Optimizing Fantasy Draft Choice

David Abrahams and Keenan Zucker

September 2015

1 Abstract

We tried to to optimize a fantasy football draft so an owner would pick the best possible player for their team each time it is their turn. We attempted doing this using game theory and game tree algorithms, similar to ones used in Chess and Checkers playing algorithms. We considered solving the tree with "minimax" and pruning it with "alpha-beta pruning." Even with these methods, our tree is too large, and solving it proves infeasible. Instead, we developed several picking algorithms and tested them against each other in order to determine a good, though likely not the best, strategy. We conclude that our "Weighted Value Over Replacement" algorithm is optimal.

2 What is Fantasy Football?

A fantasy football league normally consists of 10 teams. Before the NFL season starts, fantasy teams draft their own team, consisting of NFL players. A draft order is set by the league and all owners draft until their roster is filled. Once a player is drafted, he cannot be drafted by anyone else. A fantasy roster is comprised of 6 positions:

- 1 Quarterback
- 2 Running Backs
- 3 Wide Receivers
- 1 TE
- 1 Team Defense
- 1 Kicker
- 7 Bench players

Read tinyurl.com/qe8trlc for an explanation of what these positions mean.

Teams accumulate fantasy points based on the statistics of the real life players they are comprised of. The goal is to maximize the amount of fantasy points your team scores over the course of a season. Bench players do not contribute to a team's fantasy points, but players can be switched into and out of the bench on a week to week basis, and thus bench players still have value (just less than the starters).

3 The Question

We use pre-season projections to assign a point value to each player. Our question is: which algorithm does the best job of maximizing an entire drafted team's output, by making selections at each pick during the draft that take into account the different positional spots that need to be filled, and the value each player has for the team over the course of the season.

4 Assumptions

- Your team's "value" is a function of the total points your team scores, with starters being valued more highly.
- There are t number of teams with owners. (For this study, t=10 for 10 owners in the draft)
- There are p different positions a player can be. (For this study, $p = 6$. QB, RB, WR, TE, D/ST, K)

- Each team must draft $r_1 \dots r_p$ players of each position. (For us, $r = \{2, 5, 5, 2, 1, 1\}$, as each fantasy team will draft 2 QBs, 5 RBs, 5 WRs, 2 TEs, 1 K, 1 D/ST.)
- Each team will start $s_1 \dots s_p$ players of each position. (For us, $s = \{1, 2, 3, 1, 1, 1\}$, as each fantasy team will start 1 QBs, 2 RBs, 3 WRs, 1 TEs, 1 K, 1 D/ST.)
- The player pool consists of $n_1 \dots n_p$ players of each position type, with $n_{total} = \sum n_i$ total players.
- Each player in the pool at position i has a value $v_{i,1} \dots v_{i,n_i}$, where $v_{i,1}$ is the highest valued player at position i.

5 Decision Variable

At any given pick, your decision variable is: do I pick the best QB, RB, WR, TE, D/ST, or K available?

6 Objective Function

At the end of the draft, how many points will my team score, with starter being weighted most heavily, followed by players at the top of my bench, followed by players at the end of my bench?

$$V = \sum_{i=1}^{p} \sum_{j=1}^{n_i} v_{i,j} * (1 - 0.2 * \max(j - s_i, 0)) \tag{1}$$

Where p is the number of positions, n_i is the number of drafted players at position i, $v_{i,j}$ is the jth player on my roster at position i, and s_i is the number of starters for position i.

This summation equation iterates over a drafted team, summing the value of each player. However if we are evaluation a bench player, we decrease their value by 20% for each spot down on the bench. (Your first quarterback's value counts in full. Your second QB's value counts 80% as much. If you draft a third quarterback - which we assume you do not - his value would count 60% as much.)

7 Using a tree is impossible

7.1 The equations

We posit the best solution to this problem is to play out every single draft possibility, and work backward assuming each player will make the choice that maximizes his team's objective function. This is impossible, as this game tree (https://en.wikipedia.org/wiki/Game_tree) would be absurdly large.

First, let's come up with an expression for how large this tree could be. The largest possible tree would be if every player drafted based on need, and with p rounds left, they

had one slot at each position to fill. The expression, where m is how many nodes the last layer of this tree would have is:

$$m = \left((p!) * p^{r-p}\right)^t \tag{2}$$

Where r is the total number of players each team drafts, t is the number of teams, and p is the number of different positions.

