# Moneyballer: An Integer Optimization Framework for Fantasy Cricket League Selection and Substitution

Debarghya Das
*dd367@cornell.edu*

## Abstract

This paper, Moneyballer [1] , uses binary integer programming to create optimal sequences of teams in fantasy sports leagues, particularly cricket. Some of the properties of team selection in sport are very similar to that in finance, which make this problem somewhat similar to the former. We recalibrate our model according to the distinctions between the two. For example, like the Black-Litterman model, our technique shows theoretical guarantees on overall team performance in a league dependent on your predicted return. Like the Markowitz model, it attempts to optimize on returns, however it does not penalize for risk and assumes (perhaps incorrectly), independence between performance of assets (players). Our model breaks software limits in previous research by **33x**, and produces a team which rates **99.53%ile** in a league of above 430,000 participants. It also introduces an automated backtesting framework for cricket performance measure testing, and supplies a method to fetch rich cricket statistics automatically.

## 1 Outlining the Problem

### 1.1 Cricket - Basic Gameplay

Cricket is a bat and ball game played by two teams of 11 players each. The game is played in several different variants. The format of the game we deal with, in particular, is the **T20 format**. Regarded as a fairly complicated game, here are some general gameplay rules of T20 cricket:

- One team goes out and bats, while the other bowls. This session is called an **innings**. When the first team's innings is finished, either because all their 11 players got **out** or because they finished playing their allotted time, the second team goes out to bat with the goal of getting as many runs as the first. Two batsmen bat together at a time. And if there is only one batsman left to bat, the innings is determined to be over.

- The time allotted per team in T20 cricket is **20 overs**. Each over consists of **6 balls** or deliveries. Each over is bowled by one member of the bowling team. Consecutive overs cannot be bowled by the same bowler. Typically, one innings takes one to one and a half hours.

- Each bowler can bowl **4 overs** in one innings. Typically 5 to 6 bowlers bowl from the bowling team in one innings to make up the entire 20 over quota.

- Two batsman from the batting team bat together at a time. They can hit the ball along the ground anywhere and run from one end to the other - a distance of 22 yards. For each time they run, they accumulate that many points.

- If the balls crosses the boundary line, a circle surrounding the batting area usually 60 - 80 meters away, without bouncing, the batsmen gets **6 points**, and if it does bounce, he gets **4 points**.

- There are several ways a batsman can get out.

  - He can be **caught** by any of the 11 people in the bowling side (including the bowler) without the ball touching the ground. When the bowler catches the ball of his own delivery, the dismissal is called **caught and bowled**.
  - He protects three sticks behind him when he bats. If he misses the ball and the ball hits any of the three sticks, he is out **bowled**.

[1] The title is a portmanteau of "moneyball" - a the book and following movie, *Moneyball*, which deals with the successful application of statistics in baseball to win 19 consecutive games with a below average financial budget `http://en.wikipedia.org/wiki/Moneyball`, and "baller" - a street ball player who has made it into the big leagues.

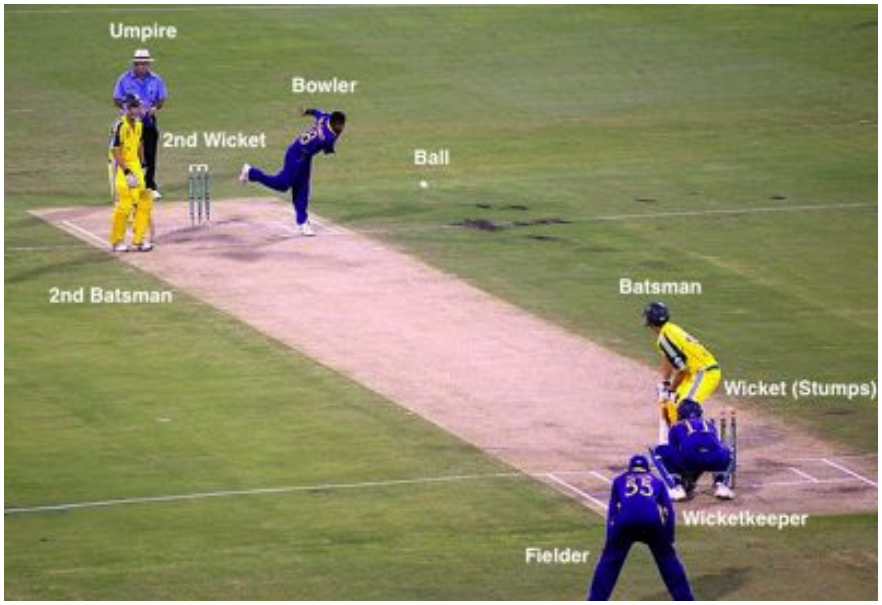


Figure 1: A snapshot of the typical action on a cricket field with some of the important characters labelled.

  - He also keeps some part of his foot behind a line called the crease near the sticks. If, while trying to play his shot, he moves out beyond that line, the person standing behind the sticks, called the wicketkeeper, can hit the sticks with the ball, to **stump** him. This is more uncommon than the previous two.
  - If he misses a ball which would have hit the stumps if he hadn't been there, then he gets dismissed **leg before wicket**.
  - While running from one end to another after hitting a ball, if any person from the opposing side throws the ball to the sticks at each end and hits it (directly or indirectly) before the batsman has crossed his line with some form of his body or his bat, then he is given out **run out**.

- If the first team puts up a score and either dismisses 10 players of the other team for a lower score, or the other team plays out their allotted overs without achieving their target, the first team wins. If the second team manages to score at least one more run than the first team, they are declared winners and the game ends.

## 1.2 Specific Rules and Restrictions

The specific Fantasy we use for testing is the biggest T20 tournament of the word, called the **Indian Premier League (IPL)**. Special rules apply to this tournament, and even more specifically to this Fantasy league, which are relevant to our problem:

- **Squad Balance** - Each player can have one of the following "skillets". There exist 3 "multiple skillet" combinations as well - Batsman + All Rounder, Batsman + Wicketkeeper, and Bowler + All Rounder:
  - **Batsman** (At least 4) A squad must have at least 4 specialist batsman. Although all 11 players can go bat, specialist batsmen typically score many more points, or runs, for his team.
  - **Bowler** (At least 2) A squad must have at least 2 specialist bowlers. Not all 11 players can bowl.
  - **All-Rounder** (At least 1) A squad must have at least 1 specialist all-rounder - some-one skilled at both batting and bowling.
  - **Wicketkeeper** (Exactly 1) A specialist role called the wicketkeeper has to be be in every team - at least one. He is the person who stands behind the sticks and collects the ball if the batsman misses, or attempts to catch balls that deflect off the bat, or stump the batsman when he steps out of his crease.
  - **Bowling Criteria** (At least 5) The sum of number of all rounders and bowlers on a particular team must be 5 or greater. Because each bowler can bowl 4 overs, and there are

20 overs in a match, every team needs at least 5 people who can bowl.

- **Overseas Limit** The Indian Premier League is primarily composed of domestic Indian players, with an abundance of foreign players from other top cricket playing nations, including England, Australia, South Africa and the West Indies.
  The format of the league restricts the number of overseas players in a playing squad to **4 players**.

- **Uncapped Quota** As one objective of the IPL is to foster domestic Indian players who have not played in any international matches before to rise to the occasion. In celebration of this, a squad must have **at least 1 uncapped player**

- **Franchise Spread** There are 8 teams in the IPL (at least this year). Each of them is based off of a particular location of the country. The 8 teams are:

  - **Kolkata Knight Riders (KKR)**
  - **Royal Challengers Bangalore (RCB)**
  - **Kings XI Punjab (KXIP)**
  - **Delhi Daredevils (DD)**
  - **Mumbai Indians (MI)**
  - **Rajasthan Royals (RR)**
  - **Sunrisers Hyderabad (SRH)**
  - **Chennai Super Kings (CSK)**

  A franchise spread criteria for Fantasy IPL is that you cannot have **more than 6 players** from one particular IPL franchise.

- **Financial Strength Criteria** Clearly, all the players available are not of equal strength - their skillets range from international legends, to upcoming uncapped 19 year olds. Based on their skill set, each player is given a price tag - ranging from **$600,000** to **$1,100,000**. The budget that you, as a manager, have while making your team


Figure 2: The logo of the 2014 Pepsi IPL T20

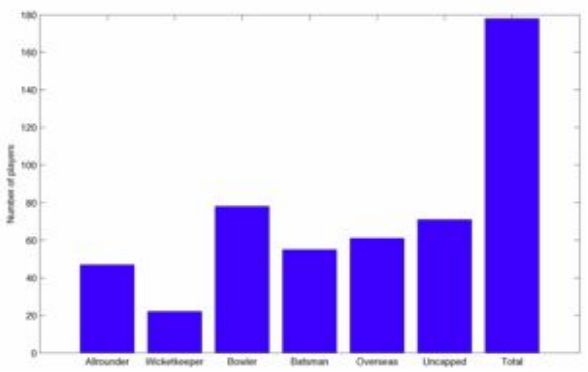


Figure 3: A distribution of the various roles of players in the IPL 2014.

- **Substitution Constraint** In the preliminary series of the tournament, there are **56 total games** played between the 8 teams. In each, a user receives points if a player he or she has is playing on that day gets a certain number of points depending on a scoring metric discussed later. If, for example, RCB plays KKR, and you have no RCB or KKR players in your team that day, you receive 0 points. Each user gets **75 substitutions** for the preliminary round.

- **Uncapped Player Substitution** To promote uncapped player, the Fantasy rules allow **one free uncapped player substitution** in every match, in addition to the 75.

- **Power Player** Every match, you get to choose your ”power player”. This player gets his points in the next match doubled.

- **Lock In** To prevent trading of players during the match, a user is only allowed to modify his team before the start of a match. After the match starts, his team is **locked**, and no further substitutions will count for points in that match.

