of fitness values for each hypothesis in the population. Think of this step as “survival of the fittest”.

The next step of a GA is the Crossover. This is the step of the algorithm were fit hypotheses “breed” with each other to produce the next generation of hypotheses. Crossover can be completed in a number of methods; the following is a brief explanation of possible methods. Each method of crossovers uses a crossover mask, which is a binary string that determines which values will be crossed from each parent to each offspring hypothesis. A single point crossover selects a single point in each of the parent hypotheses and creates an offspring by switching the binary strings at this point. This can also be done by selecting two points in the parent hypotheses where the binary strings can be switched; this is a Two-point crossover. A uniform crossover is when a set of points are chosen for a crossover where points are uniformly crossed between parents according to the crossover mask.

The next step in a typical GA is to introduce some level of mutation. This step can be ignored if desired but it does introduce an element of randomness to the hypothesis search. In a GA, a mutation selects a certain random segment of each hypothesis (the size of which is set as a parameter) and the binary representation is inverted. This can be thought of how DNA is mutated in biological species. The more mutations introduced in a GA can widen the search for the optimal hypothesis, but too much mutation can lead to possibly miss a globally optimized hypothesis.

After crossover and mutations occur (if desired), the previous population of hypotheses is updated with the current generation. Next, each hypothesis is evaluated in accordance with the fitness function. It is at this point where the fittest chromosome can be observed for the current generation. This process is then iterated for either a set number of generations or until some termination criterion is met.

Methodology:

As with all machine learning problems, the first step in the process was to understand fantasy football and how the game is played. This step of the process is typically referred to as domain understanding or knowledge. For this project, it was important to understand weekly fantasy football rules and requirements, as well as understanding concepts in football like good and bad matchups for different players. This domain knowledge would play a crucial rule in later stages of this project.

Once the domain is understood, data needs to be collected to be used for the GA to search through. The method of GA use in this project is relatively simple, so the number of attributes required is very small and the number of records is finite. The attributes required are each player’s name, position, DraftKings salary for the current week, and projected points. The number of records required is the number of players available for inclusion in lineups for the current week. The data source for the player name, position, and salary was a CSV file easily accessible on the DraftKings website. For projected points, several different sources were considered. Code 1 is the Python code used to scrape player information and fantasy projected points, organize them in a data frame, and write them to a CSV. This was easily completed using a single method from the Pandas module for Python.

```
import pandas as pd

tables = []
for i in range(0,1000,40):
    u =
'http://games.espn.go.com/ffl/tools/projections?&startInd
ex='+str(i)
    raw_table = pd.io.html.read_html(u)[0]
    clean_data = raw_table.iloc[2:,0:14]
    tables.append(clean_data)
full_df = pd.concat(tables)

full_df.fillna(0,inplace = True)
full_df.replace(['\xa0'],[' '],inplace = True)

full_df.to_csv('espn_projected.csv', encoding= 'utf-8')
```

Code 1

ESPN data was the easiest to automate with regards to collection. Due to this, only ESPN's projected points were considered for the initial execution of the algorithm, later multiple sources were used. This will be explained later in this section.

Code 2 is a section of R code used to load data and create player indices for each of the 5 position types. These indices will later be used in the fitness function used by the GA.

```
#create datafame includding name, position, salary, and
projected points
df <- read.csv("score-salary-position-11-10.csv")
#Get indices of player positions
TE_ind <- which(df[4]=='TE')
QB_ind <- which(df[4]=='QB')
RB_ind <- which(df[4]=='RB')
WR_ind <- which(df[4]=='WR')
```

```
DEF_ind <- which(df[4]=='Defense')
```

Code 2

As mentioned in the explanation of genetic algorithms, a fitness function was required for this project. This function is shown in Code 3.

```
evalFunc <- function(x) {
  team_ind <- which(x==1)
  current_solution_points <- sum(df[team_ind,2])
  current_solution_weight <- sum(df[team_ind,3])

   if(current_solution_weight > sal.limit)
return(abs(50000 - current_solution_weight))
   if(sum(x) != 9 )  return(100)
   if(sum(x[QB_ind]) >1|| sum(x[DEF_ind]) >1 )
return(sum(x[DEF_ind])*100 +sum(x[QB_ind]) )
     if(sum(x[TE_ind]) == 2 && (sum(x[RB_ind]) >2 ||
sum(x[WR_ind]) >3)) {return(sum(x[TE_ind])*50)}
   if(sum(x[WR_ind]) == 4 && (sum(x[RB_ind]) >2 ||
sum(x[TE_ind]) >1)) {return(sum(x[WR_ind])*50)}
   if(sum(x[WR_ind]) > 4  || (sum(x[RB_ind]) >3 ||
sum(x[TE_ind]) >2)) {return(50)}
   return(-current_solution_points)

 }
```

