Fantasy Baseball Trade Suggester

By: Devin Bhushan | Advisor: Prof. Luke Olson

9-May-2014

Contents

1 Abstract

Forbes reported that as of 2009, there were 11 million users for Fantasy Baseball. Since then, that number has only increased as mobile and web services have become more prominent. However, one pattern that most Fantasy Baseball users have noticed and reported is that the frequency of trades in their leagues is stagnant. Some leagues incentivize trades by allowing inclusion of draft picks in future years. However, by and large, trade frequency is low and this results in a less eventful fantasy experience. This dearth of trades can be traced back to the complicated nature of identifying a trade partner.

Before a trade is proposed, one has to explore the league to find a team that has a relative surplus at a position of one's need as well as a relative need at a position of one's surplus. Then there is the technical process of suggesting a trade that is statistically fair for both sides as well. There is a lack of a simple tool to assess a Fantasy Baseball league and identify both a trade partner and also a suggested framework (in terms of players involved) for said trade.

This paper proposes the foundations to such a tool; a tool that takes as input a Fantasy Baseball league and outputs trade suggestions for a given owner.

2 Definitions

Sabermetrics - A term derived from SABR (Society for American Baseball Research), referring to the empirical analysis of baseball and baseball statistics.

NoSQL ("Not Only SQL") - A database that is modeled as a key-value store, where instead of being stored in tables with columns, documents are without relations in the form of pairs. This model for a database allows for easy access if data is indexed by numerous unrelated keys.

Fangraphs.com - A website that provides statistics for every player in MLB history. The website keeps track of every event of every game - including pitch location, type, speed, release point as well as batter details. Fangraphs also contains an editorial section with detailed statistical analysis of baseball.

Moneyball - *Moneyball: The Art of Winning an Unfair Game* is a book by Michael Lewis, it details the Oakland Athletics' General Manager Billy Beane's statistical (sabermetric) approach to building a winning baseball team despite a low budget.

3 Introduction

I have played Fantasy Baseball since I was just 14 years old. Over the years, I have developed some techniques for success in this stimulating virtual sport. Before the start of the 2006 baseball season, I read the book *Moneyball* and was first exposed to the Sabermetric world. The book was just the tip of the iceberg for me, I engulfed myself in the statistical haven that was Fangraphs.com and rabidly studied statistical trends and metrics.

Over the past few seasons, I have found myself having a hard time analyzing trades that would be beneficial for both my team and a potential partner. It is very time consuming to look through a whole league's worth of teams (somewhere between 8 and 12) and finding a valuable deal that would be worthwhile for both teams. Often this results in very few trades taking place in the league. While this is a consequence most leagues have accepted and are complacent with, I think it would add a whole new dynamic of competitiveness to the game if a piece of software existed that could not only algorithmically identify trade partners for your team, but also suggest which positions each team needs and might want to trade away.

Therefore, the idea for **TODO NAME** was born. The goal for **TODO NAME** is to eventually be put into production such that it can hook into an active Fantasy Baseball league, a data source for statistical projections and provide a user with potential trade partners and specific trade suggestions that would be beneficial for both teams. While trades are generally subjective in nature, this paper serves as a means of analyzing trades through the lens of statistical prowess that the Sabermetric community has only recently been polishing.

4 Technologies Used

MongoDB

MongoDB is a document-based NoSQL database. It interfaces very easily with Python and is one of the fastest databases currently in use. Storing and reading data from MongoDB is exceedingly easy, Python's primary data structure is a dictionary and dictionaries directly translate to the JSON format that MongoDB expects as input and provides as output. A key-value store is perfect for this application since it's primary storage is just league and player performance data, which is indexed by *league_id* and *player_id* - resulting in `O(1)` lookups.

Source: http://www.mongodb.org/

MongoHQ

MongoHQ is a fully-managed platform for hosting MongoDB databases. It provides an easy interface to set up an instance of MongoDB for the application to use.

Source: http://www.mongohq.com/

Python

Python is a powerful and fast programming language that allows for easy prototyping and interfaces well with all other technologies being used for this project.

Source: www.python.org

Github

Github is a web-based hosting service for Git-style version control for this project.

Project Source: https://github.com/devinbhushan/FFBaseball_Trade_Suggester_Thesis

Munkres - Python Library

The Munkres python module provides an implementation of the Munkres algorithm. It models an assignment problem as an `O(NxM)` cost matrix, where each element represents the cost of assigning the i^{th} worker to the j^{th} job, and it solves for the least-cost solution. This algorithm proves essential in computing the optimal lineups for all teams.

