COMPUTING MINIMUM FEEDBACK ARC SET TO QUANTIFY THE TRANSITIVITY IN SPORTS TEAMS

BY VISHALSINGH HAJERI

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Richard Martin

and approved by

New Brunswick, New Jersey

October, 2017

ABSTRACT OF THE THESIS

Computing Minimum Feedback Arc Set to Quantify the Transitivity in Sports Teams

by Vishalsingh Hajeri

Thesis Director: Richard Martin

This thesis fulfills the need for a system of optimal ranking of sports teams. The current ranking system used for ranking the teams in tournaments like English Premier League (EPL) and Indian Premier League (IPL) only account for wins and losses, while the approach in this thesis better accounts for how teams perform against each other and the relative strength between them, for example, this approach accounts for 'upsets' where a lower ranked team beats a highly ranked team. A tournament can be represented as a weighted directed graph G, where a directed edge E from team A to team B represents that Team A beat Team B and the edge weight constitutes the Goal Difference (GD) or the points differential between the two teams for that particular match. A feedback arc set (FAS) of G is a subset of its edges which contains at least one edge of every cycle in G. These set of edges which when removed from G, leaves us with a Directed Acyclic graph (DAG). The aim of this project is to solve 'Minimum Feedback Arc Set problem' (MFAS) which in other words is removing minimum number of least weighted edges from G such that the remainder of the graph G' results in a DAG. We solved the minimum feedback arc set problem by framing it as an optimization problem and utilized Gurobi Optimization Library

to get the results. A linear Ordering of the resulting DAG gives the ranking of teams in the tournament.

Sub-part of this project is to calculate Coefficient of Transitivity (CoT) of a Tournament. The transitivity of a sports tournament would manifest itself as when a Team A, dominated Team B with a large point differential, and Team B dominated Team C, then Team A should also dominate Team C. However, if Team C beats Team A then we call it as an 'upset' and such upsets gives rise to cycles in the tournament graph G and thus decreases the transitive nature of the tournament. An important finding of this project is that the balance among sports tournaments like Soccer, Football and Cricket can be characterized by calculating the CoT for a particular season.

Acknowledgements

I would like to thank my adviser, Prof. Richard Martin, without whose help and guidance I would not have completed this project.

Dedication

This work is dedicated to everyone who supported me through this endeavor.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Tournaments are generally played either in a Round-Robin fashion like EPL (Soccer), IPL (cricket) or Single elimination fashion like Tennis and Chess. One good thing about Round Robin tournaments is that each team/player gets to play every other team/player in the tournament. Hence, the optimal ranking order generated would be fair. On the contrary, it becomes difficult to decide a ranking order in Single Elimination tournaments and other multilevel tournaments as all teams/players play an unequal number of matches.

For Sports, we have a tendency to compare the teams' strength which leads us into judging one team being "better than" the other and intuition tells us that the “better than” property manifests itself as point differentials in games. However, it is not uncommon to observe that sometimes a weaker team beats the stronger team and such scenarios call into question how often the "better than" property holds in practice. The question we want to raise is how confidently can we predict a winner between two teams by looking at their performances in other games. If sports teams are strongly transitive, then round-robin style tournaments are not really needed. On the other hand, if transitivity is weak, then many ranking functions currently employed may fail in providing an optimal ranking order.

In this project we have introduced a novel transitivity metric called the *Coefficient of Transitivity* or CoT, in order to quantify the "better than" property of sports teams. It is a real-valued number between 0 and 1 that measures the transitivity of a graph. The CoT is a single number designed to convey how much of a graph is directed back on itself. A Directed Acyclic Graph (DAG) would have a coefficient of 1, as it very transitive. A perfectly balanced graph, whose edges "fold back" on each other in equal proportions, would have a value of 0, as it is not very transitive. Graph with some backward edges would have a value somewhere between $0 - 1$.

1.2 Minimum Feedback Edge Set Problem

In the contexts of sports tournaments, when two teams play each other, they generate a graph with two nodes and one directed edge. The nodes are meant to represent the teams, and the edge direction is meant to be pointing towards the losing team while edge weight would represent the point differential between the two teams. Over the course of a season, this graph will keep expanding as more games are played. More the number of teams participating in the tournament, denser the graph becomes and thus more cycles may be formed. It is to be noted that there is exactly one edge between two nodes.

A **feedback arc set** (FAS) F is a subset of edges containing at least one edge of every cycle in a directed graph G. One of the properties of FAS is that removing the edges in FAS from the original graph G makes the remaining graph G' a directed acyclic graph (DAG). It can also be noted that reinserting one of the edges for FAS into G' will induce a cycle and we will no longer have DAG. Obtaining DAG is a necessary step in computing the linear ordering of the teams. Obtaining a Feedback Arc Set of minimum cardinality refers to the **Minimum Feedback Arc Set Problem** (MFAS). In case of weighted edges we want to minimize the cost of the FAS. This cost is nothing but the total sum of all the edge weights in FAS. **Maximum acyclic Subgraph Problem** is complementary to the MFAS problem. Therefore, a solution to one of those problems usually yields a solution to the other.Next, we define a few terms which will be helpful in the later chapters

A directed path from Node u to Node v is an alternating sequence of Nodes and edges of G leading from u to v. A path that starts and ends at the same Node is called a cycle. A **simple cycle** is a cycle with no repeating edges and no repeating nodes in the cycle.

A **topological sort** of G is a linear ordering of all its nodes such that if G contains an edge from u to v, then u appears before v in the ordering. The nodes in a directed graph can be arranged in a topological order if and only if the directed graph is acyclic. Time Complexity of Topological sort is O (N + E). where N is the total number of nodes in the graph G' and E is total number of edges in G'. Apart from its use in order to rank all the teams in the tournament, we have also used topological sort to detect cycles in the subgraph.

A strongly connected component (SCC) of a directed graph G is a maximal set of vertices C $\subseteq$ N such that for every pair of Nodes u and v in C, there is a directed path both from u to v and from v to u (u and v are reachable from each other). The strongly connected components of a directed graph can be found in linear time, that is, in O(N + E). We mention two types of SCC's in the later chapters, Trivial SCC and Non-Trivial SCC. A trivial SCC consists of a single node. It may have self-loops, but in our case of tournaments there are no self-loops in G. A non-trivial SCC consists of atleast 3 nodes as there is only 1 edge allowed between each pair of nodes.

Chapter 2

Related Work

- **MFAS Survey**

  The Feedback arc set problem in tournaments is well studies from combinatorial [1],[25], statistical and algorithmic[8],[10],[19] point of view.

  It has been known for a long time that the computation of a minimum feedback arc set is an NP-hard problem for general directed graph. This problem is included in the Karp's 21 NP-complete problems that are a set of computational problems which are NP-complete[21]. The complexity status of this problem for tournaments remained open for a long time, though several authors conjectured quite soon that it remains NP-hard for this kind of graph. One of the ways to cope with hard problems in practice is to realize that problem instances may be associated with some "parameters" related to their complexity. If the parameter of an instance is small, we might have some hope in finding a polynomial time solution to that instance. This idea has been investigated in the 1990's by Downian and Fellowes in their work on fixed parameter tractability and the parameterized complexity of the feedback arc set problem applied to tournaments has also been studied by Raman and Saurabh (2006) [26].They have shown that the feedback arc set problem for weighted or unweighted tournaments is fixed-parameter tractable (FPT) by providing efficient algorithms.