## 1.3 Scoring System

The IPL 2014's preliminary stages consist of **56 games** from **April 16** to **May 25th**. On some days no games are played, but usually each day has **1-2** games. You are allowed infinite substitutions before the beginning of the tournament. Before the beginning of the next day's match, you're allowed to change your team. If a game X vs. Y occurs tomorrow at time $t$. At $t$, your team selection is locked down. After the game is finished, you receive points as per the rules above and system below. Points are only received by players on your fantasy that belong to teams X and Y if they played that day. All

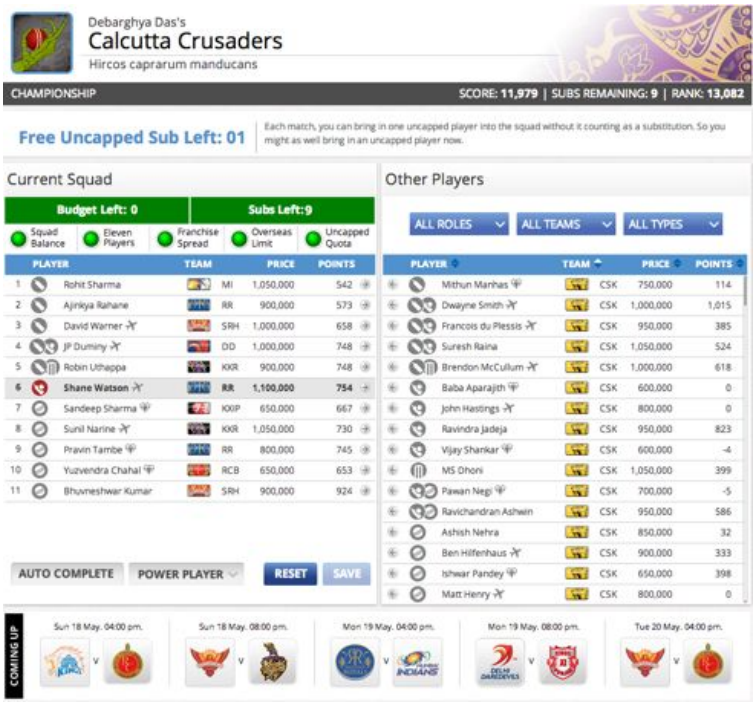


Figure 4: A look at the dashboard for exchange of players in the Fantasy IPL.

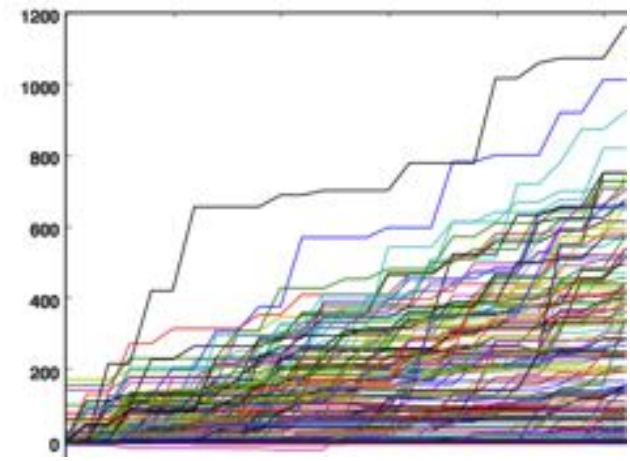


Figure 5: A graph showing the progression of points for each of the 178 players in the tournament.

players from other teams receive 0 points if they do not play on that day. After every day, you're allowed to make changes or substations to your side, but you are capped as per the rules above.

The fantasy league attempts to gauge the performance of distinct aspects of the game on one linear point scale. It uses various statistics based on a player's performance in a match to derive his points, which may be positive or negative. Note, the power player gets double his base points, regardless of whether negative or positive.

- **Batting points**
  - **Base Score** 1 point per run
    The more runs your score, the more points you get.
  - **Impact Score** 2 points per six, -5 per duck (getting out on 0 runs)
    In the smaller, 20 over format, six runs is a great boost in the scoring rate, and rewarded. Getting out without scoring is a disappointment, and penalized.
  - **Milestone Score** 10 points for every 25 runs scored
    Intuitively, 25, 50, 75 and 100 have been regarded as large milestones for runs in crickets in a given innings, and players are rewarded for it.
  - **Pace Bonus** Runs scored - balls faced
    Intuitively, players who hit more runs off fewer deliveries are more valuable, and they are rewarded thus.
- **Bowling points**
  - **Base Score** 20 points per wicket
    A wicket, or getting an opposing batsman out, is a huge prize. Only 10 can be taken in an innings, and each is rewarded with 20 points.

- **Impact Score** 1 points per dot ball, 20 points for a maiden over
  A dot ball, or a ball where no run is scored, is rare and is a desirable by the bowler. A maiden over, or one segment of 6 consecutive dot balls, is rewarded handsomely with 20 points.
- **Milestone Score** 10 points for the 2nd wicket, 10 points for each subsequent wicket
  Taking more than one wicket in a T20 is an uncommon feat, and rewarded with bonuses.
- **Pace Bonus** 1.5x Balls bowled - runs conceded. If positive, it is doubled.
  A bowlers goal is to concede as few runs as possible, also known as maintaining a good economy rate. For doing this, they are rewarded with points, even if they fail to dismiss a batsman.

- **Fielding points**
  - **Catches** 10 points
    For a player who takes a catch to dismiss a batsman.
  - **Stumping** 15 points
    For a wicketkeeper who hits the sticks with the ball when the batsman is not within his crease, or his safe line.
  - **Direct Hit** 15 points
    When a fielder throws the ball from the distance and it directly hits the sticks while the batsmen are running, and they are not in their safe zone, then they are rewarded points.
  - **Run Out** 10 points for each player involved
    Similar to a direct hit, except the throw from the distance need not directly hit the stumps. As long as a batsman is dismissed before he makes his ground, all fielding players involved in throwing the ball are rewarded.

- **Bonus points**
  - **Player of the match** 25 points
    The most valuable player of every match is rewarded 25 more points for his performance.
  - **Victory Bonus** 5 points
    A player in the victorious side is rewarded with 5 points.

## 1.4 The Problem

We broadly separate our problem in to 2 parts - **static prediction**, which creates a selection given rules and constraints on day 1 for the entire fantasy league without updating expectations after the actual cricket games take place, and **dynamic prediction**, which solves the problem after every day of play, updating its expected player performance with time.

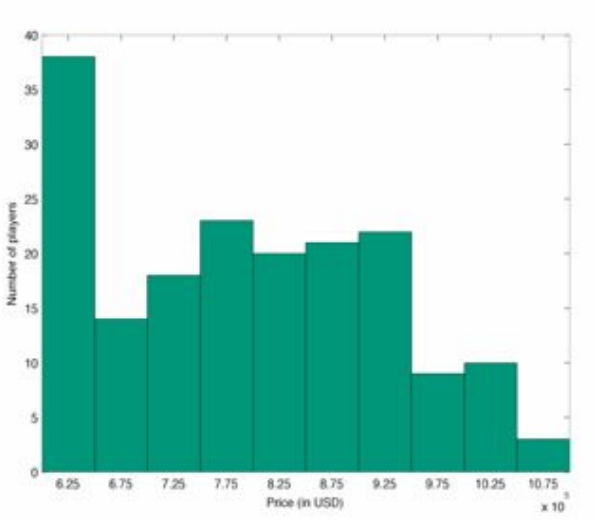


Figure 7: A distribution of the prices of all the players in the IPL. They range from $600k to $1.1m.

### 1.4.1 Static Prediction

In plain words, our problem $S$ is to pick a team of 11 players for every one of the 37 days of play, each of which should abide by certain cricketing squad balance and budgeting based team constraints. The total number of substitutions incurred in the selection of all these teams should be less than equal to 75. The resultant selection of teams should maximize the expected points that you, as a "manager" collect from the entire endeavor. The point formula is strongly correlated with the general quality of cricket performances. The initial expectations of all the players can be given by $e$, in which case this problem problem produces a solution $S(e)$. The static forecast is a solvable problem, with a "correct" solution to be found.

### 1.4.2 Dynamic Prediction

If the Static Prediction problem is called $S$, then the Dynamic Prediction problem solves $S$ after each day of play with an expectation $e$. If $e$ were to remain constant from day to day, then the solution of $S_1(e)$, the static prediction given an expectation $e$ on day 1, would be such that $S_2(e) \in S_1(e)$. If it remained constant for all $n$ days,

$$S_{k+1}(e) \in S_k(e) \;\; \forall 1 \leq k < n \;\; , k \in \mathbb{Z}$$


Figure 6: The 8 teams participating in the IPL 2014.

Therefore, applying each static forecast solution on each day would be equivalent to applying $S_1(e)$.
However, in sport, the odds of all players performing exactly as you expect is negligible. As such, we have a separate $e_k$ for every game. If $S_k(e)$ produces a list of teams for days $k...n$, we refer to the $i^{th}$ team that $S_k(e)$ predicts as $S_k(e)[i]$ The Dynamic Prediction algorithm $D$ is simply

$$D = [S_k(e_k))[k] \mid 1 \leq k \leq n]$$

This is simply an aggregate of static predictions for updated player performance expectations.

## 2 Why this is an interesting problem?

**Intellectual Merit**

- **Uniqueness** The area of using optimization based approaches for team selection in sport, especially cricket, has not been a typically wide area of research. The only previous research on it varies little, and comes from a Econometrics and Sports Modeling research group at *Nelson Mandela Metropolitan University* [2], with Gerber and Sharp [2006] being the seminal work in the field.

- **Directly applicable to Operations Research** While typically most of the work in this field is limited to Sports journals, or Statistical journals, this particular approach to the problem has been published in *Journal of the Operational Research Society* [Sharp et al., 2011].

- **The use of an NP-complete 0-1 Integer Linear Programming Method** In computational complexity theory, Karp [1972] reduced 21 combinatorial problems, including 0-1 linear integer programming, to SAT, proving it NP-complete. This makes the problem a *superpolynomial* in the input size, and a challenging problem to solve.

- **Using *Gurobi* to its limits** In Meindl and Templ [2012], we learn that Gurobi ranks 1st in all standard benchmarks [Dolan and Moré, 2002], Mittelmann LP benchmarks, Mittelmann [2003]and 2nd amongst all other benchmarks set by the paper[3] in performance while solving linear problems. Gurobi [4] is a commercial optimization software used to solve a wide variety of optimization problems. Most noticeably, Gurobi has multi-core parallelism features and a concise API. One article says "The harder the problem, they say, the better their performance and the bigger the gap." [5].
  The cutting edge performance of Gurobi makes it an ideal solution to a problem which was once previously bounded by software limits.

- **Similarity and applicability to the classic financial trading models** The general framework of the problem fits in with the algorithms traditionally used in finance - such as the Markowitz Model [Markowitz, 1952] and the Model [He, 1999]. We discuss in further sections.

- **Making available fairly inaccessible data** Previous approaches [Brettenny, 2010] use manual data collection methods using *Excel*. We automate data collection and updating using a headless browser like PhantomJS and certain jQuery selectors to scrape recent data [6].
  Our superior data collection method not only lends itself to a more comprehensive, automated framework, but provides a cohesive set of data for backtesting as the tournament progresses. This backtesting data could be invaluable for future benchmark-

[2] GD Sharp, WJ Brettenny, JW Gonsalves, M Lourens & RJ Stretch

[3] The software that ranked first was CPLEX, developed by two of the three Gurobi founders.

[4] `http://www.gurobi.com/`

[5] "First Look - Gurobi Optimization" - James Taylor - `http://jtonedm.com/2011/03/02/first-look-gurobi-optimization/`

[6] We currently have the Javascript for jQuery selectors, but have not integrated the headless browser into the update framework

ing of predictive performance metrics, which is the majority of cricket research, as discussed in the next section.