Source: https://pypi.python.org/pypi/munkres/

5 Workflow

The current application uses a mock source of baseball projections as well as a mock Fantasy Baseball league from 2010. The roster and scoring settings are those imported from a standard 10-team, Head-to-Head points-based league.

Collect Data

In order to allow future development of a plug-in for online fantasy leagues, this application is developed such that a league can be imported into the database. Thus, data from the database is standardized and the source of the data - online or offline - becomes irrelevant. Conveniently, NoSQL does not require players to be indexed by specific columns - collision of name, position, and team is a common problem when storing a large amount of athletes in a relational database. Therefore, a JSON object can easily be created for a player and his statistics to be stored in the database.

The sample statistical projections for all players are also loaded into the database in a similar manner. In the future, a module can be used that reads from a dynamic source of data for projections - allowing for more robust trade analysis.

Import Settings

Every fantasy league has unique scoring settings as well as roster settings. Therefore, there is an additional plug-in system to allow to import these settings from an online league. Currently, a sample league's settings are being utilized.

Compute Projected Points

Each player's projected point total is computed by using a league's scoring settings (that specify point distribution) in tandem with the player performance projections.

Compute Optimal Lineups

Computing optimal lineups is the first step to determining strengths and weaknesses for a team, which itself is a stepping stone to finding an appropriate trade partner. Performing

this computation is actually a difficult process since a team may have multiple players who are eligible for multiple positions.

For example, **Figure 5.1** demonstrates a group of players that play a common set of positions. It is difficult to determine the configuration of positions and players that results in the optimal lineup, as per projections. **Figure 5.2** shows one such optimal lineup. However, analyzing the lineup in **Figure 5.1** and arriving at a solution like that in **Figure 5.2** requires one to find the optimal matching between players and positions such that the projected point total is maximized.

Player	Eligible Positions	Projected Points
Joe Mauer	C, 1B, U	540
Buster Posey	C, 1B, U	573
Brandon Belt	1B, OF, U	497
Jose Abreu	1B, U	467
Miguel Cabrera	1B, 3B, U	615

Figure 5.1: Potentially Conflicting Lineup

Player	Optimal Position	Projected Points
Joe Mauer	1B	540
Buster Posey	C	573
Brandon Belt	OF	497
Jose Abreu	U	467
Miguel Cabrera	3B	615

Figure 5.2: Optimal Lineup

This problem can be represented as a Bipartite Matching where each player on the team can be represented as a node on one side of the graph and each position in the lineup can be represented as a node on the other side. Positional eligibility is reflected on this graph as an edge connecting the player and the position. The edge weights are set to the projected point total for the corresponding player. **Figure 5.3** illustrates this process for our current example.

Now, ideally, running a simple Maximum Bipartite Matching algorithm on this graph can produce **Figure 5.4** such that every position has a player mapped to it (when possible)