Code 3

Although Code 3 was written in R, an explanation of the code should be relatively easy to understand once understanding the restrictions on team line-ups imposed by the validation platform, DraftKings. The fitness function accepts one input, an array of binary values the length of which is equal to the number of players available for drafting in a given week. A value of 1 means the player is included in the lineup, and a value of 0 means the player is not selected. When a chromosome (binary input array) is input in the fitness function, a series of tests are performed. If the chromosome fails any of the tests, the returned value is a penalty relative to the failure.

Code 3 shows accessing the player information for the values of 1 in the input array by accessing the index of "1" values in the array and matching them with the index of players in a data frame (matrix). If the total salary is greater than $50,000 (the DraftKings limit), then the value returned from the fitness function is the absolute value difference of the salary associated with the chromosome and 50,000. This dynamic feature of the fitness function penalizes teams less harshly when they are closer to the salary cap.

The following fitness checks in the function serve the purpose of making sure that the player positions on the team match the requirements put in place by DraftKings. Each check on the chromosome's fitness, including the salary check, is done in order of importance. The purpose of this is to quickly discard lineups that may have some correct components but would not meet finer requirements (e.g. a team with a salary of $300,000 should be thrown out immediately even if the positions are correct).

Once salary is checked, the function confirms that only 9 players (or 1s) are input in the array. If that requirement is met then the fitness function evaluates the count of players in each position. One player in the lineup is a "Flex" which means it can be a wide receiver (WR), running back (RB), or tight end (TE). This was accounted for by adding additional conditional statements to check the count of other positions while also checking the current count of WR, RB, or TE.

With the data collected and fitness function in place, there are a couple parameters that need to be set prior to the GA execution. One parameter set as a constant is the salary cap of $50,000. This setting interesting because if there was a certain player desired for the lineup, this constant could be reduced by the amount of that player's salary and the fitness function could be modified to do a search to optimize points over the remaining salary. Mutation for the GA was set to be a value of 0.01, which is close to the default, recommended level for this parameter from the package's documentation. The other parameters will be adjusted over different versions of the GA. The following section provides interim optimal lineups coupled with explanations of parameter adjustments.

Iterative Genetic Algorithm Model Selection:

**Player points, positions, and names came from week 10 of the 2015 NFL season*

After compiling the data sources and preparing the fitness function, it is possible to begin lineup selections. Prior to executing the GA, players with projected fantasy points of zero were removed from the CSV file to improve the efficiency of the GA. The fitness function used for this initial lineup was penalizing harshly when a hypothesis went over the salary cap. Prior to this version of the GA, the fitness function was returning 0 as a penalty for all failures to meet the parameter requirements (this implementation is seeking to minimize the score so points were returned as negative values). This version set a very high positive value to be returned when a lineup went over the salary cap.

```
             PLAYER   PTS    Salary Position
83    Richard Rodgers  6.5   3000       TE
143              Jets  9.6   3000  Defense
152  Michael Crabtree 10.6   5800       WR
160  Jonathan Stewart 11.1   4300       RB
164 LeGarrette Blount 11.4   4900       RB
169  Demaryius Thomas 12.0   7400       WR
173        A.J. Green 12.7   7600       WR
192 DeAngelo Williams 15.5   6500       RB
204     Aaron Rodgers 22.8   7500       QB
> sum(df[best_ind,2])
[1] 112.20
> sum(df[best_ind,3])
[1] 50000
```