**Broader Impact**

- **Predictive Application to Fantasy Sports Selection** Fantasy sports leagues has become extremely popular. The English Premier League Fantasy Sports League has over 2,000,000 participants [Liantzas et al., 2013] and the IPL Fantasy League, in 2014, has around 432,000 participants. [7]
  This paper provides a cohesive framework that can be used an average Fantasy League player, much like an investor in finance, using either performance measures supplied by the algorithm, or their own instinct, such as in Black-Litterman. Regardless of what they choose, the framework guarantees an optimal set of teams and substitution, leaving the burden of satisfying constraints on the software. The user merely has to write an algorithm or use instinct to input his predictions on player performance, or use one pre-written our system.

- **Generalizable Application to Constrained Combinatorial Selection** Our framework does a lot of things out of the box - automatic up-to-date data retrieval, backtesting, and solves large optimization problems (using Gurobi) to select teams. The only things to modify are - schedule to backtest on, data for players, rules, and the endpoint from which to update data (or a historical store). It can be used for many more applications than merely cricket.

- **Direct Application to Statistical Cricket Selection** Selection into sports teams is a matter of pride and competition. Very often, we hear of controversial sports selections, that cause an uproar. We want to back the intuition behind team formation and selection with a theoretical, statistical framework.
  Although, the particular constraints restrict the software to the IPL Fantasy League, it can trivially be modified to asset in statistical based selection for real world international, or even by teams in the IPL, to assist in which players to buy in auctions for how much and how they will perform. With some modifications to the framework, application to a real-world cricket scenario is not difficult.

## 3 Previous Works

There exists a fairly small amount of total papers based on computational cricket in statistics, many of which are discussed below. These works can be broadly categorized into several groups, and I discuss how their contributions are relevant yet fall short of solving the problem at hand:

- **Data Aggregation and Performance Indicators** An overwhelming amount of work in the sports statistics area, particularly cricket, is done purely on aggregating and visualizing statistics. In Damodaran [2006], the author aggregates and visualizes certain data metrics of around 14 players, without making any key insights. Most papers are centered solely on finding statistics that are most indicative of success of a particular type of player in a particular sport.
  Saikia et al. [2012] uses a fairly complex machine learning method using Artificial Neural Networks for predictive performance of bowlers.
  Sharma et al. [2012] attempts to pick the best batsmen in T20 with an ordered weighted average while Amin and Sharma [2014] tries a more in-depth approach two-way regression approach.
  Lemmer [2011] attempts to find statistically significant performance measures for wicketkeepers in cricket.
  Sharma [2013] does a general study of performance indicators in the sport.
  These works are all very useful for the future, and could all be integrated into my model as a performance predictor.

- **Singular Team Selection with Little Predictive Ability** There exists a large amount of work that assists in selection of a single team in cricket. They aren't meant to be used in the same setting as the algorithm in this paper. They are not-self updating and are not built into backtesting frameworks.
  For example, in Omkar and Verma [2003], a genetic algorithm was used to create a cricket team. The study seemed to optimize one team of 11 players from a set of 23, based on certain "fitness" parameter. It does not indicate sufficiently adequately results regarding team balance. Neither is it sufficiently adaptive from one game to the next. It does not discuss how useful this "fitness" parameter is in predicting future performance. In Ahmed et al. [2011], they use a multi-objective optimization function to optimization on bowling averages and batting averages of players. This approach has many flaws. For example, it assumes that batting and bowling averages are the best metrics to optimize on to construct the perfect team. Secondly, the manner in which the paper tries to construct appropriate teams from the "knee cap" of the Pareto-optimal front seems like a add-on sanity check. Fur-

[7] Accessed 17th May, 2014. `https://fantasy.iplt20.com/ifl/leaderboard/list/allteams`

ther, it only models one team of 11 players , and chooses from a pool of 129 players.
Similarly, works like Bhattacharjee and Saikia [2013] and Lemmer [2013] purely deal with rather arbitrary measures for the far more simple task of simple selection.
Summers et al. [2007] was a pioneering work in statistical cricket selection, used in hockey leagues.

- **0-1 Integer Optimization for Modelling Team Selection and Substitution** Of all the areas of computational cricket research, this area strikes me as most relevant and closest to the work that I present [8].
  As stated before, there seems to be only one Econometrics and Sports Modeling research group that has done all the work in this area - Brettenny [2010], Lourens [2009], Brettenny et al. [2012], Sharp et al. [2011] - all talk about the same concept, similar to my approach. The seminal work in this subfield is Gerber and Sharp [2006], also from this research group. However, all these papers have key setbacks -
  - They have a manual and limited data collection process. They capture data from only one previous tournament, and only compute aggregate, and not match by math statistics.
  - Their software limits them to operating on 50 players for every matches for 4 matches total.
  - They do not develop a framework for backtesting the model.
  - They cannot be practically used in a Fantasy League environment.
  - They conform to the older Fantasy League standards, which are riddled with several overtly lax, or unreasonable constraints on cricket gameplay. The Indian Premier League started in 2008, and most of their research conforms to standards between 2008-2010. This paper uses the updated rules and scoring methods of the 7th season, [9]. Several of these newer standards are trickier to integrate into the integer optimization framework.

# 4 Relationship to Finance

In many ways, fantasy league player selection and expected point optimization is similar to well studied finance theories.

[8] At the time of writing this paper, I had no clue of the existence of these works. It turns out that I solve a very similar problem in a similar way, but in ways discussed throughout the paper, better.

[9] Official IPL T20 Fantasy League Standards, 2014 - `https://fantasy.iplt20.com/ifl/default/faq#`

## 4.1 Similarities

- **We have a portfolio we want to optimize** In the finance analogy, assets are players, and portfolios are teams we select at different time stamps. We want to optimize the performance of this changing portfolio over time (this becomes our objective function).
- **We can model an expected return based on past performance** Similar to the Markowitz model [Markowitz, 1952], we can model performance of our assets based on past performance. Also, similarly, these models are usually not consistently accurate.
- **We have a budget restriction on the assets we purchase** In finance, we have a specific amount of wealth we can choose to invest in our portfolio. Similarly, in the fantasy league, we have a specific budget restriction for selecting our portfolio.
- **Integrating (better) Investor beliefs is possible, and sought** In the Black-Litterman model [He, 1999], investor beliefs are used to guide the expected value of the assets. Similarly, in fantasy cricket, because the past predictors are fairly week, expert investor beliefs can trivially be integrated to model expected return.

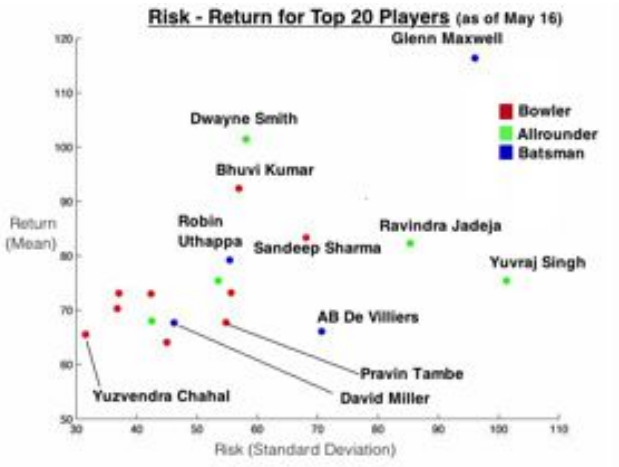


Figure 8: The demonstration of risk-return tradeoffs between 20 top IPL 2014 performers, according to their role in the team, or analogously, their "asset class".

- **There exist a risk-return tradeoff between different asset classes** In finance, for example, you have 2 distinct types of asset classes - bonds and stocks. Bonds have low risk and low return, while stocks have high risk and higher return. In cricket, player roles work in similar ways. Because bowlers always

bowl regardless of game events, and great bowlers always return a sizable number of points, they are like bonds. The way we model bowler returns is very different from batsmen. Even great batsmen can get out on a duck (without scoring runs), or alternatively score many runs very quickly, and acquire much more points than bowlers will ever be able to. They are like stocks.

- **We can potentially model riskiness in assets** Similar to finance, it is not trivial to accurately describe the riskiness of an asset. Yet with similar standard deviation metrics, it is possible to describe an asset's riskiness, and this can further be optimized in the objective function.

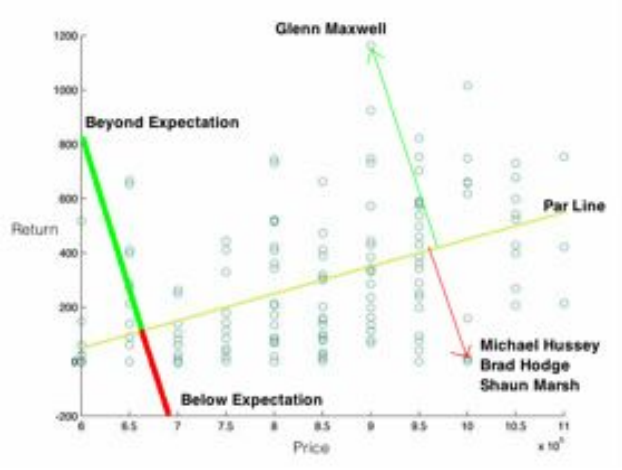


Figure 9: The demonstration of how player performances have stacked up to their price, showing the most over-priced and under-priced players of the tournament.

## 4.2 Differences

- **We cannot short assets** In finance, we can bet against an asset by short selling it. Unfortunately, in fantasy sports league, you can only buy assets (players), not bet against them.

- **Changes to the portfolio aren't charged, they are restricted** Unlike in finance, where typically changes to the portfolio are priced with a transaction cost, either fixed or variable or both. In fantasy cricket, the total number of changes you can make to your portfolio are restricted, not charged.

- **More than one "amount" of an asset can't be held** In finance, we can hold any amount of a certain asset. In fantasy sport, we cannot own a player more than once [10].

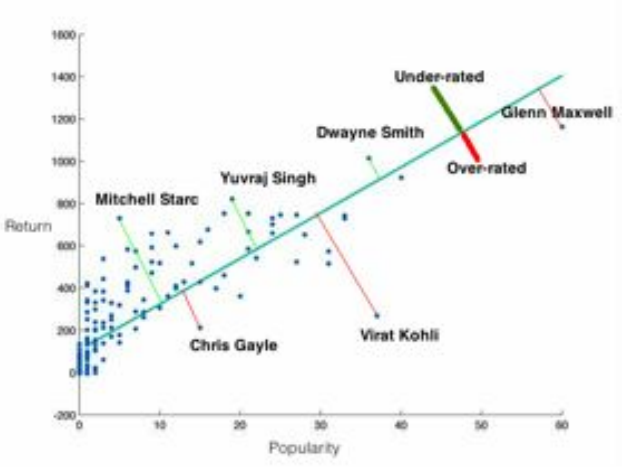


Figure 10: The demonstration of how player performances have stacked up to their popularity amongst IPL fans and fantasy players, showing the most over-rated and under-rated players of the tournament.

- **Assets are assumed to be negligibly correlated** Whereas in finance, you have sets of assets who's prices are strongly correlated, and the market tends to trend in one direction, in fantasy cricket, one assumption (debatable) is that the assets move independently of one another.