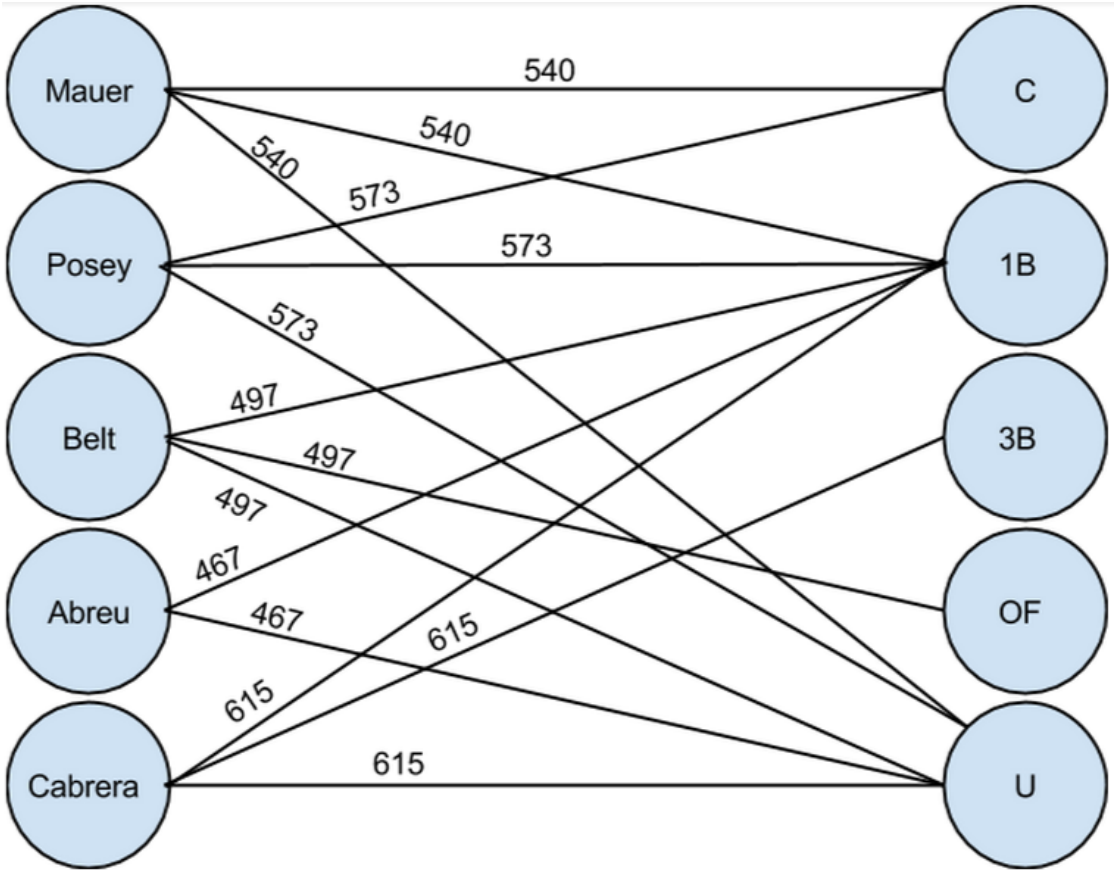

Figure 5.3: Lineup possibilities in bipartite graph

and the total number of points contributed to all positions is the maximum amount possible from all permutations.