The smallest possible tree would be if every player drafted a position player with the least number of roster slots at that position. Meaning, if they chose a Kicker first, they would never again have to choose a Kicker, so in every round subsequent the algorithm would only consider drafting from five positions, rather than six. If we order $r_1 \dots r_p$ in ascending order, so that r_1 is the smallest number of position slots (for our study, $r_1 = 1$, because you only draft one kicker – or one defense), then the expression, where m is how many nodes the last layer of this tree would have is:

$$m = (p^{r_1} * (p-1)^{r_2} * (p-2)^{r_3} * \dots * 1^{r_p})^t \tag{3}$$

Where r_i is the number of slots at a given position.

7.2 Explanations

For the largest possible tree, we can derive equation 2 by thinking about the draft happening in reverse. In the final round, each team will draft the player to fill the one remaining position slot it has. In the second to last round, each team has two possible picks, etc. In the first 10 rounds ($10 = r - p$, because $r = 16$ and $p = 6$) each team can pick from any one of the 6 positions.

For the smallest tree, we begin by thinking about the start of the draft. Each player keeps drafting at a position until that position is filled. For example, everyone fills their Kicker slot in the first round, so there are 6^{10} possibilities for the round, because no position slot has been filled yet. This is reflected in the $p^{r_1 t}$ term of our equation (6^{1*10}). In the next round, everyone's Kicker slot is filled, so they have 5 possible picks. The second round has 5^{10}, or $(p-1)^{r_2 t}$ possibilities. We continue this until we only have one slot to fill, which in our study would be either RB or WR.

7.3 Study Tree Sizes

In our study, the smallest number of choices, which is a very unrealistic draft scenario, would still be big. Like really big:

$$m = (6 * 5 * 4^2 * 3^2 * 2^5 * 1^5)^{10} \tag{4}$$

$$m \approx 2.5 * 10^{51} \tag{5}$$

The largest possible tree is huge. Like, really really huge. Imagine the number of grains of sand on Earth. Now, each grain is actually a copy of our universe. Take the total number

of atoms in that universe, for each individual grain of sand, and add it all together. That huge.

$$m = \left(6^{10} * 6!\right)^{10} \tag{6}$$

$$m \approx 2.4 * 10^{106} \tag{7}$$

It is impossible for us to evaluate every single possibility. Therefore, we must use a heuristic algorithm. This means that we cannot find the optimal solution, but we can find one that is an approximate solution by implementing picking algorithms.

8 Our Picking Algorithms

We knew we could not use the game tree approach to solve our problem, so instead, we decided to write several picking algorithms and compare them by playing them against each other in fantasy drafts.

- **Random** – Choose the best player at a random position
- **Weighted Random** – Choose the best player randomly, but weighted based on the total number of players you must draft at each position. (It is most likely to draft a RB or WR at a given pick.)
- **Worst Dropoff** – Choose the player that has the greatest difference between him and the next best player at his position. For example, for two positions with values: QB1: 200, QB2: 190, RB1: 170, RB2: 140, the algorithm would pick RB1 because the difference between him and the next best RB is larger (30) than between the QBs (10).
- **Worst Guaranteed** – Similar to the Worst Dropoff, except the difference is between the best player at the position and the next player at that position you are guaranteed to get at your next pick. Meaning, if you have the 1st pick, the next pick you have in the draft is not until pick 20. There are 18 picks before your next pick, so you are at worst going to get the 19th RB with your next pick. So the 1st RB's value is the difference between RB1 and RB19.
- **Value Over Replacement** – Similar again to the Worst Dropoff, except this time the difference was calculated based off of a predefined value for each position of what "replacement-level" was. We calculated replacement based on how many players at each position are predicted to be drafted in the first 100 picks using average draft position data from ESPN (http://games.espn.go.com/ffl/livedraftresults). Read more at http://tinyurl.com/7ddma.
- **Value Over Replacement Weighted** – Nearly identical to Value Over Replacement, except it accounts for the weighting used in our objective function (see Equation 1). This algorithm would rather fill its starters before filling its bench.

9 Results

To test our different picking algorithms, we decided to make a 'league' of 10 owners, and give the owners different strategies.