- **All Assets do not change price everyday, but we know which ones do when** Whereas in finance, every asset moves in value every day, in fantasy cricket, the value of a player changes only if he plays a match. Otherwise, their scores are guaranteed to remain constant.

- **Tomorrow's asset price expectation is not centered on todays** In finance, we express the gain or loss as a function of yesterday's asset price. Everything moves in percentages about the center.
  In cricket, the batsman or bowler starts from scratch in every match. While there is a slight correlation between a good performance in the match before and this match, a consistency element called "form", a player scores do not tend to their score in the last match.

- **The returns on our assets are a distinct value from the spending on them** In finance, when you invest in assets and make a profit, the profit can be

[10] A point to note is that the IPL T20 Fantasy Cricket League does have a facility for setting a **power player** on every game. This power player gets rewarded with double his or her original score. In some sense, 2 of the same asset is held.

reinvested into your portfolio. In fantasy cricket, your reward and spending are separate entities. No matter how much your reward, you will always be restricted to the same budget restrictions as before throughout the tournament.

- **Prices of assets do not change** Continuing from last point, the price of the asset also doesn't change in fantasy cricket.
- **Negative returns are rare** Although small negative returns are theoretically possible, they are few and far between.

These key difference force us to adapt our finance model to fit the criteria.

# 5 Math & Methodology

## 5.1 Theory

We attempt to formulate our problem as a mathematical optimization problem. Most mathematical optimization problems have 3 key components:

- **Decision Variables** These variables make the ultimate decision of the optimization problem.
- **Constraints** These are the restrictions imposed upon the decision variables.
- **Objective Function** It defines what must be optimized, in terms of the decision variables.

These 3 components are essential to formulate an optimization problem. Particularly relevant to our approach is Linear Programming(LP) , Integer Programming (IP) and Binary Integer Programming(BIP).

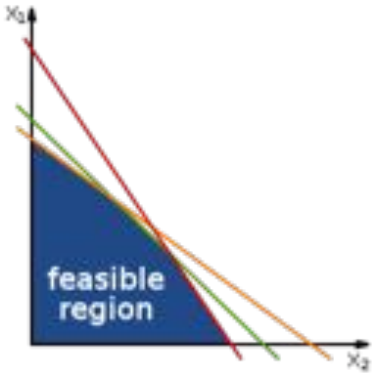


Figure 11: The feasible region, or convex polygon satisfying all constraints in a Linear Program(LP)

**Linear Programming (LP)**

A LP method is an optimization method that operates on linear constraints and objectives.
Geometrically, this can be interpreted in 2D. The set of constraints on a linear program bounds its decision variables to a convex polygon in the 2D plane. It then tries to find the point inside this convex polygon that maximizes or minimizes the objective function. Typically, for linear programs, this will lie on the edge of the polygon.
Linear programs are expressed as follows:

$$\text{maximize} \quad c^T x$$

$$\text{subject to} \quad Ax \leq b$$

$$\text{and} \quad x \geq 0$$

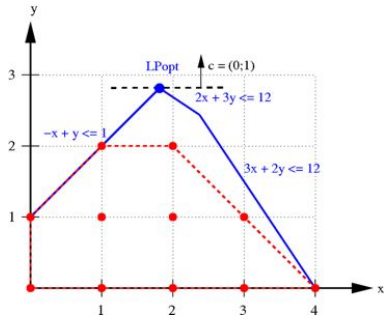


Figure 12: A demonstration of LP-relaxation on an Integer Program

**Integer Programming (IP)**

Integer Programming (IP) is a special case of Linear Programming(LP) where the decision variables are integral, i.e, the problem comes with an additional constraint:

$$x \in \mathbb{Z}$$

They are typically a harder class of problems than LP because they are NP-hard and cannot be efficiently solved in the worst case. Algorithms used to solve IP include *Cutting-Plane Method*, *Branch and Bound*, *Branch and Cut*, *Branch and Price* and *Delayed Column Generation*. NP-complete problems such as *SAT*, *Travelling Salesman*, *Vertex Cover* and *Set Packing* can all be solved using IP methods.

**Binary Integer Programming(BIP)**

Binary Integer Programming (BIP) is a special case of Integer Programming (IP) where the decision variables

can only be 0 or 1, i.e., the problem has the additional constraint

$$x \in \{0,1\}$$

Gurobi typically solves these problems using **Branch and Bound**. The algorithm works as follows:

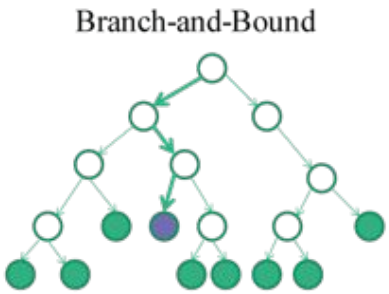


Figure 13: Each node of the branch-and-bound algorithm is a new IP problem.

- Remove all integrality restrictions. This step is known as *LP-relaxation*.
- Proceed to solve the *LP-relaxation*. If our solution adheres to the binary integrality constraints, proceed.
- If not, we pick an arbitrary non-binary variable and impose the restriction $x \leq 0$ and $x \geq 1$ and branch out the problem into 2. Eventually we will take the optimal value from the two to attain optimality.
- We recursively compute LP-relaxations and branch on our children, picking the optimal amongst our children until we get the optimal value at the root. However, simple Branch and Bound on a BIP program will usually be exponential. To solve this, we use the concept of **Fathomed and Incumbent nodes**. Essentially,
  - If at any node, we find a feasible integral solution after LP-relaxation, we mark the node as *fathomed* and do not branch on it.
  - The best solution found so far is called the *incumbent*. If we have no incumbent, the *fathomed* node becomes *incumbent*. Otherwise, no update is made and the search continues.
  - *Fathomed* nodes that yield a higher value than the incumbent is dismissed, and branching stops. If the LP-relaxation itself is unsolvable, the same thing happens

  As we continue our branch and bound using our *fathomed* and *incumbent* node signals, our *incumbent* always represents our best bound. As the branch and bound search continues, the minimum of all the nodes is also stored and reupdated. When the difference between the incumbent and the other bound become 0, optimality is attained.

### Optimizing the Optimization

In recent times, high-performance software like Gurobi uses several techniques to further optimize the speed of BIP. Some of the biggest optimizations are:

- **Presolve** This technique attempts to eliminate redundant constraints using simple numerical analysis. For example if $x_1 + x_2 \geq 3$ and $x_1 \leq 2$ and $x_2 \leq 1$, we know that $x_1 = 2$ and $x_2 = 1$ and we can eliminate all these constraints. This really optimizes the solver's speed.
- **Cutting Planes** This technique, as opposed to pre solve, adds in necessary right constraints on the BIP. This judicious addition of constraints around the optimal value is extremely beneficial to the process.

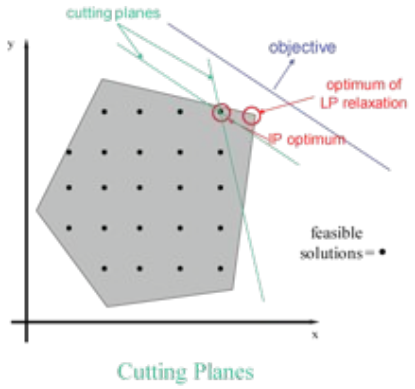


Figure 14: Demonstration of the cutting planes algorithm used to solve Integer Programmign problems

- **Heuristics** Heuristics round results of the LP-relaxation to find integral solutions that may be valid *incumbents*. Finding these helps reduce the amount of branching that might have happened in the future and makes BIP faster.
- **Parallelism** When branching out, the optimization process is easily parallel-izable. The algorithm works on each of the nodes in parallel, speeding up the process, and utilizing more cores.

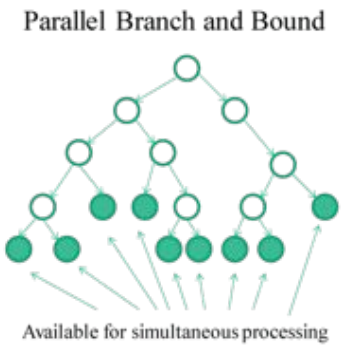


Figure 15: How branch-and-bound can be parallelized across cores in Gurobi's optimization strategy.

## 5.2 Formulating the Binary Integer Programming Model

Now that we know how our Optimization works, we need to formulate the problem's decision variables, constraints and objective function.
Our BIP decision variables somewhat represent a portfolio distribution - 1 represents possession of the asset and 0 represents its absence. In other words, each decision variable represents a player, and is 1 if he is selected and 0 otherwise.

### 5.2.1 Decision Variables

Suppose there are $n$ players and there are $q$ days on which games are played. We want to select a team for every day of play, and therefore need $nq$ decision variables. Our decision vector $\boldsymbol{X}(nq \times 1)$ is defined as

$$\boldsymbol{X} = [x_{11}, x_{12}, x_{13}, \ldots, x_{1n}, x_{21}, x_{22}, \ldots, x_{nq}]$$

where $x_{ij}$ is a binary variable which indicates whether player $i$ will be chosen on match $j$. Formally,

$$x_{ij} = \begin{cases} 1 & \text{if player } i \text{ is chosen for match } j \\ 0 & \text{otherwise} \end{cases}$$

### 5.2.2 Constraints

For our convenience, we define vectors $\mathbf{0}(1 \times n)$,

$$\mathbf{0} = [0, 0, \ldots, 0]$$

and $\mathbf{1}(1 \times n)$,

$$\mathbf{1} = [1, 1, \ldots, 1]$$

- **Team Size** For each of the $q$ teams we pick, we must ensure the number of players in the team for a given day of play is 11. In other words, $\sum_j x_{ij} = 11$. We define the team size vector $\boldsymbol{T_i}(1 \times nq)$:

$$\boldsymbol{T_i} = \left[ \underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\mathbf{1}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}} \right]$$

The constraint is

$$\boldsymbol{T_i X} = 11 \;\; \forall 1 \leq i \leq q, \;\; i \in \mathbb{Z}$$

- **Budget** For each of the $q$ teams, each team must not be worth more than the cap of $\$1m$. We define a price vector $\boldsymbol{P}(1 \times n)$ defined as:

$$\boldsymbol{P} = [p_1, p_2, p_3, \ldots p_n]$$

The price of a player at $i$ is given by $p_i$. We now introduce $\boldsymbol{F_i}(1 \times nq)$:

$$\boldsymbol{F_i} = \left[ \underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{P}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}} \right]$$

The constraint is

$$\boldsymbol{F_i X} \leq 1{,}000{,}000 \;\; \forall 1 \leq i \leq q, \;\; i \in \mathbb{Z}$$

[11]