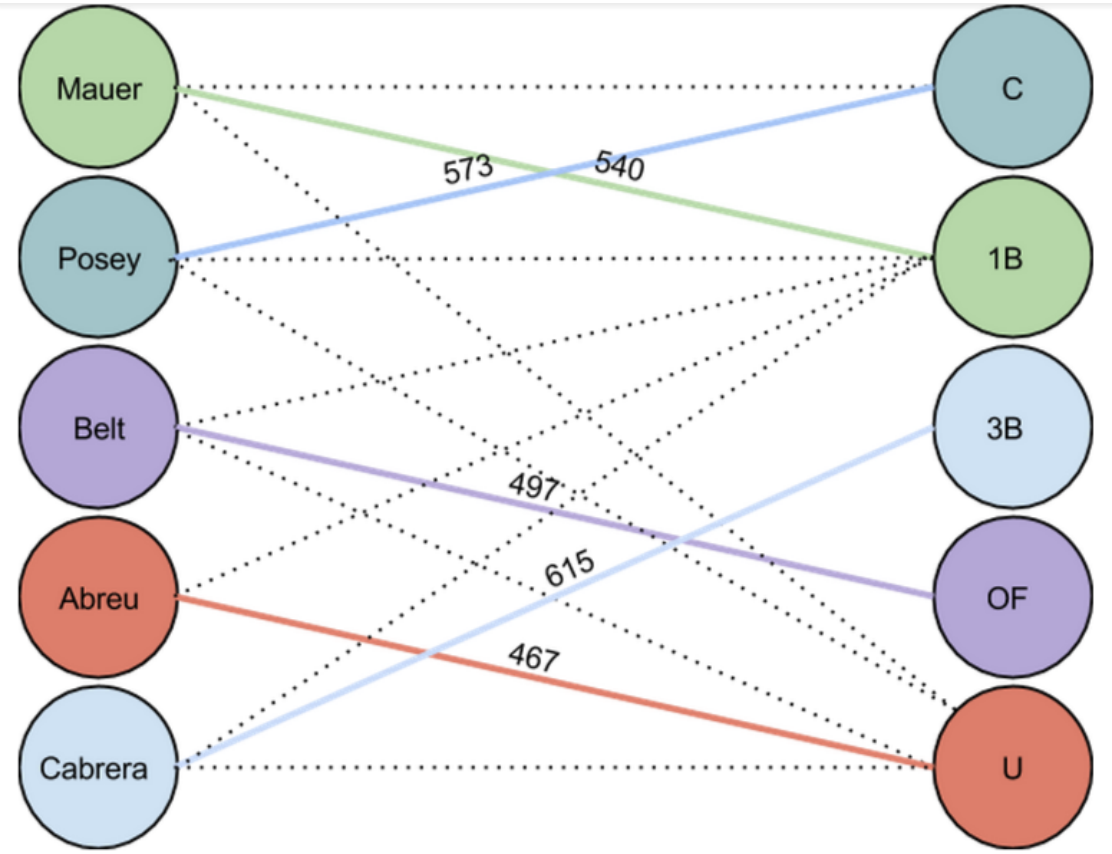

Figure 5.4: Optimal lineup using maximum bipartite matching

However, the easiest implementation of a Max-Flow, Bipartite Matching, or similar algorithm happens to be Munkres Assignment Problem (also known as The Hungarian Algorithm[1]). This approach to the problem results in an `O(NxM)` running time, in terms of `N` players and `M` positions - therefore, it does not scale well for large input sizes [2]. However, within the scope of a fantasy baseball team, the input size is a constant (about 20 players and 10 teams) so the subpar performance is never exposed. This solution will scale for any league since the largest, most eccentric, leagues still have a small, finite upper bound on the number of teams and number of players per team.

This algorithm models an assignment problem as an `NxM` cost matrix, where each element represents the cost of assigning the i^{th} worker to the j^{th} job, and it solves for the least-cost solution. In order to represent a fantasy team along with possible positions as an assignment problem, each column in this `NxM` matrix represents a player on the team

[1] http://en.wikipedia.org/wiki/Hungarian_algorithm
[2] https://pypi.python.org/pypi/munkres/

while each row represents a position the team needs to fill. Each element, `(i,j)`, in this matrix represents the number of projected points player `i` could potentially contribute to position `j`. Ineligibility for a position is represented by a 0.

However, Munkres' algorithm solves for a least-cost solution. Each player's projected points are subtracted from the maximum representable integer. Therefore the least-cost solution actually selects the maximum-point lineup while satisfying all positional eligibility conflicts.

	C	1B	3B	OF	U
Joe Mauer	max-540	max-540	max	max	max-540
Buster Posey	max-573	max-573	max	max	max-573
Brandon Belt	max	max-497	max	max-497	max-497
Jose Abreu	max	max-467	max	max	max-467
Miguel Cabrera	max	max-615	max-615	max	max-615

Figure 5.5: Sample inverted-cost assignment matrix to find optimal lineup

The previous sample lineup of players is illustrated in **Figure 5.5** in a matrix appropriate to be used for Munkres' assignment problem. Solving this case by hand would require all permutations of this matrix such that no row and column are used more than once. Then the matrix corresponding to the permutation which produces the least amount of points (since it is an inverted matrix) would be the solution. In this case, there are multiple solutions that are considered optimal so any of them are acceptable. Running the assignment problem on this problem gives us the indexes of elements which belong in a lineup that belongs to the set of optimal lineups. **Figure 5.6** outlines this process for the sample case.

TODO FIX 5.5, 5.6 WITH HIGH-RES

Points Above Average (PAA)

Explain that once you have optimal lineups, you can figure out what the projected standard performance will be at each position. Take an average of entire league at each position then we can calculate the number of Points Above Average that each player is. This is a useful metric because it allows you to compare in context of league performance at the position. So now we can identify surplus for certain teams, even on the bench. It's possible a team could have 2 players who play only one position, U, but both are well above average for that position. So one of these is deemed surplus now with the help of

Optimal Lineup Indexes	Points
(1,0)→Joe Mauer @ 1B	540
(0,1)→Buster Posey @ C	573
(3,2)→Brandon Belt @ OF	497
(3,4)→Jose Abreu @ U	467
(2, 4)→Miguel Cabrera @ 3B	615
Total Points	**2692**

Figure 5.6: Solution indexes and corresponding projected points

PAA. Talk about how we can identify weaknesses as well based on PAA. Even identify a weak bench by computing how many points the bench is expected to contribute.

Identify Strengths and Weaknesses

Suggest Trade Targets

Talk about how we can match up one team's marked strengths with another's marked weakness. Now the challenge is to figure out how fair a trade is. Often a position like C has fewer points scored... but this doesn't mean that you need to give up a 500 point C to get a 500 1B. The latter is very common. So the PAA metric comes in very handy in analyzing how many points are being exchanged across the trade. So multiple positions and players can be exchanged but really at the end of the day, the overall PAA evens everything out.

Provide simple example of 2 for 2 trade where it's even but 1 strong player at different positions on each side of the deal makes it even.

Provide examples of 3 for 5 trades where the 5 side is actually getting ripped off.

6 Future Considerations