9.1 Control vs Weighted Control

	control_weighted 0	control 1	control_weighted 2	control 3	control_weighted 4	control 5	control_weighted 6	control 7	control_weighted 8	control 9
Starters										
qb1	329.6	278.8	303.2	295.6	278.9	258.8	249.1	281.2	298	325.2
rb1	138.3	238.6	193.5	226.2	218.9	167.9	214.5	237.6	157.6	214.2
rb2	122.3	188.6	166.9	116.9	202.6	158.4	173.2	232.3	153.1	154.7
wr1	181.7	114.2	189.6	134.8	192.4	212.6	210.2	167.2	203.4	202.3
wr2	177.2	113	133	121.1	135.7	171.9	150.8	137.5	165.9	139.6
wr3	149.7	103.7	121.1	119.5	129.7	110	124.9	119.3	159.4	123
te1	86.6	108.5	136.1	114.2	83.5	117.3	79.4	125.4	171.2	81.5
dst	113	126	187	137	132	124	109	117	111	114
k	127.3	129.6	123.3	129.6	119.3	130.8	139.9	120.6	120.5	128.3
Bench										
qb2	232.8	272.9	262.7	270.3	244.5	236.6	235	252.2	254	271.3
rb3	122.2	166.5	147.8	94	182.4	151.4	172.2	102.2	143.5	96.5
rb4	92.5	117.3	94	85.3	130	140	144.8	83.5	125.7	82.1
rb5	90.4	70.2	88.9	74.9	109.3	123.6	141.5	75.3	83.3	79
wr4	147.2	95.9	114.7	115.2	116	99.7	122.8	117.9	143.2	108.8
wr5	133.7	95.1	102.8	112.5	105.7	94.6	102.6	109.3	105.8	102.4
te2	85.4	92.1	74.1	92.5	74.4	82.2	79.4	89.2	74.2	81
score	**2084.74**	**2076.86**	**2201.6**	**2019.64**	**2186.86**	**2114.26**	**2159.44**	**2150.94**	**2219.08**	**2087.38**
Average Control:	**2089.816**	**Average Weighted Control:**	**2170.344**							

Figure 1: Control vs Weighted Control

This table shows our two 'control' algorithms in comparison. As shown, the Weighted Control had slightly better scores, by about 80 points. This makes sense, as while it is still random, adding the bias towards the positions that the owners team needs does indeed make a difference and a better algorithm.

9.2 Weighted Control vs Worst Dropoff vs Guaranteed Dropoff

	worst_dropoff 0	worst_guaranteed 1	control_weighted 2	worst_dropoff 3	worst_guaranteed 4	control_weighted 5	worst_dropoff 6	worst_guaranteed 7	control_weighted 8	control_weighted 9
Starters										
qb1	249.1	278.9	236.6	262.7	278.8	281.2	295.6	329.6	303.2	325.2
rb1	214.2	154.7	238.6	125.7	237.6	202.6	166.5	226.2	188.6	232.3
rb2	172.2	140	214.5	116.9	218.9	158.4	94	193.5	173.2	182.4
wr1	167.2	203.4	171.9	210.2	192.4	212.6	189.6	119.3	147.2	181.7
wr2	143.2	114.7	165.9	202.3	177.2	150.8	149.7	110	135.7	134.8
wr3	139.6	114.2	137.5	159.4	108.8	133.7	133	109.3	123	129.7
te1	108.5	171.2	79.4	136.1	86.6	117.3	125.4	92.5	81.5	81
dst	187	132	114	137	113	111	124	126	109	117
k	120.5	119.3	129.6	127.3	123.3	128.3	139.9	129.6	130.8	120.6
Bench										
qb2	244.5	258.8	232.8	254	270.3	235	252.2	298	271.3	272.9
rb3	138.3	130	157.6	90.4	153.1	151.4	83.3	166.9	147.8	167.9
rb4	102.2	109.3	141.5	82.1	143.5	117.3	75.3	144.8	122.2	94
rb5	85.3	96.5	88.9	79	123.6	92.5	70.2	122.3	83.5	74.9
wr4	105.7	105.8	122.8	113	102.8	121.1	119.5	102.4	117.9	124.9
wr5	94.6	103.7	121.1	95.1	95.9	102.6	112.5	99.7	115.2	116
te2	83.5	114.2	74.1	89.2	82.2	74.2	92.1	85.4	79.4	74.4
score	**2128**	**2104.68**	**2165.78**	**2070.64**	**2232.84**	**2145.04**	**2014.56**	**2170.86**	**2077.04**	**2187.62**
	Avg. Weighted Control:	2143.87								
	Avg. Worst Dropoff:	2071.07								
	Avg. Worst Guaranteed:	2169.46								

Figure 2: Weighted Control vs Worst Dropoff vs Guaranteed Dropoff

This table shows owners with three of each of the tested strategies. The Worst Guaranteed algorithm scores the best out of these three, which makes sense. That is a relatively valid strategy and comes into play in real drafting, as an owner has to think about what they are missing out on in the other picks between their own. The Worst Dropoff algorithm actually scores lower than the Weighted Control, which is quite interesting. It is actually better to choose a random position that basing your choice based off the difference between best and second best player.