- **Squad Balance** In order to maintain squad balance in terms of distribution of bowlers and batsmen within a team, we add the following variables:

$$b_i = \begin{cases} 1 & \text{if player } i \text{ is a batsman} \\ 0 & \text{otherwise} \end{cases}$$

$$c_i = \begin{cases} 1 & \text{if player } i \text{ is a bowler} \\ 0 & \text{otherwise} \end{cases}$$

$$a_i = \begin{cases} 1 & \text{if player } i \text{ is a all rounder} \\ 0 & \text{otherwise} \end{cases}$$

$$w_i = \begin{cases} 1 & \text{if player } i \text{ is a wicketkeeper} \\ 0 & \text{otherwise} \end{cases}$$

These variables are defined for $1 \leq i \leq n$. We then define the following vectors of size $(1 \times n)$:

$$\boldsymbol{B} = [b_1, b_2, \ldots b_n]$$

$$\boldsymbol{C} = [c_1, c_2, \ldots c_n]$$

[11] The strategy used to generate constraints of this sort are reused throughout this problem. Essentially, you have $q$ consecutive days, and we generate $q$ vectors of the form $[\mathbf{0} \ldots \boldsymbol{Y}\mathbf{0} \ldots \mathbf{0}]$ in order to apply the constraint $\boldsymbol{Y}(1 \times n)$ to every one of the $q$ teams.

$$\boldsymbol{A} = [a_1, a_2, \ldots a_n]$$

$$\boldsymbol{W} = [w_1, w_2, \ldots w_n]$$

We therefore define $\boldsymbol{BB_i}(1 \times nq)$, $\boldsymbol{CC_i}(1 \times nq)$, $\boldsymbol{AA_i}(1 \times nq)$ and $\boldsymbol{WW_i}(1 \times nq)$, standard constraints for the balance of squad in terms of wicket keeper, all rounder, batsman and fielder representation.

$$\boldsymbol{BB_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{B}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

The constraint is that we need at least 4 batsmen (we use negative signs to preserve the ordering of the inequality):

$$-\boldsymbol{BB_i X} \leq -4 \ \forall 1 \leq i \leq q, \ i \in \mathbb{Z}$$

$$\boldsymbol{CC_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{C}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

The constraint is that we need at least 2 bowlers:

$$-\boldsymbol{CC_i X} \leq -2 \ \forall 1 \leq i \leq q, \ i \in \mathbb{Z}$$

$$\boldsymbol{AA_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{A}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

The constraint is that we need at least 1 all rounder:

$$-\boldsymbol{AA_i X} \leq -1 \ \forall 1 \leq i \leq q, \ i \in \mathbb{Z}$$

$$\boldsymbol{WW_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{W}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

The constraint is that we need 1 wicketkeeper in our team at all time (note the inequality):

$$-\boldsymbol{WW_i X} = -1 \ \forall 1 \leq i \leq q, \ i \in \mathbb{Z}$$

Finally, we have a **bowling requirement** that states that 5 players on the team must be capable of bowling. We construct a vector $\boldsymbol{D}(1 \times n)$as follows:

$$\boldsymbol{D} = [\max(a_1, c_1), \max(a_2, c_2), \ldots \max(a_n, c_n)]$$

The $k^{th}$ element of $\boldsymbol{D}$ represents whether the $k^{th}$ player can bowl or not. We thus introduce a vector

$$\boldsymbol{DD_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{D}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

Then, applying the constraints, we get

$$-\boldsymbol{DD_i X} \leq -5 \ \forall 1 \leq i \leq q, \ i \in \mathbb{Z}$$

- **Franchise Spread** This constraint ensures that no team we select can have more than 6 members from the same IPL franchise.
The $n$ players each belong to a franchise. Let there be $k$ franchises from $1, 2, \ldots, k$. We introduce a variable:

$$f_{ij} = \begin{cases} 1 & \text{if player } i \text{ is a member of franchise } j \\ 0 & \text{otherwise} \end{cases}$$

There are $kn$ such variables. For each team, we create $k$ vectors of the form $\boldsymbol{F_j}(1 \times n)$

$$\boldsymbol{F_j} = \begin{bmatrix} f_{1j}, f_{2j}, \ldots, f_{nj} \end{bmatrix}$$

For each of those $k$ vectors we have a $\boldsymbol{FF}_i(k \times nq)$

$$\boldsymbol{FF_i} = \begin{bmatrix} \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{F_1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{F_2} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \boldsymbol{F_3} & \mathbf{0} & \ldots & \mathbf{0} \\ & & & \vdots & & & \\ \underbrace{\mathbf{0}}_{1^{st}\text{day}} & \ldots & \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} & \underbrace{\boldsymbol{F_k}}_{i^{th}\text{day}} & \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} & \ldots & \underbrace{\mathbf{0}}_{q^{th}\text{day}} \end{bmatrix}_{k \times nq}$$

If a vector $\mathbf{6}(1 \times k)$ represents a vector of $k$ 6s, then

$$\boldsymbol{FF_i X} \leq \mathbf{6}' \ \forall 1 \leq i \leq q \ i \in \mathbb{Z}$$

- **Uncapped Constraint** This constraint requires there to be at least one uncapped player in every team we create.
We define a uncapped vector $\boldsymbol{U}(1 \times n)$ defined as:

$$\boldsymbol{U} = [u_1, u_2, u_3, \ldots u_n]$$

The binary variable $u_i$ decides whether the $i^{th}$ player is uncapped or not. This can be expressed formally as:

$$u_i = \begin{cases} 1 & \text{if player } i \text{ is uncapped} \\ 0 & \text{otherwise} \end{cases}$$

We now introduce $\boldsymbol{U_i}(1 \times nq)$:

$$\boldsymbol{U_i} = \left[\underbrace{\mathbf{0}}_{1^{st}\text{day}} \ldots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{U}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \ldots \underbrace{\mathbf{0}}_{q^{th}\text{day}}\right]$$

The resulting constraint is

$$-\boldsymbol{U_i X} \leq -1 \ \ \forall 1 \leq i \leq q, \ \ i \in \mathbb{Z}$$

- **Overseas Constraint** This constraint requires there to be less than equal to 4 overseas players in the team you create. We define an "overseas" vector $\boldsymbol{O}(1 \times n)$ defined as:

$$\boldsymbol{O} = [o_1, o_2, o_3, \ldots o_n]$$

The binary variable $o_i$ decides whether the $i^{th}$ player is uncapped or not. This can be expressed formally as:

$$o_i = \begin{cases} 1 & \text{if player } i \text{ is an overseas player} \\ 0 & \text{otherwise} \end{cases}$$

We now introduce $\boldsymbol{O_i}(1 \times nq)$:

$$\boldsymbol{O_i} = \left[ \underbrace{\mathbf{0}}_{1^{st}\text{day}} \cdots \underbrace{\mathbf{0}}_{(i-1)^{th}\text{day}} \underbrace{\boldsymbol{O}}_{i^{th}\text{day}} \underbrace{\mathbf{0}}_{(i+1)^{th}\text{day}} \cdots \underbrace{\mathbf{0}}_{q^{th}\text{day}} \right]$$

The resulting constraint is

$$\boldsymbol{O_i X} \leq 4 \ \ \forall 1 \leq i \leq q, \ \ i \in \mathbb{Z}$$

- **Substitution Constraint** This constraint ensures that for the series of $q$ teams generated, the number of total changes between the teams is less than 75. Without this constraint, the currently formulated BIP would be a series of team selection optimizations accumulated into one problem with no change barriers.
Given two binary vectors $b_1$ and $b_2$ of length $l$, the number of substitutions incurred by changing one to the other is

$$\frac{\sum |b_1 - b_2|}{2}$$

For example, consider a universal set of size 4 where we must select 3 players. If $b_1 = [1,1,1,0]$ and $b_2 = [0,1,1,1]$, clearly 1 substitution has been made. $|b_1 - b_2| = [1,0,0,1]$, and summing and dividing indeed gives us 1.
We can use this logic is a constraint for substitution in our $q$ matches. Let $\boldsymbol{cur}$ denote the team you initially enter into the entire optimization with, i.e., the team at match 0. Recall that $\boldsymbol{X}$ is a concatenated binary array denoting $q$ teams selected from $n$ players. The term

$$c = |\boldsymbol{X} - \text{concat}(\boldsymbol{cur}, \boldsymbol{X}[\mathbf{0} : \boldsymbol{n(q-1)}])|$$

is analogous to the previous example, and denotes all changes in our $q$ team transitions. Similarly

$$\frac{\sum c}{2}$$

thus gives the total number of substitutions that takes place. The final constraint is

$$\frac{\sum |\boldsymbol{X} - \text{concat}(\boldsymbol{cur}, \boldsymbol{X}[0 : n(q-1)]}{)} |2 \leq 75$$

#### 5.2.3 Objective Function

The goal of the objective function is to somehow model expected point return on our players.
We define a vector $\boldsymbol{P}(1 \times nq)$

$$\boldsymbol{P} = [p_{11}, p_{12}, p_{13}, \ldots, p_{1n}, p_{21}, p_{22}, \ldots, p_{nq}]$$

Where

$$p_{ij} = \begin{cases} \gamma_{ij} & \text{the expected points for player } i \text{ on day } j \\ 0 & \text{if player } i \text{ is not playing a game on day } j \end{cases}$$

The ultimate objective function, which we try to minify is

$$-\boldsymbol{PX}$$

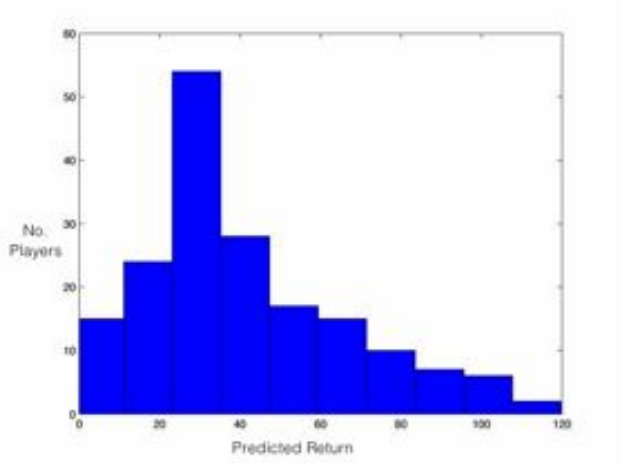


Figure 16: A distribution of predicted performances of the various players in the IPL based on our algorithm, interpolating career history averages as well as current season performances.

## 5.3 Performance Predictors

Our approach to the problem is theoretically sound from an optimization perspective. They key differentiator to how successful it can be is how well it can predict performance. This is similar to the issues we face in finance.

One critical issue to note is that it is extremely difficult to predict the performance of a player in an upcoming game. Much like finance, the movements are very random. At best, we can say a good player will perform well

in the long run. Durbach and Thiart [2007] argue that there isn't any empirical evidence supporting the perception of batsman having periods of "form". They claim it impossible to predict future performance based on past performance.