- **Probabilistic Transitivity in sports :** Johannes and Philipp [30] in their paper have constructed a statistical model that describes the outcome of sports matches. They view the current methods to be arbitrary in finding the "right" ranking order. The way they approached this problem is by assuming that there indeed is a "correct ranking" and they try to find the one which is most likely identical to it. They have begun with the assumption that the outcome of each match follows a trinomial distribution, with a fixed probability for loss, win and tie. The branch and bound algorithm which they put forth is capable of solving the problem for up to 10 teams. Along with that, they also propose a tabu search heuristics for larger data sets.

- **Heuristics** The literature on various heuristics for the minimum feedback arc set and linear ordering problem is overwhelming and we resort to mentioning only a few noteworthy amongst many. Since the MFAS problem is approximation resistant, difference between the solution found by heuristics and the optimal solution can be as large as O(n).
  - Rather simpler form of solving the MFAS problem is applying sorting heuristics in which we arbitrarily order the given nodes in G and then we can unambiguously categorize all the edges as either forward or backward edges depending on whether the sink node of the edge appears after or before the source node of the same edge. In the former case the edge is a forward edge (it is pointing forward in the ordering); in the latter case it is a backward edge. These set of backward edges form the MFAS.
  - The algorithm by Eades et al.[2] can be regarded as a two sided selection sort, which looks for the smallest and largest elements and moves them left and right, respectively. Elements are the nodes in the graph, which are organized as left and right stacks. At any stage, first sources are removed from the graph and appended to the left stack, whereas sinks are removed and added to the right stack. The decision of assigning a stack to the node depends on $max\left\{|out(N_i)-in(N_i)|\right\}$, which gives the absolute value of the degree difference. $out(N_i)$ is the number of outgoing arcs from the node, while $in(N_i)$ is the number incoming arcs for the node. Finally, the left and right lists are concatenated to give the relative order.

- In [19] and in [29] authors have provided solutions for the real world scenarios by solving the MFAS problem. In [19], authors have put forth a cutting plane procedure for the solution of the linear programming relaxation of the linear ordering problem. This paper is of particular interest to us because they have compared the results of the actual standings of the German soccer championship season 1981/82 with the rankings obtained by linear optimum ordering. We too have taken a similar approach to obtain a ranking order for the teams participating in sports tournaments and compare them with actual league standings. They also mention that the optimum linear ordering obtained is not unique and it is possible to obtain more than one optimum rankings. They have also produced extensive results for the input-output analysis on the economy of a region (usually a state). [29] focuses on solving the problem of machine translation in the field of natural language processing It provides an example of difference between the ordering of verbs in English and German language. German verbs often occur at the very ends of their clauses. English verbs, on the other hand, usually occur between their subjects and their objects. Their solution combines a reordering model derived from the Linear Ordering Problem and other permutation problems with monotone finite-state translation.

- **Elo Rating System:** Originally invented by Hungarian master level chess player Arpad Elo as a way of comparing the skill levels of chess players. Its strong predictive power soon got the attention of sport analysts, who have since adapted it for several sports and it is now widely used in assessing the performance of soccer teams. The aim of Elo ratings in soccer is to measure the relative strength levels of the teams. Each club has a single Elo value for each point in time, with higher figures indicating stronger teams.

  When clubs play against each other, the winning side takes points from the loser, with the exact number of points determined by the Elo difference between the two teams. If a team plays an opponent who is significantly weaker and loses, the rating would reduce a lot more than if the team lost to an opponent who is only slightly weaker. Similarly, beating much stronger opponents earns a team more points than beating only slightly stronger teams. In case of a draw, the lower rated club will take points from the higher rated club, thus making the system self-correcting.

Table 2.1: Rankings based of Elo Rating as of 05/21/2017

Rank	Club	Elo Rating
1	Chelsea	1906
2	Tottenham	1885
3	Man City	1864
4	Man United	1855
5	Arsenal	1845
6	Liverpool	1835
7	Everton	1748
8	Leicester	1714
9	Southampton	1689
10	West Ham	1669
11	Stoke	1659
12	Bournemouth	1651
13	Swansea	1646
14	West Brom	1642
15	Crystal Palace	1640
16	Burnley	1626

It would be unwise to compare these rankings with our results because we have calculated rankings on a season by season basis, while Elo Rating system is in place since the inception of EPL and it keeps updating the rankings on a match by match basis. Table 2.1 provides the latest rankings of the top 16 teams in the English Premier League as obtained from `http://clubelo.com/ENG`

Chapter 3

Coefficient of Transitivity

We have introduced CoT as a novel metric for measuring transitivity in Sports Tournaments. Transitive functions have a unidirectional quality. The most used example of a transitive function is the "greater than" operator. So, as $10 > 9$, and $9 > 8$, then $10 > 8$. Thus a transitive function can be represented by a Directed Acyclic Graph (DAG) and a Linear Ranking Order can be obtained.

Some papers like [8],[9] have worked on similar problem by representing the graph having unit edge weights. Edge weight of 1 is assigned for a win while an edge weight of 0 is assigned for a tie. However, with such a technique of edge weight assignment it is difficult to gauge the point differential between the two teams. We therefore decided on assigning the Goal Difference as edge weight which gives a better idea of how close was both the teams' performances or how badly either of the teams performed in comparison to the other. The COT seeks a metric of how strong the "backward" edges are. The more the backward edges weights, the less the transitivity. Any DAG is maximally transitive, and thus should have maximum score of 1. On the other hand, a balanced graph should have a transitivity of 0.

The COT is computed by taking the difference between the sum of the edge weights of the FAS and the DAG, and then dividing the difference by the sum of all the edge weights. If the FAS is empty, a zero is used. Lets denote tournament graph as G, FAS as F and the Graph remaining after the removal of Feedback edges as G'.

$$COT = \frac{|G' - F|}{G} \tag{3.1}$$

If there are no edges in the DAG, the COT is undefined.

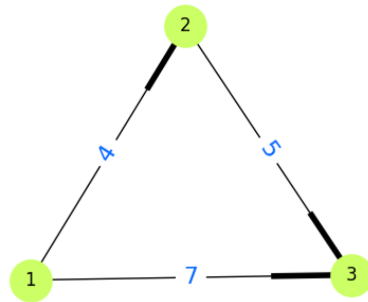


Figure 3.1: Sample Tournament Scenario - 1

Figure 3.1 represents a graph where there are three teams in the tournament and each team has played against each other. To calculate COT, we perform the following steps.

1. Since there are no feedback arcs in this Graph, the total edge weight for F = 0

2. Sum of Edge weight for Graph G' is 4 + 5 + 7 = 16

3. Total sum of edge weights in F and G' is 16 + 0 = 16.

4. COT is $\frac{16-0}{16} = 1$

As we observe from the graph that input graph G was already a DAG and there were no Feedback arcs to be removed. Thus as Team 1 beats both team 2 and team 3, it would be Ranked No. 1, while team 2 beats team 3 and hence it would assume the No. 2 position and Team 3 will come in last.

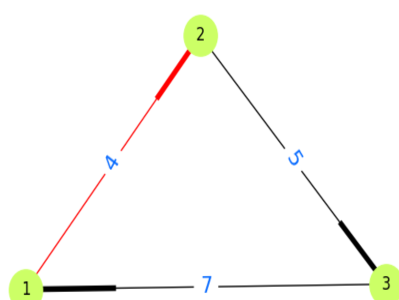


Figure 3.2: Sample Tournament Scenario - 2

Figure 3.2 represents a graph similar to that in 3.1 but only with one major difference of a feedback arc present in the graph. We resort to the convention of assigning a Red color to the feedback edges which are a part of FAS and Black Color to the edges of DAG. We follow the process of COT calculations as below.