9.3 Weighted Control vs Value Over Replacement vs Weighed Value Over Replacement

	vorp 0	pos_vorp 1	control_weighted 2	vorp 3	pos_vorp 4	control_weighted 5	vorp 6	pos_vorp 7	control_weighted 8	control_weighted 9
Starters										
qb1	325.2	295.6	329.6	278.8	272.9	298	235	271.3	244.5	303.2
rb1	238.6	237.6	232.3	226.2	218.9	188.6	214.5	214.2	202.6	193.5
rb2	154.7	173.2	122.2	182.4	172.2	122.3	167.9	166.9	125.7	166.5
wr1	181.7	189.6	150.8	121.1	192.4	177.2	202.3	203.4	210.2	212.6
wr2	137.5	159.4	122.8	114.2	165.9	134.8	133.7	167.2	171.9	119.3
wr3	123	135.7	113	103.7	149.7	115.2	129.7	147.2	143.2	102.8
te1	92.5	86.6	136.1	89.2	79.4	171.2	92.1	85.4	108.5	114.2
dst	114	126	109	187	132	137	113	117	111	124
k	120.6	123.3	130.8	129.6	127.3	119.3	128.3	120.5	139.9	129.6
Bench										
qb2	254	252.2	278.9	270.3	249.1	262.7	232.8	258.8	236.6	281.2
rb3	153.1	138.3	90.4	157.6	141.5	85.3	158.4	144.8	109.3	123.6
rb4	130	79	88.9	151.4	102.2	83.3	147.8	116.9	96.5	94
rb5	117.3	70.2	75.3	140	82.1	74.9	143.5	83.5	92.5	94
wr4	105.7	124.9	110	102.6	121.1	109.3	119.5	133	139.6	95.9
wr5	99.7	112.5	102.4	94.6	114.7	105.8	116	108.8	117.9	95.1
te2	74.1	81.5	125.4	81	74.2	117.3	82.2	74.4	79.4	83.5
score	**2156.88**	**2163.8**	**2100.34**	**2141.2**	**2157.24**	**2090.16**	**2122.94**	**2165.6**	**2090.94**	**2100.82**
	Avg. VORP	2140.34								
	Avg. Weighted VORP	2162.21								
	Avg. Weighted Contro	2095.57								

Figure 3: Weighted Control vs Value Over Replacement vs Weighed Value Over Replacement

This table also shows comparisons between owners, three of each strategy. We looked at the principle of Value Over Replacement. To no surprise, the VORP strategy was better than the weighted control algorithm. Also, as expected, the weighted VORP does the best, as accounting for the objective function player weights adds value.

10 Conclusion

Our Weighted Value Over Replacement algorithm picks the best team. We believe this is a valid algorithm because the results line up closely with the average draft positions of each of the players across all of ESPN leagues. This means that averaging together people's opinions results in a generally smart decision (this is known as Wisdom of the Crowds), a decision supported by our algorithms. Additionally, most fantasy experts advocate for a Value Based Drafting approach, which utilizes VORP (Value Over Replacement Player).

11 Future Work

In the future, we want to see if we can outsmart the computer by drafting ourselves vs the computer. We predict that we would be better than most of the strategies except for possibly Value Over Replacement and Weighted Value Over Replacement. If we are able to beat our best algorithm, we will attempt to identify what human strategies we were able to use that our algorithms did not, and then incorporate them into the current algorithms.

12 Reflection

We learned a lot about game trees and game algorithms over the course of this project. It is hard to realize how large these trees can grow, and we had trouble letting go of the idea of "solving" fantasy. But just as "solving" chess has never and will never be done, we cannot solve fantasy. We wish we had unlimited computing power to be able to 'solve' the best draft possible by playing out all of the possibilities.

We learned about heuristic algorithms, and how they can be helpful solving large game spaces. We also wrote a ton of Python code in order to simulate all the different draft strategies, which was fun.