This proved that the fitness function worked to select players for the right positions and stay within the salary cap, but is a higher point value possible? In the following lineup, the fitness function was adjusted. The hypothesis was penalized based on the absolute error between the salary and the salary cap.

```
                 PLAYER  PTS Salary Position
116        Jordan Reed  8.1   4600       TE
138    Brandon LaFell  9.1   4100       WR
143               Jets  9.6   3000  Defense
152 Michael Crabtree 10.6   5800       WR
153   Giovani Bernard 10.6   4700       RB
160 Jonathan Stewart 11.1   4300       RB
171    Julian Edelman 12.2   8200       WR
197        Todd Gurley 17.2   7300       RB
204     Aaron Rodgers 22.8   7500       QB
> sum(df[best_ind,2])
[1] 111.30
> sum(df[best_ind,3])
[1] 49500
```

Although the points and salary were a bit worse than the original trial, it intuitively made more sense to keep the penalty for the salary cap to be dynamic rather than a constant error figure. In the following lineup, the number of generations was increased to be 500 from 400. This lineup also had a population size reduced from 500 to 300.

```
                 PLAYER  PTS Salary Position
127     Gary Barnidge  8.6   4800       TE
138    Brandon LaFell  9.1   4100       WR
143               Jets  9.6   3000  Defense
152 Michael Crabtree 10.6   5800       WR
171    Julian Edelman 12.2   8200       WR
178   Darren McFadden 13.2   4900       RB
184    Justin Forsett 14.0   6000       RB
197        Todd Gurley 17.2   7300       RB
200     Blake Bortles 17.6   5600       QB
> sum(df[best_ind,2])
[1] 112.10
> sum(df[best_ind,3])
[1] 49700
```

These adjustments are resulting in different player combinations, but not a significant boost to points per lineup. For experimentation, the number of generations increased to be 1000 from 500 and the size of the population was reduced from 300 to 200.

```
                PLAYER  PTS Salary Position
114      Sammy Watkins  8.1   5000       WR
116        Jordan Reed  8.1   4600       TE
138    Brandon LaFell  9.1   4100       WR
143               Jets  9.6   3000  Defense
160  Jonathan Stewart 11.1   4300       RB
173        A.J. Green 12.7   7600       WR
192 DeAngelo Williams 15.5   6500       RB
197       Todd Gurley 17.2   7300       RB
204     Aaron Rodgers 22.8   7500       QB
> sum(df[best_ind,2])
[1] 114.20
> sum(df[best_ind,3])
[1] 49900
```

Once again, there was a very marginal improvement on points. Since these changes were not resulting in significant changes, the mutation level was adjusted from 0.01 to 0.05. This is to increase the search area and increase the chance of mutations of the genetic algorithm. All other settings remained constant.

```
           PLAYER  PTS Salary Position
58  Darren Sproles  5.5   3600       RB
105  Pierre Garcon  7.5   4800       WR
112   Willie Snead  8.0   4900       WR
127  Gary Barnidge  8.6   4800       TE
156 Marshawn Lynch 10.8   6700       RB
165           Rams 11.8   3600  Defense
173     A.J. Green 12.7   7600       WR
196    Eli Manning 15.9   6700       QB
197    Todd Gurley 17.2   7300       RB
> sum(df[best_ind,2])
[1] 98.00
> sum(df[best_ind,3])
[1] 50000
```

Increasing the mutation parameter let to significantly worse performance with regards to total points, although salary was optimized at $50,000. The ‘genalg’ documentation has a suggested mutation level that resembles Formula 2.

$$m = \frac{1}{(len(h) + 1)}$$

Formula 2

Based on the other settings for this GA, the level for m was set to 0.002.

```
               PLAYER  PTS Salary Position
83  Richard Rodgers  6.5   3000       TE
129 Antonio Andrews  8.6   3700       RB
133     Eric Decker  9.0   5300       WR
135        Panthers  9.1   3300  Defense
138  Brandon LaFell  9.1   4100       WR
171  Julian Edelman 12.2   8200       WR
184  Justin Forsett 14.0   6000       RB
197     Todd Gurley 17.2   7300       RB
200   Blake Bortles 17.6   5600       QB
> sum(df[best_ind,2])
[1] 103.3
> sum(df[best_ind,3])
[1] 46500
```

This also performed more poorly when compared other tests point values. It seems that 0.01 for a mutation value returns the best results.