1. Sum of edge weights in F = 4
2. Sum of Edge weight for Graph G' is 5 + 7 = 12
3. Total sum of edge weights in F and G' is 12 + 4 = 16.
4. COT is $\frac{12-4}{16} = 0.5$

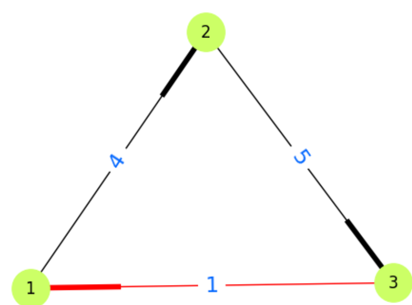


Figure 3.3: Sample Tournament Scenario - 3

One major distinction between Figure 3.3 and figure 3.2 is the number of Goals scored between the match of Team 1 and Team 3. In scenario - 2 Team 3 beat Team 1 convincingly with a Goal Difference of 7 while in scenario-3 Team 3 just barely managed to beat beat Team 1 with a Goal Difference of 1. Thus as per the concept of Minimum Feedback arc set we remove the minimum number of least weighted edges from G. Thus, we see Edge (3,1) being added to FAS. By following the procedure to calculate the COT, we get:

COT is $\frac{9-1}{10} = 0.8$

Chapter 4

Data

We have focused our attention on Round Robin Tournaments because there is symmetry in the input data as every team gets an opportunity to face every other team in the tournament. In multi-level elimination tournaments like chess and Tennis every player plays an unequal number of matches. Hence, Ranking order generated from such a tournament may not do justice with team/player rankings. For example, a top seed player in Tennis may have a bad day and he may get eliminated in the first round itself, robbing him the opportunity to prove his merit.

We obtain the data from different sources. We have focused on the following three tournaments.

1. **English Premier League(EPL):** We chose this tournament primarily because of its popularity as it is the most-watched sports league in the world, broadcast in 212 territories to 643 million homes and a potential TV audience of 4.7 billion people. The other main reason for research on this tournament is because of the consistency in the competition format across all seasons since its inception in 1992. There are 20 clubs in participating in the EPL. During the course of a season every team faces every other team exactly twice. One home game and one away game for a total of 38 games. The tournament uses a 3-point system to rank the teams where three points are awarded for a win and one point for a draw. No points are awarded for a loss. Teams are ranked by total points earned during the course of the season, then goal difference, and then goals scored. If still equal, teams are deemed to occupy the same position.

The total number of games played in the tournament will be 2 times ${}^{n}C_2$, where n is the total number of teams participating in the tournament which is 20. We get a total of 380 games per season. Since, we have only one directed edge between two nodes, we pre-process the input data to generate an edge weight and edge direction between two teams by calculating the net goal difference form home and away matches. Thus the maximum possible edges in the tournament graph could be 190. Edges having a zero weight can be left out from the graph, since they won't make a difference in the COT calculations.

Table 4.1 illustrate the Input Format for the pre-processing stage of the Algorithm and figure 4.1 is the resultant edge formed from that sample input. All the 20 clubs participating in the tournament have a Team ID assigned to them. Everton's ID is 4 while chelsea's ID is 6. These two teams clash exactly twice during the course of the season. From the data obtained we find that Chelsea has scored a total of 7 goals in those two matches while Everton has managed to score only 3. Thus, a Goal Difference of 4 assumes the value of the edge weight and the edge direction pointing towards the less superior of the two teams. Intuitively we can judge the superiority of a particular club by looking at the in-degree and out-degree of that particular node. Greater the In-degree, less superior the team is, while greater out-degree hints towards the team being more dominant in the tournament.

Table 4.1: Sample Input Format for EPL

HomeTeam	Home Team ID	AwayTeam	Home Team ID	FTHG	FTAG
Everton	4	Chelsea	6	3	6
Chelsea	6	Everton	4	1	0

FTHG = Full Time Home Goals

FTAG = Full Time Away Goals

We have obtained data from http://www.football-data.co.uk/englandm.php for 1993/94 season to all the way upto 2016/17 season.

2. **Indian Premier League (IPL):** IPL is a professional Twenty-over cricket league in India

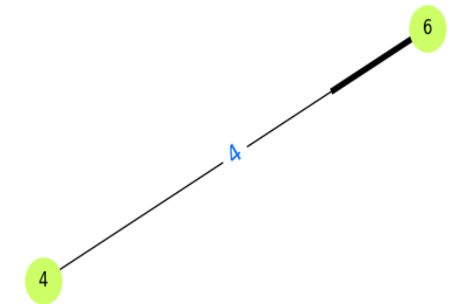


Figure 4.1: Sample generated Edge in EPL

contested during April and May of every year by teams representing Indian cities. Since its inception in 2008 there have been 8-10 teams participating in each season. Currently, with eight teams, each team plays each other twice in a home-and-away round-robin format in the league phase. At the conclusion of the league stage, the top four teams will qualify for the playoffs. Two out of these four teams are eliminated during the qualifying matches and the winner of the tournament is determined by the final match between the top 2 teams.

We ignore the final three matches played during the playoffs and focus only on the matches played during the league stages, as this helps in maintaining the symmetry in the input data. Since each team plays a Home and Away match with every other team, the total number of games played in the tournament will be 2 times ${}^{n}C_{2}$, with n being equal to 8, we get a total of 56 matches played during a season. IPL uses a 2-point ranking system to decide the team rankings. In this system, 2 points are awarded to the winning team, 0 points for the losing team and 1 point to each of the teams in a scenario where the match has to be called-off. In case of ties, each team proceeds to play a super over and a winner is decided. We obtained the data from https://www.kaggle.com/harsha547/indian-premier-league-csv-dataset which maintains a detailed record of every match played in the tournament.

There is a slight difference between IPL and EPL in terms of how the edge weights are calculated. Unlike Soccer or Football, matches in cricket are either won by the differential in the runs scored by the two teams or based upon wickets in-hand when the score was chased down

Table 4.2: Sample Input Format for IPL

HomeTeam_ID	AwayTeam_Id	Win_Type	Won_By	Winner_Id
3	2	by runs	21	3
2	3	by wickets	3	3

by the winning team. Table 4.2 is a depiction of the sample input format for the IPL dataset. "Chennai Super Kings" (CSK) have a team ID of 3 , while "Royal Challengers Banglore" (RCB) have a team ID of 2. Data shows that both the matches were won by "Chennai Super Kings".

- Unlike soccer where the performance difference between the two teams is accounted by the goal difference, in cricket there are two different performance metrics that differentiate the two teams. Firstly, the performance of the two teams can be differentiated based upon the differnce in the runs scored by them. Secondly, the performance can also be differentiated based on the total wickets it took to achieve a particular score. As seen form 4.2 in match one team CSK batted first and won by a margin of 21 runs, while in the later match they batted second and chased down the opponent's score to win the match with 3 wickets to spare. In the first instance, the performance metric is runs while in the second, it is wickets. In order to normalize the "won-by" column we decided to make a minor adjustment by making 10 runs = 1 wicket. The won-by column is then rounded-up to the nearest integer. The resulting edge will have a weight of 5 (2+3) and it will be directed towards node two as shown in figure 4.2.