Like finance, cricket is a statistician's delight. There is a plethora of statistics with which we can attempt to gauge the future. Here we list some performance predictors $\gamma$ we tried and tested and some of some well known ones in literature:

- **Career Statistics** A simple career statistics average projected into the next game. This is the basic model.
- **Exponentially weighted average** Similar to Markowitz, we simply weight statistics for a player for the last several games. Using this weighted average, we compute a fantasy league score expectation for the current game.
  This, however, has much the same disadvantages as it does in Markowitz - the assumption of trends in unclear.
- **Lemmer's Combined Bowling Rate** [12] In Lemmer [2002], given a bowler with predicted runs given $r$, wickets taken $w$, balls bowled $b$ and maidens $m$, the paper introduces the measure "fantasy combined bowling rate" (FCBR) where

$$FCBR = \frac{4r}{w + \frac{b}{6} + \frac{rw}{b} + \frac{rb}{6m}}$$

- **Barr and Kantor Measure** [12] In Barr and Kantor [2004], they introduce a formula to predict a batsman's worth. Given a predicted strike rate of $s$, with $r$ runs scored in $m$ matches, they use a variable $\alpha$, an importance weight on strike rate (how fast runs are scored) to average (how many runs are scored). Their formulation is

$$BK_{bwl} = s^{\alpha}(\frac{r}{m})^{1-\alpha}$$

  They also use a similar formulation for bowlers with an economy rate $e$ and average $a$ and a preference $\beta$ to gauge whether conserving runs or taking wickets is important. The resultant formulation is

$$BK_{bat} = e^{\beta}a^{1-\beta}$$

- **Saikia, Lemmer and Bhattacharjee's Artificial Neural Network Prediction for Bowlers**[12] In Saikia et al. [2012], they introduce a neural network to weigh different statistics by how predictive they are. Such a method would play nicely with our algorithm if it could give give a bound to how well it fit historical data.
- **Amin and Sharma's two-stage regression-ordered weighted average method for Batting** [12] In Amin and Sharma [2014], he builds off a common exponential average method by applying regression in two stages and yields successful results.
- **Support Vector Machine Prediction** Although, it hasn't been seen in previous literature, we can use SVMs as a performance predictor. Much finance literature has been which adopts the use of SVMs on the fundamentals of an asset to create a model to predicts its trajectory. Similarly, we can run such a model on a plethora of cricket data, agnostic of the type of player, to see which statistics play a bigger role in performance prediction.
- **Dynamic Exponential Average** This somewhat novel method used here is essentially the use of exponential averages, updated after every match. In previous literature, usually the prediction is static. This dynamic exponential average method has seen great success and has done a great job in picking up, and capitalizing on trends in form.

[12] These methods are theoretically applicable in $\gamma$ approximation, but were not implemented for this project.

### A False Simplifying Assumption

The idea of performance measure in any sport is hard. One simplifying assumption we make is that the performance of a cricketer is independent of another's. This assumption is, in fact, not true. If player A bats in a match, and plays brilliantly without being dismissed, player B who has been performing consistently may not even get a chance to bat. There are, in fact, complicated relationships between batting strength and bowling strength within ones own team and their interactions between their rival teams. Statistics have shown batsmen to be weak to certain bowlers, certain types of bowlers (pace or spin), certain grounds, and score freely in certain areas. Many of these complex interactions are difficult or impossible to model without *richer* data.

## 6 High Level System Design

One of the primary objectives of this paper is to present a cohesive framework for actual use in the Fantasy IPL, as well as for backtesting to optimize your prediction algorithm. We were fairly successful at accomplishing these objectives.

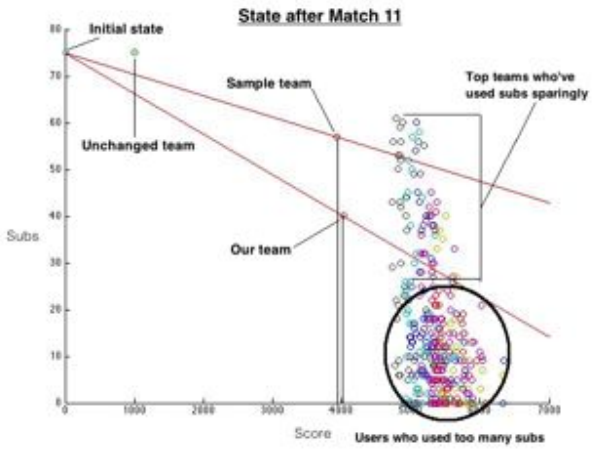


Figure 17: A graph showing the comparison between points scored by our team, and another team, and the trends of many of the teams in the leaderboard.

## 6.1 Technologies

The following technologies were used:

- **Python** We chose Python as it was fairly simple to model structured objects and save and retrieve them from disk via pickling. Initially, we wanted to optimize on Python itself, but Matlab seemed a much better option. Python also has nice online scraping tools.

- **Matlab** Matlab was an inevitable choice because of the familiarity with its use in optimization. It played better with Gurobi than Python did, and was syntactically superior to Python with regard to optimization, using CVXR.

- **CVXR** CVXR is a modeling framework for optimization using Matlab which makes it easy to express constraints and objectives in Matlab syntax and supports the cutting-edge in optimization solvers.

- **Gurobi** As discussed before, Gurobi is probably the fastest solver out there today, and was a great pick for our optimization needs. It played well with CVXR and Matlab, and got the job done.

- **PyMatBridge** Python and Matlab interconnectivity support is highly lacking. PyMatBridge gets the job done, but is fairly buggy and lacks developer support. Matlab is run and interfaced through a local server. Eventually, it got the job done. One difficulty was halting and proceeding with extremely long optimization instances. Gurobi only allows a maximum precision bound of 0.01 and sometimes that takes unto 4-5 minutes or more to reach. In Matlab, halting the process proceeds with the best values found till that point, but using pyMatBridge, it's hard to replicate this behavior.

- **SciPy & NumPy** SciPy and NumPy are essential for Matlab like array and vector transformations. It also helps python read and save .mat files for data communication between the two systems.

- **Pickle** Pickling is a fast useful way Python can store data to disk. We use it to store and update data stores when running our simulation.

- **Javascript & jQuery** Typically, we retrieve current data regarding the data, the number of substitutions we have left, the updated player statistics, and our current team from the IPL Fantasy League interface.

- **PhantomJS & Ghost** We've worked on methods to completely automate this entire process by using PhantomJS with Python via the Ghost driver. PhantomJS is a headless browser, which renders website content, including dynamic content, and can fetch and parse it. Simple GET and POST requests don't let us retrieve dynamic content that we need to update our test after every iteration.

- **urllib2 & BeautifulSoup** In the initial archived rich statistical data collection about players, we comprehensively used urllib2 to query and retrieve and BeautifulSoup to parse web pages.

## 6.2 Functionality

Our system was designed to work in the following stages:

- **Parse Past Tournament Stats Aggregate**
  *parsePlayers.py*
  This was another offline function we wrote to retrieve aggregate stats of previous tournaments which was freely available off an endpoint of the IPL 2014 [13]. This data wasn't rich, nor clean, and had redundancies, so we replaced it with better data, below.
  This also parsed some data associated which player's Fantasy League attributes, such as price, team, and role in the squad.

- **Rich Historical Stats Retrieval**
  *cricketstatsrequester.py*
  This is an offline function which we used to retrieve rich historical match by match player data in T20

[13] `http://www.iplt20.com/`

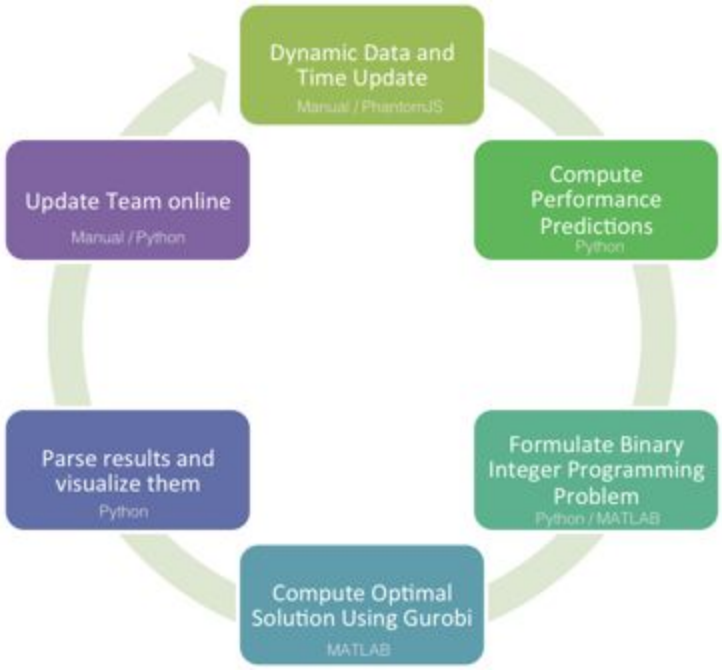


Figure 18: A high level visualization of the feedback loop of the entire framework.

matches from espnCricInfo [14], which is the most comprehensive and accurate data source for cricket T20 statistics we could find online.

- **Parse and Store Schedule**
  *parseSchedule.py*
  This offline function queried the IPL T20 2014 site again to parse and store a schedule as a pickle file for use in the simulation.

- **The Player Class**
  *player.py*
  This was a class used to store individual player data. It was convenient to query, and stored all the Fantasy League attributed, including name, team, age, role, and price as well as detailed past tournament aggregates and individual match history.

- **Live - BIP Formulation**
  *pickTeam.py*
  This is the first live code, used for the daily updating of my Fantasy League team. It takes in inputs - subs left, todays date, current team, schedule, and player statistical history [15], and formulates the Binary Integer Programming as described previously. It uses a $\gamma$ to compute predictions, assembles all the matrices required, and store all the data to passed onto Matlab in .mat files.

- **Live - Optimization Solve**
  *python2matlab.mat, cvx_ipl_optimization.m*
  At this point, the Matlab part of the system is run. It uses the data passed from the BIP formulation, and modifies it to create all the details of a BIP instance. It then performances the optimization with minimal precision (performance is more crucial for us) using Gurobi and CVXR and saves the results to a .mat file.

- **Live - Result Visualization**
  *showResults.py*
  Control is passed back into the Python script at this point. It reads the latest data that the solved optimization has outputted in its .mat files, using SciPy, and proceeds to interpret the binary numbers, and outputs a sequence of teams, which subs to make, the expected objective maximized, number of subs left, date, matches taking place, and other details necessary to visualize the result.

- **Complete Backtest**
  *backtest.py backtest_pickteam.py backtest_results.py*
  Complete backtest is an integrated version of the last 3 parts of the software for dynamic backtesting. It uses pyMatBridge to seamlessly switch control between Matlab and Python, and it runs multiple iterations to backtest a model, updating the data, time, player stats automatically.