The prior sets of lineups were all set using ESPN for the source of projected fantasy points. Now the performance of GA will be tested with adjusted weight values based on internet start and sit recommendations from a popular analyst, Michael Fabiano of the NFL. If a player was a "start of the week", their projected points were weighted by 1.5, and if they were just recommended to start their points were weighted by 1.25. If a player was recommended to be sat for the current week their points were multiplied by 0.75. Since the GA only cares about maximizing points and staying in the salary cap, these changes can be made quickly to the input. The following lineup has a much higher points total (136.53) but this is due to the points adjustments.

```
                 PLAYER NewPts Salary Position
6       Allen Robinson 16.500   6700       WR
8       Alshon Jeffery 11.300   7100       WR
20      Brandon LaFell  9.100   4100       WR
58     Darren McFadden 13.200   4900       RB
64  DeAngelo Williams 23.250   6500       RB
74         Jordan Reed 10.125   4600       TE
93                Jets  9.600   3000  Defense
139    Justin Forsett 17.500   6000       RB
177        Cam Newton 25.950   7000       QB
> sum(df[best_ind,2])
[1] 136.525
> sum(df[best_ind,3])
[1] 49900
```

Using this adjusted dataset, the parameters on the GA were adjusted to a population size of 500 hypotheses and 500 iterations (from 1,000 and 200 respectively). This change shows an even better performance.

```
                 PLAYER NewPts Salary Position
6       Allen Robinson  16.50   6700       WR
20      Brandon LaFell   9.10   4100       WR
44       Gary Barnidge  12.90   4800       TE
58     Darren McFadden  13.20   4900       RB
64  DeAngelo Williams  23.25   6500       RB
67      DeMarco Murray  17.00   6200       RB
118               Rams  14.75   3600  Defense
124  Michael Crabtree  10.60   5800       WR
177         Cam Newton  25.95   7000       QB
> sum(df[best_ind,2])
[1] 143.25
> sum(df[best_ind,3])
[1] 49600
```

Finally, two additional lineups were created after completing further research and incorporating points from NFL.com. Similar to the previous adjustments, points were increased or decreased in the dataset based on the new projected point values. Once again, these lineups have high projected point values.

```
                 PLAYER NewPts Salary Position
2         Andy Dalton  23.75   6500       QB
3  DeAngelo Williams  23.25   6500       RB
11     Justin Forsett  17.50   6000       RB
14     Allen Robinson  16.50   6700       WR
23           Panthers  13.65   3300  Defense
26    Darren McFadden  13.20   4900       RB
30      Gary Barnidge  12.90   4800       TE
46       Randall Cobb  11.20   6700       WR
73     Brandon LaFell   9.10   4100       WR
> sum(df[best_ind,2])
[1] 141.05
> sum(df[best_ind,3])
[1] 49500
```

```
                 PLAYER NewPts Salary Position
10      Allen Robinson 16.500   6700       WR
50     Darren McFadden 13.200   4900       RB
54  DeAngelo Williams 23.250   6500       RB
55      Delanie Walker 12.000   4700       TE
58          Derek Carr 21.625   5800       QB
93     Jeremy Langford 15.000   4800       RB
106     Julian Edelman 12.200   8200       WR
```

```
148          Panthers 13.650   3300  Defense
171      Stefon Diggs 13.200   5100       WR
> sum(df[best_ind,2])
[1] 140.625
> sum(df[best_ind,3])
[1] 50000
```

The output presented for the above lineups is the result of Code 4 below. This code uses the best chromosome or hypothesis from the GA and takes the index of '1' values to pull the players, positions, salaries and points from the main data frame. The final step is to transfer this lineup into a contest on DraftKings.

```
   GAmodel <- rbga.bin(size = player_size, popSize = 500,
iters = iter, mutationChance = 0.01,
                       elitism = 20, evalFunc = evalFunc)

   bestSolution<-
GAmodel$population[which.min(GAmodel$evaluations),]
   best_ind <- which(bestSolution==1)
   df[best_ind,]
   sum(df[best_ind,2])
   sum(df[best_ind,3])
```

Code 4

Results:

An example of one execution of the GA is shown in Figure 1. This plot shows the mean evaluation score (fitness value) for each generation. The GA seeks to minimize this value. The very early generations' mean values are high while the GA is learning that there is a salary cap and going over it creates a penalty. Those hypotheses are soon cut out of the population due to their poor fitness and stronger hypotheses remain.