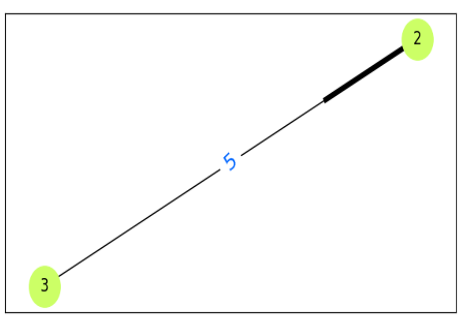


Figure 4.2: Sample generated Edge in IPL

3. **National Football league (NFL):** Concerning American football, we will focus exclusively on the NFL. It is a professional American football league consisting of 32 teams, divided equally between the National Football Conference (NFC) and the American Football Conference (AFC). The National Football League runs a seventeen-week, 256-game regular season. We have data on the scores of every NFL game since 1978 from the website http://www.repole.com/sun4cast/data.html. The NFL comprises from 28 (in the season 1978) to 32 teams in 2017. The following link *http://www.drwagpicks.com/p/nfl-statistics-downloads.html* also provides useful statistical information about each game. This is by far the largest group of teams we are analyzing and quite obviously there are many teams that do not face each other during a season as each team plays a total of only 16 matches in the regular season. NFL regular season matchups are determined according to a complicated scheduling formula which shall not be further discussed here. In football draws are possible, but only happen very rarely.

Chapter 5

Integer Programming Approach to compute MFAS

The FAS can be computed using standard integer programming techniques. For the graphs generated by sports teams, standard integer programming techniques are quite feasible as the size of the graphs is not very big. For example, the NFL data set for the 2012/2013 season only has 32 nodes and a maximum of 267 edges.

5.1 Gurobi

The Gurobi Optimizer is a state-of-the-art solver for mathematical programming and we aimed at converting the minimum feedback edge set problem into an optimization problem by utilizing linear programming Solvers to find the solution. We needed a license in order to install and use the Gurobi Optimizer. We were able to obtain free academic license available for students. To explain in simple terms, Gurobi optimizer allows us to state the MFAS problem as a classical optimization problem and then automatically considers billions of possible solutions to find the best one. It used all four of the available processors to perform the computations and it produced results within a few seconds.

In order to formulate MFAS as an optimization problem, we'll need to do three things.

1. First, we'll need to define the decision variables. The goal of the optimization is to choose values for these variables. They capture the results of the optimization. In a feasible solution, the computed values for the decision variables satisfy all of the model constraints.

2. Second, we'll define a linear objective function. This is the function we'd like to minimize.

3. Third, we'll define the linear constraints. A constraint in Gurobi captures a restriction on the values that a set of variables may take. linear constraint is the simplest type of constraint, which states that a linear expression on a set of variables take a value that is either less-than-or-equal, greater-than-or-equal, or equal another linear expression

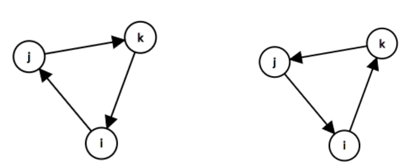
The Gurobi Optimizer will consider all assignments of values to decision variables that satisfy the specified linear constraints, and return one that optimizes the stated objective function.

5.2 Integer Programming formulation with triangle inequalities

The equations in this section are referred from [2], [19] which proposes multiple exact methods in obtaining MFAS based on integer programming approach.

Equation 5.1 defines a Linear Objective function which needs to be minimized by the Gurobi model. We seek a minimum cost ordering π^* of the input graph G having N nodes E edges. $c_{i,j}$ denotes the cost associated with the directed edge (i,j) $\in$ E. $c_{i,j} = 0$ if it is $\notin$ E. $x_{i,j}$ in the equation represent the binary variables associated with a given ordering π. Decision variables capture the results of the optimization. In a feasible solution, the computed values for the decision variables satisfy all of the model constraints. $x_{i,j} = 0$ if node i precedes node j in π else $x_{i,j} = 1$ otherwise. Any ordering π uniquely determines a corresponding x.

To get a Linear ordering from the given tournament, we have to formulate the inequalities in such a way that we can detect and avoid such a scenario where i comes before j, j comes before k and k comes before i. Equivalently, we prevent $x_{i,j} = x_{j,k} = x_{k,i} = 1$.



(a) (b)

Table 5.1: Scenarios that lead to the formulation of triangle inequalities

Scenario (a) is prevented by the equation $x_{i,j} + x_{j,k} - x_{i,k} \leq 1$, while scenario (b) is prevented by $-x_{i,j} - x_{j,k} + x_{i,k} \leq 0$. Equations 5.2, 5.3 , 5.4 set the constraints on the decision variable and they are called the triangle inequalities. Any x that satisfies the triangle inequalities (3) must correspond to an ordering as proven in [24],[19]. Note that there are O(n2) binary variables, and O(n3) constraints in our model.

$$min \sum_{j=1}^{n} (\sum_{k=1}^{j-1} c_{k,j} x_{k,j} + \sum_{l=j+1}^{n} c_{l,j}(1 - x_{j,l})) \tag{5.1}$$

$$x_{i,j} + x_{j,k} - x_{i,k} \leq 1, \qquad 1 \leq i < j < k \leq n \tag{5.2}$$

$$-x_{i,j} - x_{j,k} + x_{i,k} \leq 0, \qquad 1 \leq i < j < k \leq n \tag{5.3}$$

$$x_{i,j} = \{0, 1\}, \qquad 1 \leq i < j \leq n \tag{5.4}$$

5.3 Hardware and Software Environment

Processor: Intel(R) Core(TM) i7-5500U CPU @ 2.40 GHz with 8 Gb RAM.

Operating system: Windows 10; State of the art integer programming solver Gurobi 7.5 was called through its API from Python 2.7.11; Gurobi uses all 4 of the available processors to do the computations. NetworkX (version 1.11) is a Python language software package that provides a standard programming interface for graph implementation and manipulation.

5.4 Linear Ordering of Teams

Gurobi computations explained in the previous sections results in MFAS. These set of edges can also be referred to as "torn edges", since their removal(being torn) from G results in a DAG. Once we have obtained a DAG we can focus on the next part of the project, which is linear ordering of the nodes to generate a ranking for the teams. Our approach is based on a simple fact that A DAG has at least one node with in-degree 0 and one node with out-degree 0. Proof to the above fact is that a DAG does not contain a cycle which means that all paths will be of finite length. Now let S be the longest path from u(source) to v(destination). Since S is the longest path there can be no incoming edge to u and no outgoing edge from v, if this situation had occurred then S would not have been the longest path => indegree(u) = 0 and outdegree(v) = 0.

We define the term Degree(N_i) = OutDegree(N_i) - InDegree(N_i). Out-degree for a Node represents the number of matches won by the team , while in-degree represnts the number of matches lost by team. So the degree of node can act as a tie-breaker in a scenario where the algorithm described below finds more than one node suitable for a particular rank.

Algorithm 1: Find Linear ordering of Teams in DAG

```
Input: DAG with N nodes
Output: List of Linearly ordered Nodes
begin
    Initialize a list LinearOrder[]
    for node ← 1 to N do
        indegree[node] ← 0
    for edge ← Edges(src, dst) do
        indegree[dst] ← indegree[dst] + 1
    for node_i ← 1 to N do
        if indegre(node_i) = 0 then
            Queue.append(node_i)
    while Queue is not empty do
        u ← Queue.front
        LinearOrder.append(u)
        for n_i ← AdjacentNodes(u) do
            indegree[n_i] = indegree[n_i] − 1
            if indegre(node_i) = 0 then
                Queue.append(node_i)

    return LinearOrder
```

Chapter 6

Results

6.1 English Premier League

Table 6.1 populates the CoT values calculated after the MFAS produced by the Integer Programming Formulation discussed in section 5.2.