[14] `http://www.espncricinfo.com/`

[15] This will soon be replaced by automated calls to the IPL Fantasy League server using PhantomJS and Ghost

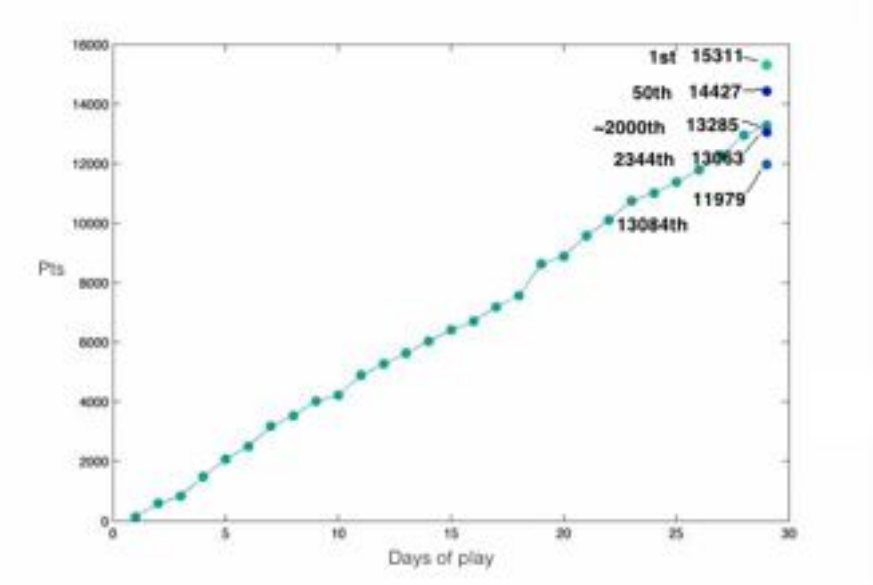


Figure 19: The result of the full backtest of the algorithm - 99.54%ile

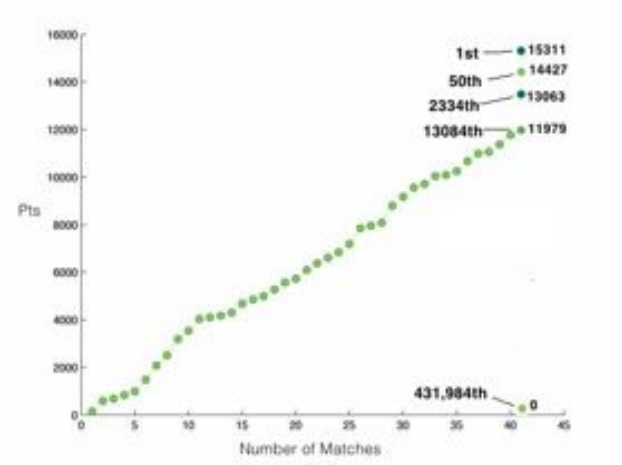


Figure 20: The result of the real algorithm's performance on the Fantasy IPL site - 97.23%ile

## 7 Results

As we discussed in Section 1.4, our algorithm provides two types of prediction - static and dynamic. Each optimization problem takes on average of 5 seconds to solve, with earlier instances of the time sometimes going beyond 100 seconds. This sometimes occurs early in the fantasy time period for dynamic testing, because the problem size is typically larger. The results for the dynamic are discussed in later sections.

In practice, the value of $n$, the number of players, was 178. The value of $q$, the number of days on which matches are played is 37. This gives us a total of $nq = 37 \times 178 = 6586$ binary variables. Our final optimization was of the form

$$\text{minimize} \quad Qx$$

$$\text{subject to} \quad Ax = B$$

$$\text{and} \quad Gx \leq H$$

$$\text{and} \quad \frac{\sum |x - \text{concat}(\boldsymbol{cur}, x[0 : n(q-1)])|}{2} \leq 75$$

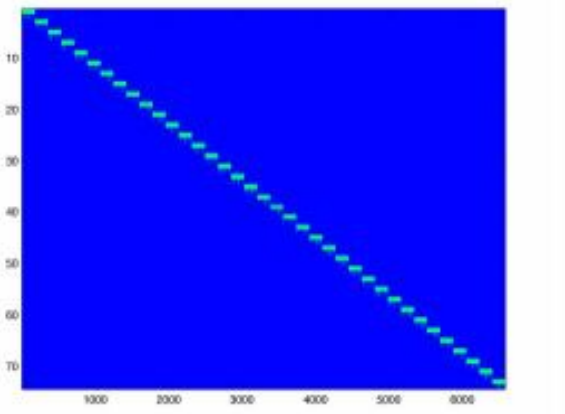

Figure 21: A visualization of the contents of equality constraints $A$. Most of the matrix is empty because these constraints attempt to restrict number of people and wicketkeepers in a team, and do not constrain relationships between teams at different times.

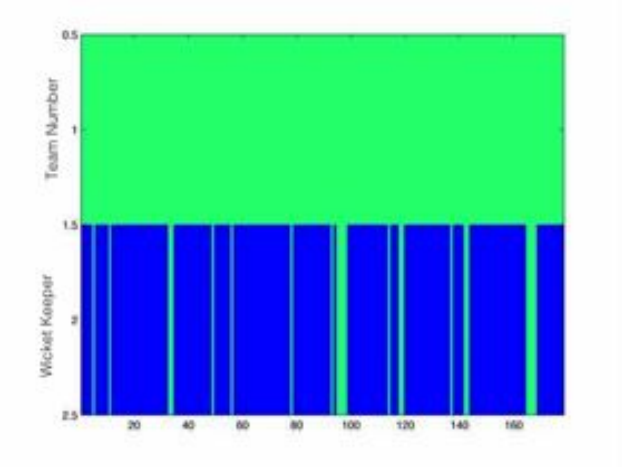


Figure 22: A visualization of the contents of $A$ for one team. The top represents the total team count (11), and the bottom represents the total wicketkeeper count (1)

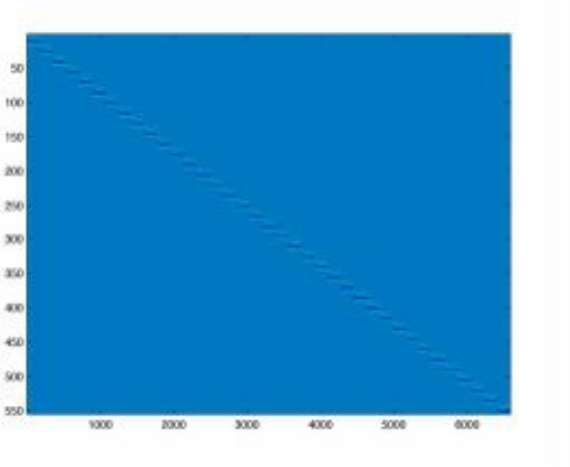

Figure 23: A visualization of the contents of inequality constraints $G$. Most of the matrix is empty because these constraints attempt to restrict quantities within a team, and not model inter-team restrictions

| | | |
|---|---|---|
| Q | $1 \times 6586$ | The vector of expected returns in the objective function |
| x | $6586 \times 1$ | Binary vector of decision variables to be computed |
| A | $74 \times 6586$ | Matrix of equality constraints - team size and wicketkeeper presence. |
| B | $74 \times 1$ | Vector of equality constraint expectations. |
| G | $555 \times 6586$ | Matrix of inequality constraints - squad balance, budget, etc. |
| H | $555 \times 1$ | Vector of inequality constraint restrictions. |
| cur | $1 \times 178$ | The binary vector denoting the current team possessed. |
| n | 178 | The size of the player pool |
| q | 37 | The number of days on which matches are played |

| | |
|---|---|
| Variables | 33523 |
| Equality Constraints | 13839 |
| Rows | 13839 |
| Columns | 33523 |
| NonZeros | 89296 |
| Quadratic Constraints | 13172 |
| Presolved Columns | 593 |
| Presolved Rows | 0 |
| Presolve Time | 0.26 |
| Continuous variables | 26344 |
| Binary variables | 6586 |
| Time | 58.82s |
| Best Objective | 18071.8pts |

The details of the variable sizes used is as follows:
Some statistics of a Gurobi solve of a Static Prediction (from the very beginning of the tournament, with no changing expectations) are as follows:
Our key results were:

- **Functionality** The optimization did what it promised fairly accurately, and we were able to run and use our algorithm to "trade" and decide teams in the IPL T20 Fantasy League.

- **Achieved 97.23%ile in the real fantasy league** Our algorithm has been in work ever since the start of the 2014 IPL on April 16. Therefore, the modifications to algorithm over time as well as a manual error[16], cost us. Nonetheless, as of 5/17, we accumulated **11979pts** and ranked **13,084th** out of **431,984**, which is a 97.23%ile - fairly successful results.

- **Achieved 99.54%ile in the world on backtest** In our live backtest, to account for the flaw in the real world fantasy league, we achieved **13285pts** up to 5/17, and ranked **2000th**[17] out of **431,984**, which is a 99.54%ile worldwide.

[16] On the first day of play, 7 subs were used to prepare the team for the first game (users have infinite subs available before the tournament). However, "lockdown" had happened, and these subs wouldn't be in effect for the 4/16 match. This effectively cost us almost 10% of our subs.

[17] The IPL leaderboard doesn't allow you to accurately check - just estimate.

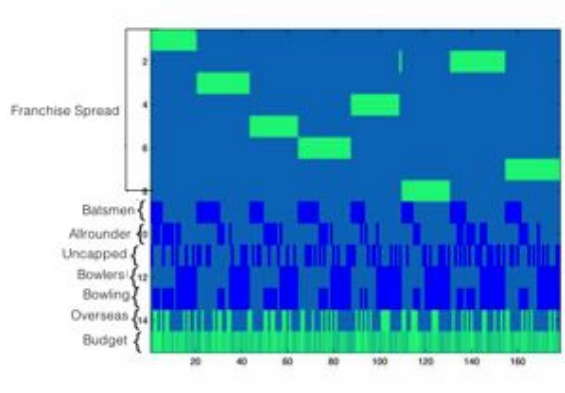


Figure 24: A visualization of the contents of $G$ for one team. It clearly shows the different constraints - the franchise constraints, batsmen constraints, all rounder constraints, uncapped player constraints, bowlers, bowling players, overseas players and finally the budget constraint. The dark blue ones are negative values because we require maximization of them, and the light green are positive values because we require mini fiction of them.

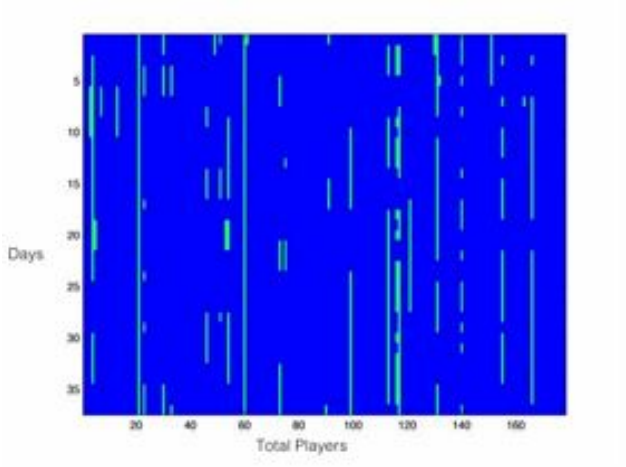


Figure 25: The change of the binary variables of the portfolio from week 1 to the end of the tournament, using the Static Prediction (constant expectation).