Figure 1 is one component of proof that the GA performed to accomplish what was intended. The fitness level is quickly optimized according to the function used to evaluate each lineup. One other component of testing the success of the GA and the quality of the fitness function was to check the quality of lineups. The prior section demonstrated that the best lineup consistently remained under the salary cap of $50,000 and met the requirements for having players at each position. The original research question was if a GA could complete this task, which has been confirmed in this experiment.

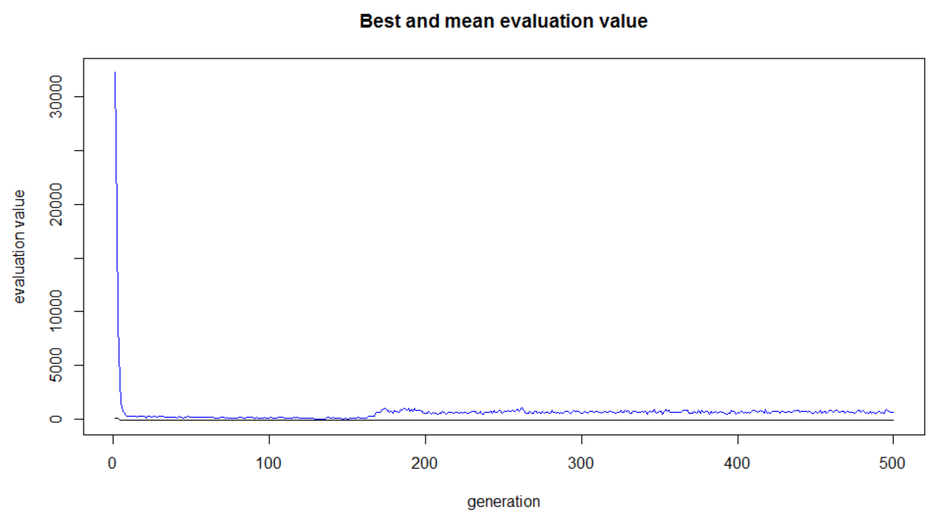


Figure 1

The other question and component for evaluation were how the lineups performed in the contests on DraftKings and the actual points scored. Figure 2 demonstrates the performance of many different lineups that were shown in the previous section. In this figure, the horizontal bar indicates how the lineup was scoring compared to other teams in the contest. When a lineup has higher relative points it is shown farther right on this horizontal bar. The contest with 6 entries was populated with lineups generated using solely ESPN projected points. As can be inferred from this figure, these lineups were unsuccessful in this contest. The detailed view for these lineups can be seen in the appendix.

The first two contests shown in Figure 2 had the best performing lineup found for this project. This lineup was chosen by consolidated ESPN projected points, NFL projected points, and additional research done manually (using Start 'em and Sit 'em guides). The detailed view of this lineup is shown in the appendix. Even though this lineup was successful, Figure 3 demonstrates that it could barely place in a position where it would win. This shows that the actual performance of the lineups on DraftKings still needs to be improved.