In all seasons of EPL only 20 teams participate in the Tournament. We also ignore those edges in G which have a 0 weight, as there exclusion won't make any difference on CoT computations and it might eliminate any simple cycles.

- Total Nodes N = 20
- Total Edges E $\leq$ 190
- Weight of G is the sum of all the edge weights of the Tournament Graph G
- Weight of FAS is the sum of all the weights of edges which are a part of FAS.
- Torn Edges is cardinality of all the edges in the FAS
- CoT is the transitivity metric with a value between 0 - 1 of one particular season of the EPL calculated by examining data of up to 380 games.

Figure 6.1 gives a graphical representation of the variation of CoT values season by season.

Since, we are the first one's to introduce CoT metric we didn't really find any related material to compare our results. We, also went a step ahead and calculated the linear order of the teams to rank them. The ranking we came up with is compared with the 3 - point ranking system of

Table 6.1: Computational Results for EPL tournament

Season	Weight of G	Weight of FAS	Torn Edges	CoT
2016/17	448	26	21	0.8839
2015/16	362	30	17	0.8342
2014/15	379	39	25	0.7941
2013/14	482	31	19	0.8713
2012/13	379	37	23	0.8047
2011/12	438	39	27	0.8219
2010/11	381	53	37	0.7217
2009/10	461	26	20	0.8872
2008/09	402	39	28	0.8059
2007/08	450	37	23	0.8355
2006/07	365	31	22	0.8301

the EPL in Table 6.2 for 2014/15 season. We have performed similar comparisons across all seasons, but we decided on providing results for only one of the season here.

By looking closely at the Table 6.2 and comparing the two ranking orders we can say that even though teams rankings seem almost similar in the two ranking systems, there are teams like Crystal Palace, Swansea City and QPR which have shifted significant number of places in our results as compared to 3-point system. Such difference in the ranking system can be attributed to the fundamental approach in implementation of both the ranking systems. We factor in the strength of the opponent: Beating a higher ranked team or beating a team with a higher goal difference would be major factor in team being ranked higher. While, 3-point system directly awards 3 points for a win, no matter how high or low the win margin. The Goal Difference comes into consideration only when there is a tie between the two teams.

On a deeper analysis of the dataset we observed that one of the reasons why Crystal Palace jumped from Rank 10 in the Actual Standings to Rank 5 in our results could be because they managed to beat top teams like Tottenham, Manchester City, Liverpool, Swansea City and Stoke City.

Table 6.2: Comparison of EPL standings with Gurobi Results for 2014/15 Season

RANK	Gurobi Results	Actual Standings	MP	W	D	L	GD	PTS
1	Chelsea	Chelsea	38	26	9	3	41	87
2	Man United	Man. City	38	24	7	7	45	79
3	Arsenal	Arsenal	38	22	9	7	35	75
4	Man City	Man United	38	20	10	8	25	70
5	Crystal Palace	Tottenham	38	19	7	12	5	64
6	Stoke	Liverpool	38	18	8	12	4	62
7	Liverpool	Southampton	38	18	6	14	21	60
8	Tottenham	Swansea City	38	16	8	14	-3	56
9	Southampton	Stoke City	38	15	9	14	3	54
10	Everton	Crystal Palace	38	13	9	16	-4	48
11	West Ham	Everton	38	12	11	15	-2	47
12	Leicester	West Ham	38	12	11	15	-3	47
13	Swansea	West Brom	38	11	11	16	-13	44
14	Newcastle	Leicester City	38	11	8	19	-9	41
15	Hull	Newcastle	38	10	9	19	-23	39
16	QPR	Sunderland	38	7	17	14	-22	38
17	Aston Villa	Aston Villa	38	10	8	20	-26	38
18	Sunderland	Hull City	38	8	11	19	-18	35
19	West Brom	Burnley FC	38	7	12	19	-25	33
20	Burnley	QPR	38	8	6	24	-31	30

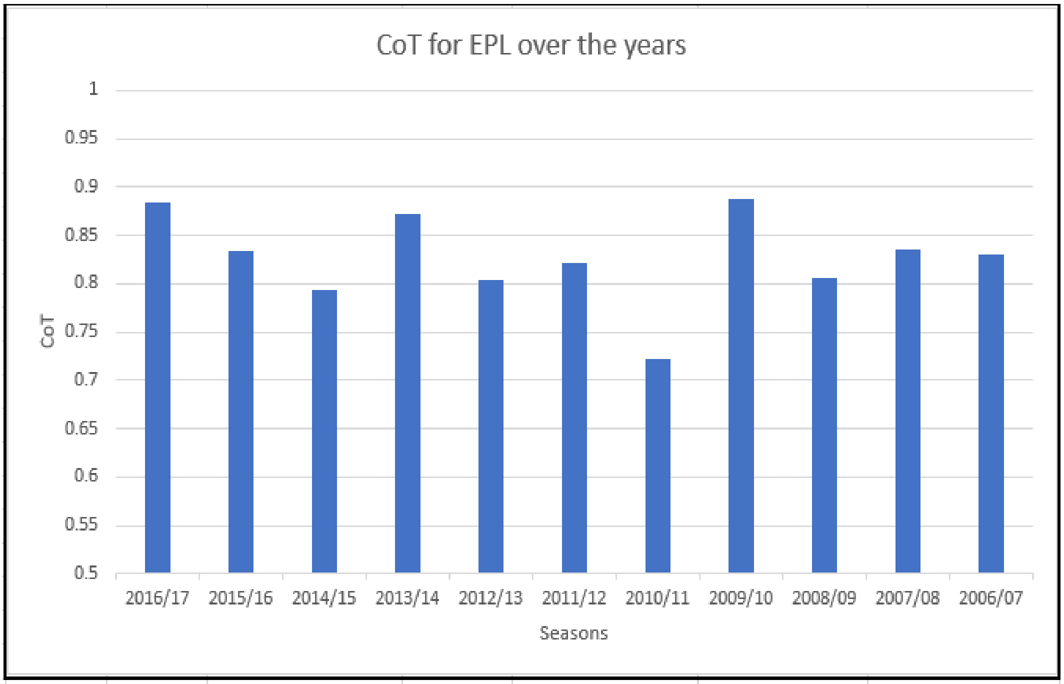


Figure 6.1: CoT spread across EPL seasons

6.2 Indian Premier League

Similar to the previous section, table 6.3 populates the CoT values for every season of IPL. Teams participating in an IPL tournament vary from 8 to 10 as seen by the No. of teams column in the table 6.3. There can be 28 edges at most in the graph which gives us an opportunity to test our algorithm on relatively sparse graph.

- Total Nodes N = 8 to 10
- Total Edges E $\leq$ 28

Figure 6.2 gives a graphical representation of the variation of CoT values season by season.