## 8 Analysis

We can summarize the major achievements of this paper and what we've learnt as follows:

- **33x better than previous software limits in the field** Previously, the field of team selection in cricket was bottlenecked by software speed. They previously could only simulate $n = 50$ players for $q = 4$ stages. With $n = 178$ and $q = 37$, our results eclipsed the former best by **33x**.

- **Novel study and results** Our study was novel in the sense that we were the first to actually apply optimization based techniques in a real Fantasy League, automate the entire process, automatically parse and update data, and create a framework for backtesting. Our results were fairly groundbreaking. No such results have been reported previously.

- **Created a backtesting framework for IPL cricket data** We were the first ones to create a complete backtesting framework for IPL cricket data. This is a *huge* achievement, because it creates a great way to test **performance predictors** in cricket, which is what most of the research on the field is about.

- **Acquired tons of accurate, structured data** We automated the data collection process and acquired a ton of accurate structured data for future use. Previously, primarily manual methods were being used.

- **BIP frameworks are successful in sports fantasy league prediction** We've established, by application, that BIP frameworks indeed work in fantasy league selection. Our results were far above baselines.

- **For Dynamic Prediction, greedy substitution restriction** One discovery we made was that the dynamic prediction framework set up in 1.4 doesn't quite function optimally. Although, given some information, it guarantees local optimality, when the information changes, that optimality is lost. For example, if I have a team $n$ at time $t$. Given a certain expectation, the algorithm deems it optimal to make 7 subs between $t$ and $t+1$, and claims the number of subs required later on will be minimal. Similarly, at $t+1$, with updated expectations, it tends to predict a high number of initial subs. This quickly depletes the number of substitutions you, as a manager, has. Consequentially, it is probably better to penalize a high amount of immediate subs in the objective function with a parameter lamb. The theoretical backing behind this warrants another paper.

- **Provide theoretical bounds on optimality** For static prediction, and dynamic prediction, we guarantee theoretical bounds on optimality. In other words, if our expectation $\gamma$ is identical to reality $r$ for all subsequent matches, we guarantee the optimal output with a precision of our input.

- **Independence assumption between player performances may be incorrect** As stated before, assuming that the performance of 2 players in the same team or not playing the same game is indepen-

dent is a simplifying assumption. Let's take a simple example. If one team has 8 players who've been performing well and the other has 1, then the optimization algorithm tends to return 5 players from team 1 and none from team 0. This is probably not optimal, because it's unlikely that all 5 players from the same team will perform well that day, even if they are in form, or expected to. This is discussed further in future work.

## 9 Future Work

Our research paves the way for various new work in the field of sport statistics, operations research and cricket, in particular:

- **Use Gamma distribution to rank cricketers** In Brettenny [2010], the author alludes to a method of ranking cricketers using a Gamma distribution which has proven to be more accurate than mere past statistics.
- **Account for variance** In our current work, we do not account for variance (or consistency) in a cricketer's performances, because unlike in Markowitz [Markowitz, 1952], our asset movement is expected to be normally distributed. However, a model which incorporates and tries to maximize consistent cricketers is a great future direction for this work.
- **Richer Cricket Statistics** One bottleneck of accurate complex modeling of inter-player and intra-team relationships is the lack of more detailed, consolidated statistics in cricket, exposed to an API. Knowing things like which bowlers the batsman is strong against, and weak against, the nature of the pitch, average scores on a certain ground and so forth, would really open up the possibilities when it comes to prediction in cricket, resulting in a model which would only be affected by *systematic risk* - luck.
- **Speed** Although we improved drastically over former works, speed in the dynamic backtest is still an issue. It can take unto a few minutes, and is unbounded in time. If the optimization problem does not converge to its precision restrictions quickly enough, it just tends to continue indefinitely.
  There are many prospective software modifications, as well as algorithmic modifications we could make to potentially allow backtests for multiple years in different formats of the game - not just one tournament.
- **Exploration in the space of performance measures** Our paper does not focus in the contrasting uses of different performance measures and which ones work worse or better. However, it sets up the perfect benchmark for such comparisons in the future.
- **Use of Support Vector Machines to predict performances** Certain Machine Learning techniques have been used by Saikia et al. [2012] fairly successfully in cricket prediction. Unlike finance, the use of Support Vector Machines has never been studied in a cricket setting. It would be an interesting direction to work with. Many financial institutions have had luck using machine learning approaches to trading and SVMs based on fundamentals may be successful.
- **Aggregation of External Factor Data** One of the biggest drawbacks of this models is the failure to predict the following unexpected events;
  - Players getting injured
  - Players being dropped from the side
  - Players leaving the tournament to play international cricket
  - The weather forecast for games.

  These unexpected events always happen at least thrice or four times in the space of a tournament, and lead to unexpected massive point losses. Knowing such things would be extremely useful for better performance.
  Currently, this information can only be injected manually into the model.
- **Modelling Inter-Team Interactions in a Black-Scholes like approach** Black-Scholes used options pricing predictions to hedge their risk in the financial markets [Black and Scholes, 1973]. There may be a similar approach to hedge risk in cricket using inter-team interactions. Picking both a good bowler from team A and a good batsman from team B, who are likely to face each other could be a hedged bet, because one of them will perform. This is just an idea to explore. It would probably require richer data, but it would be an interesting direction for statistical cricket analysis.

.

## References

Faez Ahmed, Abhilash Jindal, and Kalyanmoy Deb. Cricket team selection using evolutionary multi-objective optimization. In Bijaya Ketan Panigrahi, Ponnuthurai Nagaratnam Suganthan, Swa-


Figure 26: A detailed look at the schedule of matches so far in the IPL and the points received using my algorithm for each respective day of play.

gatam Das, and Suresh Chandra Satapathy, editors, *Swarm, Evolutionary, and Memetic Computing*, volume 7077 of *Lecture Notes in Computer Science*, pages 71–78. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-27241-7. doi: 10.1007/978-3-642-27242-4_9. URL http://dx.doi.org/10.1007/978-3-642-27242-4_9.

Gholam R Amin and Sujeet Kumar Sharma. Cricket team selection using data envelopment analysis. *European journal of sport science*, (ahead-of-print):1–8, 2012.

Gholam R Amin and Sujeet Kumar Sharma. Measuring batting parameters in cricket: A two-stage regression-owa method. *Measurement*, 53:56–61, 2014.

GDI Barr and BS Kantor. A criterion for comparing and selecting batsmen in limited overs cricket. *Journal of the Operational Research Society*, 55(12):1266–1274, 2004.

Dibyojyoti Bhattacharjee and Hemanta Saikia. Selecting the optimum cricket team after a tournament. *Asian Journal of Exercise & Sports Science*, 10(2), 2013.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The journal of political economy*, pages 637–654, 1973.

Warren Brettenny. *Integer optimisation for the selection of a Fantasy League Cricket Team*. PhD thesis, 2010. URL http://dspace.nmmu.ac.za:8080/xmlui/bitstream/handle/10948/1230/Warren\%20Brettenny.pdf.

Warren J Brettenny, David G Friskin, John W Gonsalves, and Gary D Sharp. A multi-stage integer programming approach to fantasy team selection: A twenty20 cricket study. *South African Journal for Research in Sport, Physical Education & Recreation (SAJR SPER)*, 34(1), 2012.

Uday Damodaran. Stochastic dominance and analysis of odi batting performance: the indian cricket team, 1989-2005. *Journal of Sports Science & Medicine*, 5 (4):503, 2006. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3861748/.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

Ian N Durbach and Jani Thiart. On a common perception of a random sequence in cricket: application. *South African Statistical Journal*, 41(2):161–187, 2007.

Hannah Gerber and Gary D Sharp. Selecting a limited overs cricket squad using an integer programming model. *South African Journal for Research in Sport, Physical Education and Recreation*, 28(2):p–81, 2006.

Guangliang He. The intuition behind black-litterman model portfolios. *Goldman Sachs Investment Management Series*, 1999.

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

Hermanus H Lemmer. The combined bowling rate as a measure of bowling performance in cricket. *South African Journal for Research in Sport, Physical Education and Recreation*, 24(2):p–37, 2002.

Hermanus H Lemmer. Performance measures for wicket keepers in cricket. *South African Journal for Research in Sport, Physical Education and Recreation*, 33(3): 89–102, 2011.

Hermanus Hofmeyr Lemmer. Team selection after a short cricket series. *European Journal of Sport Science*, 13(2):200–206, 2013.

Konstantinos Liantzas, Tim Matthews, Sarvapali D Ramchurn, and Georgios Chalkiadakis. Competing with humans at fantasy football: Team formation at large partially-observable domains. 2013.

Mark Lourens. *Integer optimization for the selection of a twenty20 cricket team*. PhD thesis, 2009. URL http://dspace.nmmu.ac.za:8080/xmlui/bitstream/handle/10948/1000/Integer\%20Optimization\%20for\%20the\%20Selection\%20of\%20a\%20Twenty20\%20Cricket\%20Team\%20by\%20Mark\%20Lourens\%20\%282008\%29.pdf?sequence=1.

Harry Markowitz. Portfolio selection*. *The journal of finance*, 7(1):77–91, 1952.

B Meindl and M Templ. Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*, 2012.

Hans D Mittelmann. An independent benchmarking of sdp and socp solvers. *Mathematical Programming*, 95 (2):407–430, 2003.

SN Omkar and R Verma. Cricket team selection using genetic algorithm. In *Proceedings of the International Congress on Sport Dynamics, Melbourne, Australia*, pages 1–9. Citeseer, 2003. URL http:

//citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.109.1230&rep=rep1&type=pdf.

Hemanta Saikia, Dibyojyoti Bhattacharjee, and Hermanus H Lemmer. Predicting the performance of bowlers in ipl: An application of artificial network. *International Journal of Performance Analysis in Sport*, 12(1):75–89, 2012.

Sujeet Kumar Sharma. A factor analysis approach in performance analysis of t-20 cricket. *Journal of Reliability and Statistical Studies*, 6(1):1, 2013.

Sujeet Kumar Sharma, Gholam R , and Said Gattoufi. Choosing the best twenty20 cricket batsmen using ordered weighted averaging. *International Journal of Performance Analysis in Sport*, 12(3):614–628, 2012.

GD Sharp, WJ Brettenny, JW Gonsalves, M Lourens, and RA Stretch. Integer optimisation for the selection of a twenty20 cricket team. *Journal of the Operational Research Society*, 62(9):1688–1694, 2011.

Amy E Summers, Tim B Swartz, and Richard A Lockhart. Optimal drafting in hockey pools. *Statistical Thinking in Sports, J. Albert and RH Koning (eds.), Boca Raton: Chapman and Hall/CRC*, pages 263–276, 2007.