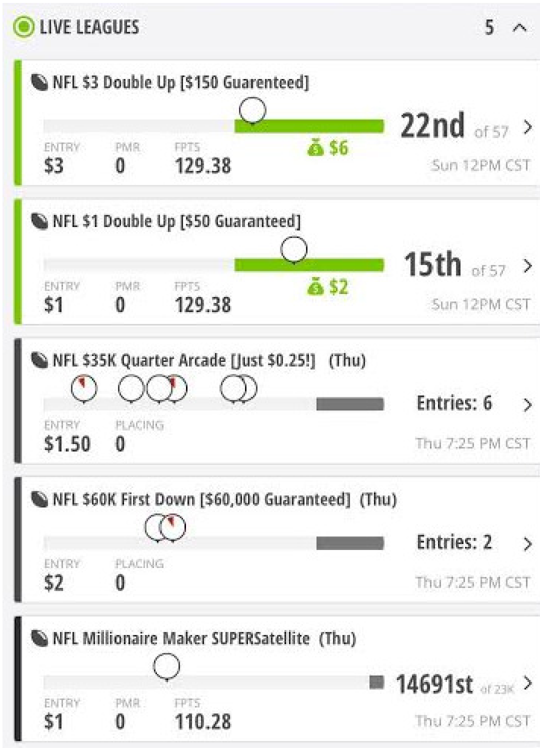


Figure 2

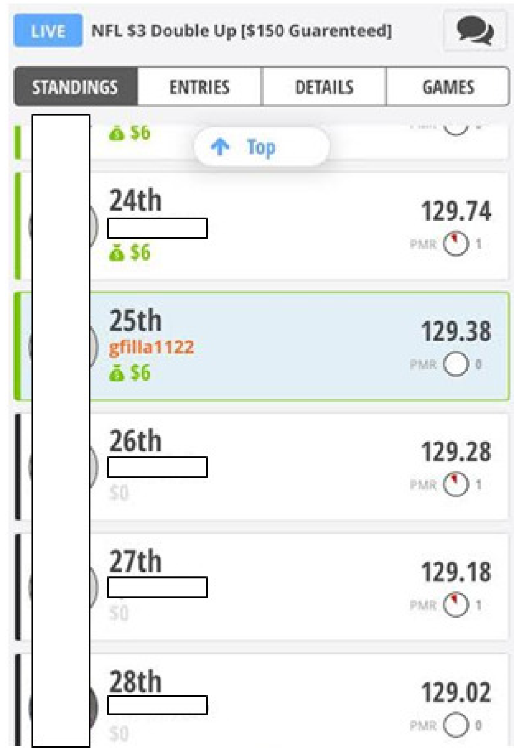


Figure 3

Final Analysis and Conclusions:

Using genetic algorithms to select optimal team lineups for weekly fantasy football is possible but it comes with a caveat. GAs are a powerful, flexible tool that can pick the optimal lineup based on the data provided. The caveat to the success is that the GA is only as successful in picking a winning lineup as the projected points used. To further improve this project, a custom defined point value could be implemented that would aggregate fantasy knowledge from multiple sources to make sure the lineup was truly optimized. This enhancement can be executed by focusing on automating and expanding the data collection required for this project. Another interesting expansion or enhancement is to use a similar technique on other sports rather than football. Since the GA only cares about salary and points, it could easily be applied to other sports by adjusting the fitness function to select the right positions and number of players. Overall, this application of genetic algorithms is successful in regards to execution. The limitation experienced with this system is directly proportionate to the quality of data used to choose lineups.

References:

Fabiano, Michael. "Start 'Em, Sit 'Em".
http://www.nfl.com/news/author?id=09000d5d800219d7

Mitchell, T. (1997). Genetic Algorithms. In *Machine Learning*. New York: McGraw-Hill.

Willighagen, E & Ballings, M (2015). R Package 'genalg'. https://cran.r-project.org/web/packages/genalg/genalg.pdf

Appendix: DraftKings Results

Lineups selected using only ESPN projected points

NFL — LIVE — # OF ENTRIES: 3 — PMR: 0 — REM. SALARY: $400

Pos	Player	FPTS
QB	Cam Newton	20.98
RB	DeMarco Murray	17.9
RB	D. Williams	7.9
WR	Michael Crabtree	9.5
WR	Brandon LaFell	8.6
WR	Allen Robinson	16.1
TE	Gary Barnidge	18.5
FLEX	Darren McFadden	8.8
DST	Rams	2

TOTAL FANTASY POINTS: 110.28

LAST EDIT: NOV 12 2015 5:34PM EST

EDIT ENTRIES

NFL — LIVE — # OF ENTRIES: 2 — PMR: 60 — REM. SALARY: $0

Pos	Player	FPTS
QB	Aaron Rodgers	25.12
RB	L. Blount	15.7
RB	D. Williams	7.9
WR	Michael Crabtree	9.5
WR	A.J. Green	0
WR	Demaryius Thomas	14.1
TE	Richard Rodgers	14.2
FLEX	Jonathan Stewart	15.1
DST	Jets	5

TOTAL FANTASY POINTS: 106.62

LAST EDIT: NOV 12 2015 5:42PM EST

EDIT ENTRIES

NFL — LIVE — # OF ENTRIES: 1 — PMR: 60 — REM. SALARY: $500

Pos	Player	FPTS
QB	Aaron Rodgers	25.12
RB	Todd Gurley	17.9
RB	Jonathan Stewart	15.1
WR	Julian Edelman O	9.3
WR	Allen Hurns Q	17.2
WR	Brandon LaFell	8.6
TE	Jordan Reed	17.9
FLEX	Giovani Bernard	0
DST	Jets	5