Table 6.4 provides the comparison of the ranking order obtained by our approach with the 2-point ranking system employed by the IPL. There are three teams highlighted in the table which differ in the rank position between the two ranking systems and Figure

Table 6.3: Computational Results for IPL tournament

Season	No. of Teams	Weight of G	Weight of FAS	Torn Edges	COT
2016	8	152	5	2	0.9342
2015	8	156	15	5	0.8076
2014	8	212	20	4	0.8113
2013	9	231	22	4	0.8095
2012	9	209	16	5	0.8468
2011	10	271	50	12	0.6309
2010	8	169	22	9	0.7396
2009	8	118	20	8	0.661
2008	8	213	20	5	0.8122

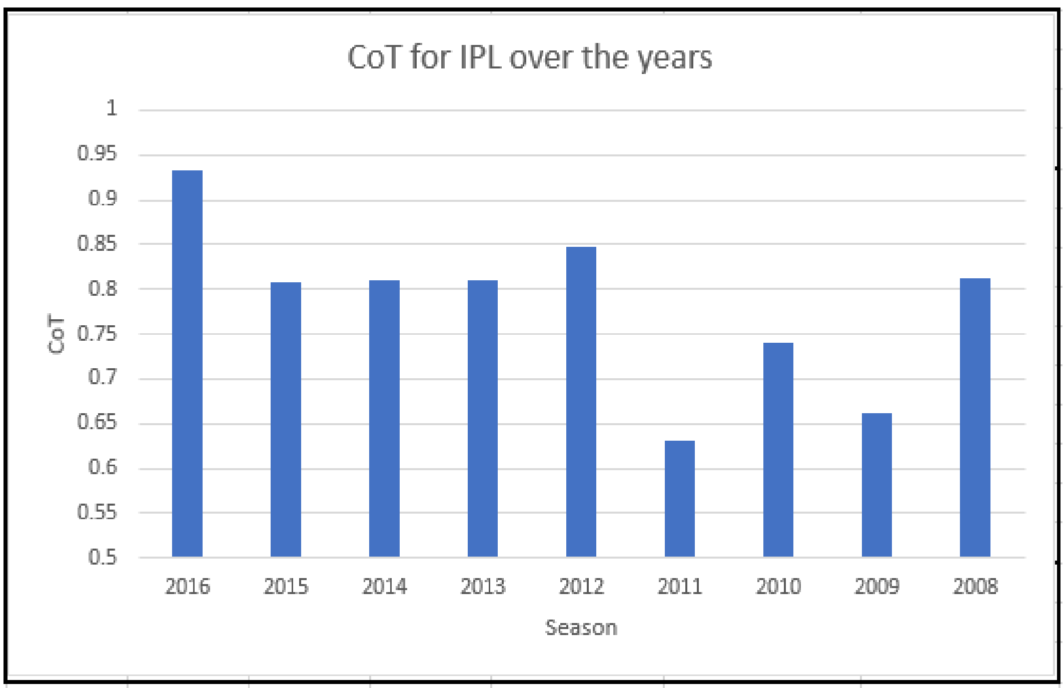


Figure 6.2: CoT spread across IPL seasons

6.3 gives a good visual representation of how our ranking order was generated. Net Run Rate (NRR) is a statistical method used in analysing performance of a team in the sport of cricket. The NRR for a team is the average runs per over that a team scores across the whole tournament, minus the average runs per over that is scored against them across the whole tournament. Higher NRR value for a team indicates a stronger team, who have won matches by huge margin. Unfortunately, in the 2-point ranking system , NRR is considered only while breaking ties, while our ranking approach is based on team strength and its dominance in the tournament.

Table 6.4: Comparison of IPL standings with Gurobi Results for 2010 Season

Rank	Gurobi results	Actual Standings	Match	Won	Lost	Tied	Net RR	Points
1	MI	MI	14	10	4	0	1.084	20
2	DC	DC	14	8	6	0	-0.297	16
3	DD	CSK	14	7	7	0	0.274	14
4	CSK	RCB	14	7	7	0	0.219	14
5	RCB	DD	14	7	7	0	0.021	14
6	KKR	KKR	14	7	7	0	-0.341	14
7	RR	RR	14	6	8	0	-0.514	12
8	KXIP	KXIP	14	4	10	0	-0.478	8

Figure 6.3 represents the IPL 2010 season, where red edges represent the torn edges which are to be removed from the input graph G and they become a part of FAS, while black edges represent DAG.

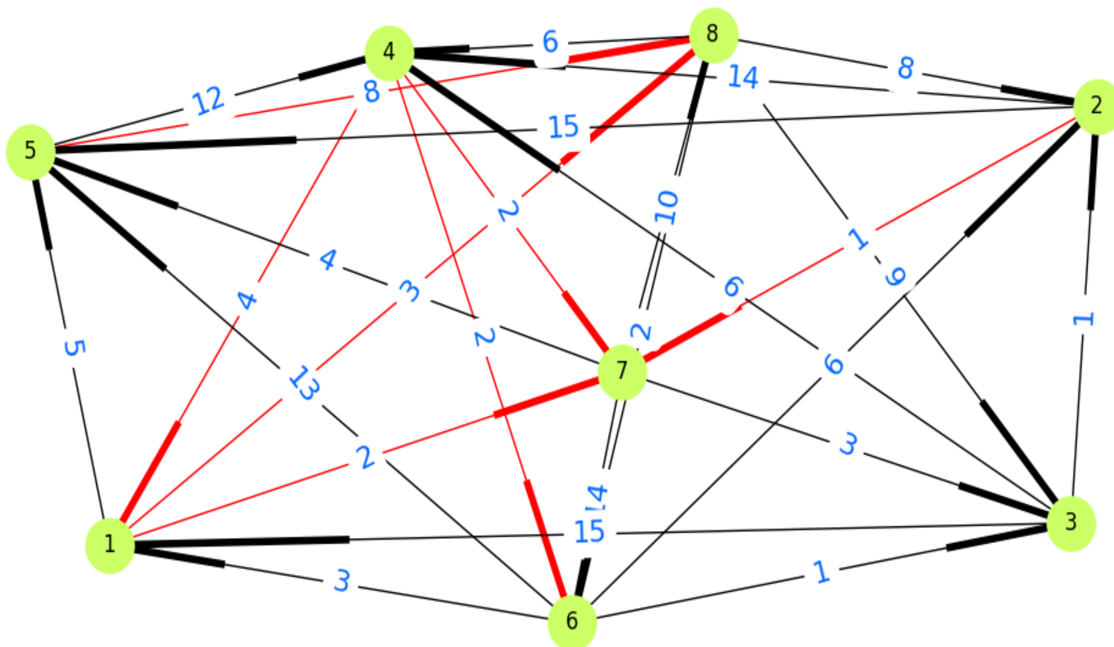


Figure 6.3: IPL 2010 Season

6.3 National Football League

Table 6.5 populates the CoT values calculated for 2007 to 2016 Regular Season of NFL. There are 32 Teams participating in the NFL tournament with each team playing exactly 16 matches. NFL dataset gave us an opportunity to test our algorithm on relatively denser graph.

- Total Nodes N = 32
- Total Edges E $\leq$ 256

  Table 6.6 provides the comparison of the ranking order obtained by our approach with the actual NFL standings. The ranking order as given in the Actual standings column results from the net points as accounted by the total number of wins and losses. The grey part of the table is the point system which is in practice currently and the data is obtained from the following link: `http://www.nfl.com/standings?`

Table 6.5: Computational Results for NFL tournament

Season	Weight of G	Weight of FAS	Torn Edges	COT
2016/17	2470	204	31	0.8381
2015/16	2628	239	35	0.8181
2014/15	3063	287	35	0.8126
2013/14	2716	188	33	0.8615
2012/13	2972	203	36	0.8633
2011/12	3053	203	30	0.867
2010/11	2741	256	36	0.8132
2009/10	3192	192	31	0.8796
2008/09	2871	268	40	0.8133
2007/08	3076	212	31	0.8621

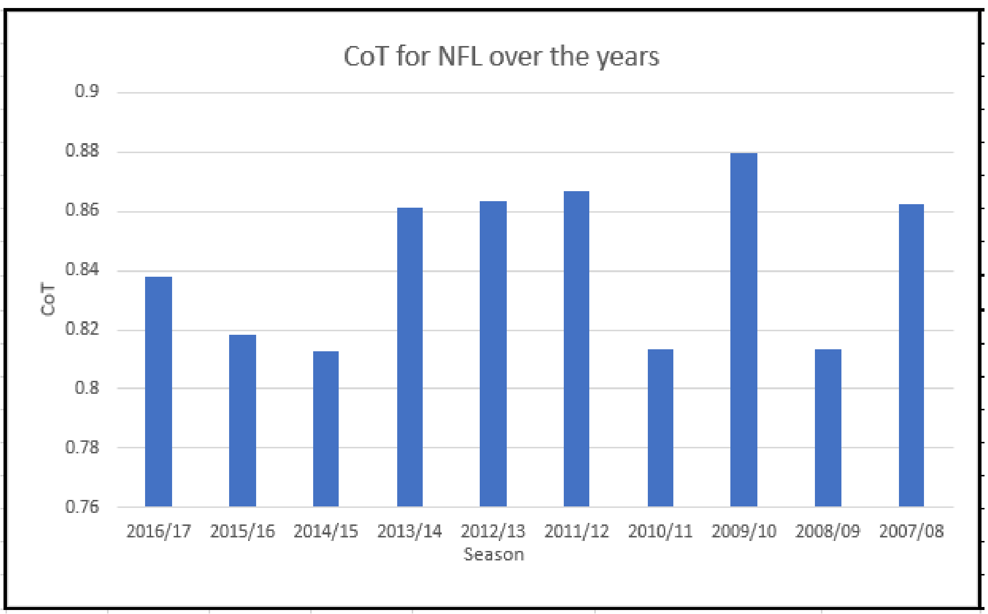


Figure 6.4: CoT spread across NFL seasons

`category=league&season=2015-REG&split=Overall`. In the table 6.6 the net points column is the point differential between PF (Points For) and PA (Points Against) for a team.

Table 6.6: Comparison of NFL standings with Gurobi Results for 2015 Regular Season

Rank	Gurobi Results	Actual Standings	W	L	Net Pts
1	New England Patriots	Carolina Panthers	15	1	192
2	Carolina Panthers	Arizona Cardinals	13	3	176
3	Seattle Seahawks	New England Patriots	12	4	150
4	Pittsburgh Steelers	Seattle Seahawks	10	6	146
5	Arizona Cardinals	Cincinnati Bengals	12	4	140
6	Denver Broncos	Kansas City Chiefs	11	5	118
7	Green Bay Packers	Pittsburgh Steelers	10	6	104
8	Minnesota Vikings	New York Jets	10	6	73
9	Atlanta Falcons	Minnesota Vikings	11	5	63
10	Washington Redskins	Denver Broncos	12	4	59
11	Cincinnati Bengals	Green Bay Packers	10	6	45
12	Chicago Bears	Houston Texans	9	7	26
13	St. Louis Rams	Buffalo Bills	8	8	20
14	Kansas City Chiefs	Washington Redskins	9	7	9
15	Detroit Lions	Atlanta Falcons	8	8	-6
16	Philadelphia Eagles	New York Giants	6	10	-22

17	New York Giants	Oakland Raiders	7	9	-40
18	Buffalo Bills	Detroit Lions	7	9	-42
19	Houston Texans	St. Louis Rams	7	9	-50
20	New Orleans Saints	Philadelphia Eagles	7	9	-53
21	Oakland Raiders	Chicago Bears	6	10	-62
22	San Diego Chargers	New Orleans Saints	7	9	-68
23	New York Jets	Jacksonville Jaguars	5	11	-72
24	Dallas Cowboys	Baltimore Ravens	5	11	-73
25	Jacksonville Jaguars	Indianapolis Colts	8	8	-75
26	Indianapolis Colts	Tampa Bay Buccaneers	6	10	-75
27	Miami Dolphins	San Diego Chargers	4	12	-78
28	Cleveland Browns	Miami Dolphins	6	10	-79
29	Tennessee Titans	Dallas Cowboys	4	12	-99
30	Tampa Bay Buccaneers	Tennessee Titans	3	13	-124
31	San Francisco 49ers	San Francisco 49ers	5	11	-149
32	Baltimore Ravens	Cleveland Browns	3	13	-154

6.4 Discussions

In this section we have compared the results obtained from the Charon and Hudrey approach which is detailed in [10]. The implementation of this approach is available at `http://perso.telecom-paristech.fr/~hudry/tournament/main.c`. The input format is similar to ours, where a weighted tournament graph is represented in a N by N matrix format. N being the total number of nodes in the graph which represent the teams. Their approach is to find a ordering of nodes of minimum value. The program output indicates the best value and the best order.

Results of the Linear order ranking obtained by Charon and Hudrey approach as seen in 6.7 are not just "out of order" when compared with the Actual standings and the team points, but also different ranking order is obtained each time the program is run. Table 6.7 gives a complete comparison of these results with Gurobi Results and Actual league standings. Crystal palace for example, which tops the ranking in Charon and Hudrey approach, have gained only 41 points during the entire season. They are placed at rank 19 and rank 14 in the Gurobi output and Actual standings respectively. On similar lines, Tottenham is placed at rank 2 in Gurobi results and Actual standings while in Charon and Hudrey's approach they are placed at rank 16. Even though, Charon and Hudrey's approach could obtain a Minimum Feedback Arc Set but their Linear ordering of teams is out of order.

Table 6.7: Comparison of Results obtained by Charon and Hudrey Approach vs Gurobi Approach vs Actual Standings for 2016/17 season

Rank	**C & H Results**	**Gurobi results**	**Actual Standings**	**Points**
1	Crystal Palace	Liverpool	Chelsea	93
2	Everton	Tottenham	Tottenham	86
3	West Ham	Arsenal	Man. City	78
4	Bournemouth	Chelsea	Liverpool	76
5	Watford	Everton	Arsenal	75
6	Southampton	Man City	Man United	69
7	Sunderland	Man United	Everton	61
8	West Brom	West Brom	Southampton	46
9	Man. City	Southampton	Bournemouth	46
10	Burnley FC	Bournemouth	West Brom	45
11	Chelsea	Leicester	West Ham	45
12	Arsenal	Stoke	Leicester City	44
13	Stoke City	Watford	Stoke City	44
14	Man United	West Ham	Crystal Palace	41
15	Leicester City	Middlesbrough	Swansea City	41
16	Tottenham	Swansea	Burnley FC	40
17	Hull City	Burnley FC	Watford	40
18	Swansea City	Sunderland	Hull City	34
19	Liverpool	Crystal Palace	Middlesbrough	28
20	Middlesbrough	Hull	Sunderland	24

Chapter 7

Concluding Remarks

We have implemented an integer programming formulation with triangle inequalities for solving the Minimum Feedback arc set problem as proposed in [2],[30],[19]. In [2] an integer programming approach with lazy constraint generation was also proposed to solve the MFAS problem which produced better results as this method scaled better with sparse random graphs. Its approach involves enumerating over all the simple cycles to compute the MFAS. However with the dataset we had, enumerating over all the simple cycles proved to be intractable. Hence we gave up on this approach and used state of the art solver like Gurobi to implement the approach detailed in section 5.2. Our method is tailored for tournaments of size n $\leq$ 32. Trying to compute MFAS on highly structured sparse graphs by our approach is intractable as number of constraints becomes too high. All the source code is available at the following link : `https://github.com/vsh15/MFAS`. With the right input format the code can be easily extended to compute CoT and ranking order for other tournaments such as Spanish Football League (La Liga), German Football League (Bundesliga), Basketball (NBA), Ice Hockey (NHL) and many more.