TOTAL FANTASY POINTS: 116.12

LAST EDIT: NOV 12 2015 5:46PM EST

EDIT ENTRIES

NFL — LIVE — # OF ENTRIES: 1 — PMR: 60 — REM. SALARY: $0

Pos	Player	FPTS
QB	Eli Manning	25.44
RB	Todd Gurley	17.9
RB	Marshawn Lynch	12
WR	Pierre Garcon	3
WR	A.J. Green	0
WR	Willie Snead	0
TE	Gary Barnidge	18.5
FLEX	Darren Sproles	8.4
DST	Rams	2

TOTAL FANTASY POINTS: 87.24

LAST EDIT: NOV 12 2015 5:50PM EST

EDIT ENTRIES

NFL — LIVE — # OF ENTRIES: 1 — PMR: 0 — REM. SALARY: $300

Pos	Player	FPTS
QB	Cam Newton	20.98
RB	DeMarco Murray	17.9
RB	D. Williams	7.9
WR	Brandon LaFell	8.6
WR	Jarvis Landry	18.8
WR	Allen Robinson	16.1
TE	Jordan Reed	17.9
FLEX	Darren McFadden	8.8
DST	Jets	5

TOTAL FANTASY POINTS: 121.98

LAST EDIT: NOV 12 2015 5:40PM EST

EDIT ENTRIES

NFL — LIVE — # OF ENTRIES: 1 — PMR: 0 — REM. SALARY: $300

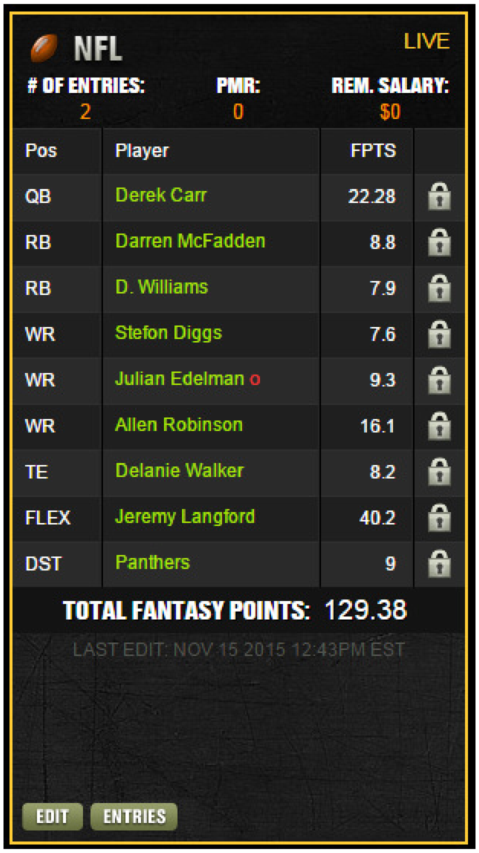
Pos	Player	FPTS
QB	Blake Bortles	17.02
RB	Justin Forsett	10.4
RB	Todd Gurley	17.9
WR	Michael Crabtree	9.5
WR	Julian Edelman O	9.3
WR	Brandon LaFell	8.6
TE	Gary Barnidge	18.5
FLEX	Darren McFadden	8.8
DST	Jets	5

TOTAL FANTASY POINTS: 105.02

LAST EDIT: NOV 12 2015 5:48PM EST

EDIT ENTRIES

Team selected using ESPN projected points, NFL projected points, and additional research material



NFL LIVE

\# OF ENTRIES: 2 PMR: 0 REM. SALARY: $0

Pos	Player	FPTS
QB	Derek Carr	22.28
RB	Darren McFadden	8.8
RB	D. Williams	7.9
WR	Stefon Diggs	7.6
WR	Julian Edelman o	9.3
WR	Allen Robinson	16.1
TE	Delanie Walker	8.2
FLEX	Jeremy Langford	40.2
DST	Panthers	9

TOTAL FANTASY POINTS: 129.38

LAST EDIT: NOV 15 2015 12:43PM EST

EDIT ENTRIES