7.1 Contribution 1

The COT metric which we have introduced in this paper will help in characterizing balance of tournament. In other words, COT could be used to compute the competitiveness of a tournament. A tournament with COT value close to 0 would mean that the participating teams are evenly matched, while a value closer to 1 would mean that the tournament is highly transitive and we could clearly observe the distinction between the strong teams and the weak teams as number of "upsets" are fairly low. We also compared the CoT metric from season to season to see if the balance within a league was changing over time.

7.2 Contribution 2

We generate a linear ordering of teams from the DAG obtained by solving the MFAS problem for a tournament. The ranking generated by our method takes into consideration the performance of the teams and hence the teams are ranked based on their relative strengths. This ranking system proves to be fairer than the 3-point and the 2-point ranking system incorporated today in EPL and IPL. Unlike the Charon and Hudrey approach where the ranking order shifts a little on each run, our approach generates the same exact linear order on every run of a tournament graph.

Chapter 8

Future Work

8.1 Correlating Revenue with Dominance in DAG

The CoT could be correlated with a team's revenue. The project could be framed to seek the answers of the questions like is it true that leagues with revenue sharing (e.g. the NFL) are more balanced than leagues without it? How much is a team's revenue (i.e. wealth) correlated with dominance in the DAG, especially over many seasons? Some would argue it is positively correlated, as richer teams can afford better players. However, it has also been argued that wealth might be negatively correlated with rank, as a team with a local monopoly might be able to "coast" - generating a lot of revenue without having to spend money on better players and resources.

8.2 Correlating CoT with popularity of the sports tournament

Sports league have beginning to expand on the global fan base with the advent of digital technology and high speed internet services available on the streamable devices. Multiple studies have been done in trying to rank a sports league based on its popularity. Some of the criteria to name a few are:

1. TV viewership numbers
2. Average athlete salary
3. Prominence in sports headlines and media outlets

Sports Tournaments like NBA, La Liga, English Premiere League, ATP have world-wide popularity and it would be an interesting study to see the correlation between COT and popularity of the tournament. It can be argued that a more balanced tournament having a lower COT are popular as majority of the teams are equally competitive and viewers can expect a closely matched contest. On the other hand, Tournaments in which a minority of the teams dominate the entire tournament may not have such high viewership with the exception of finals.

References

[1] ACHTERBERG, T. Scip: solving constraint integer programs. *Mathematical Programming Computation 1*, 1 (Jul 2009), 1–41.

[2] ALI BAHAREV, HERMANN SCHICHL, A. N. T. A. An exact method for minimum feedback arc set problem. 1071 – 1078.

[3] ARORA, S., FRIEZE, A., AND KAPLAN, H. A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. *Mathematical Programming 92*, 1 (Mar 2002), 1–36.

[4] AUSTRIN, P., MANOKARAN, R., AND WENNER, C. *On the NP-Hardness of Approximating Ordering Constraint Satisfaction Problems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 26–41.

[5] BAK, P., TANG, C., AND WIESENFELD, K. Self-organized criticality: An explanation of the 1/f noise. *Phys. Rev. Lett. 59* (Jul 1987), 381–384.

[6] BARKLEY, R., AND MOTARD, R. Decomposition of nets. *The Chemical Engineering Journal 3* (1972), 265 – 275. An International Journal of Research and Development.

[7] BERGER, B., AND SHOR, P. W. Approximation alogorithms for the maximum acyclic subgraph problem. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 1990), SODA '90, Society for Industrial and Applied Mathematics, pp. 236–243.

[8] BESSY, S., FOMIN, F. V., GASPERS, S., PAUL, C., PEREZ, A., SAURABH, S., AND THOMASSÃL', S. Kernels for feedback arc set in tournaments. *Journal of Computer and System Sciences 77*, 6 (2011), 1071 – 1078.

[9] CHARBIT, P., THOMASSÉ, S., AND YEO, A. The minimum feedback arc set problem is np-hard for tournaments. *Comb. Probab. Comput. 16*, 1 (Jan. 2007), 1–4.

[10] CHARON, I., AND HUDRY, O. Noising methods for a clique partitioning problem. *Discrete Applied Mathematics 154*, 5 (2006), 754 – 769. IV ALIO/EURO Workshop on Applied Combinatorial Optimization.

[11] CHARON, I., AND HUDRY, O. An updated survey on the linear ordering problem for weighted or unweighted tournaments. *Annals of Operations Research 175*, 1 (2010), 107–158.

[12] CHEN, J., LIU, Y., LU, S., O'SULLIVAN, B., AND RAZGON, I. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM 55*, 5 (Nov. 2008), 21:1–21:19.

[13] CHUNG, F. R. K., AND HWANG, F. K. Do stronger players win more knockout tournaments? *Journal of the American Statistical Association 73*, 363 (1978), 593–596.

[14] CHVATAL, V. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research 4*, 3 (1979), 233–235.

[15] D. BRUNK, H. Maximum likelihood estimates of monotone parameters.

[16] DAVIDSON, R. R., AND SOLOMON, D. L. A bayesian approach to paired comparison experimentation. *Biometrika 60*, 3 (1973), 477–487.

[17] EADES, P., LIN, X., AND SMYTH, W. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters 47*, 6 (1993), 319 – 323.

[18] FLIER, H. F. *Optimization of Railway Operations. Algorithms, Complexity, and Models*. PhD thesis, ETH Zurich, 2011. Diss., EidgenÃűssische Technische Hochschule ETH ZÃijrich, Nr. 20115, 2011.

[19] GRÖTSCHEL, M., JÜNGER, M., AND REINELT, G. A cutting plane algorithm for the linear ordering problem. *Oper. Res. 32*, 6 (Dec. 1984), 1195–1220.

[20] GUO, J., GRAMM, J., HÜFFNER, F., NIEDERMEIER, R., AND WERNICKE, S. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences 72*, 8 (2006), 1386–1396.

[21] KARP, R. M. *Reducibility among Combinatorial Problems*. Springer US, Boston, MA, 1972, pp. 85–103.

[22] LAGUNA, M., MARTI, R., AND CAMPOS, V. Intensification and diversification with elite tabu search solutions for the linear ordering problem. *Comput. Oper. Res. 26*, 12 (Oct. 1999), 1217–1230.

[23] LE, M. H., AND PHAN, T. H. D. Strict partitions and discrete dynamical systems. *Theoretical Computer Science 389*, 1 (2007), 82 – 90.

[24] MITCHELL, J. E., AND BORCHERS, B. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In *High performance optimization*. Springer US, 2000, pp. 349–366.

[25] MORRISON, H. W. Testable conditions for triads of paired comparison choices. *Psychometrika 28*, 4 (Dec 1963), 369–390.

[26] RAMAN, V., AND SAURABH, S. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science 351*, 3 (2006), 446 – 458. Parameterized and Exact Computation.

[27] SIMPSON, M., SRINIVASAN, V., AND THOMO, A. Efficient computation of feedback arc set at web-scale. *PVLDB 10* (2016), 133–144.

[28] SLATER, P. Inconsistencies in a schedule of paired comparisons. *Biometrika 48*, 3-4 (1961), 303–312.

[29] THOMPSON, W. A., AND REMAGE, R. Rankings from paired comparisons. *Ann. Math. Statist. 35*, 2 (06 1964), 739–747.

[30] TIWISINA, J., AND KÃIJLPMANN, P. Probabilistic Transitivity in Sports, 2014.

[31] WÄCHTER, A., AND BIEGLER, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106*, 1 (Mar 2006), 25–57.