An Exploration of Grammatical Encodings to model Six Nations Rugby Match Outcomes

Michael O'Neill, Anthony Brabazon, David Fagan
Natural Computing Research & Applications Group
UCD Business
University College Dublin
Ireland
Email: m.oneill@ucd.ie, anthony.brabazon@ucd.ie, david.fagan@ucd.ie

***Abstract*—We explore the application of grammar-based Genetic Programming, specifically Grammatical Evolution, to the problem of modeling the outcome of Six Nations Rugby matches. A series of grammars are developed in attempts to generate different forms of predictive rules, which might be useful in pre-match and mid-match scenarios. A number of interesting models are generated and their utility discussed.**

I. Introduction & Related Work

While the collection and use of quantitative data in sports has a long history, dating back at least to the collection of basic statistics such as hits and pitches in 19th century baseball in the United States, the last fifty years has seen an ever-expanding level of sophistication in these applications [1].

With the commercialisation of sporting activity, the motivation and payoffs for the collection of sporting data has increased dramatically, leading to the development of an entire industry devoted to 'sports analytics'. The idea of sports analytics was popularised in the book MoneyBall, written by Michael Lewis in 2003 [2], which described the use of analytics by the Oakland Athletics baseball team. The book was subsequently made into a film starring Brad Pitt in 2011. The ideas have been further explored by others including Sawchik [3].

In parallel to the commercial development of the field of sports analytics, new research journals have emerged ranging from sports performance journals focussing on aspects of human and animal (for example, horse) performance, to specialist sports analytics journals such as the Journal of Quantitative Analysis in Sports (JQAS), an official journal of the American Statistical Association, and the Journal of Sports Analytics. International conferences in the area of sports analytics have also become common with the best known being the annual MIT Sloan Sports Analytics Conference which commenced in 2006. While the precise definition of the term sports analytics is elusive, the coverage of JQAS includes 'game outcome models, measurement and evaluation of player performance, tournament structure, analysis of rules and adjudication, within-game strategy, analysis of sporting technologies, and player and team ranking methods' [4].

Commercial software service providers such as SAS now offer an array of specialist analytical tools for sporting organisations. These tools cover areas such as monitoring fan insight & loyalty, optimisation of ticket pricing, and player / team analytics. In the context of player and game management, analytical tools can be used for a variety of applications including the analysis of historical or real-time game information in order to obtain competitive intelligence, to anticipate and avoid player safety / injury issues, and to assist with player recruitment.

Typical data inputs include in-game and player biometric data, demographic and CRM data on fans, and information from social media websites. The quantity of data available has increased significantly as players wear biometric sensors routinely during training and competitive games in many professional sports, motion capture technology can track everything that occurs on the field of play, and data from social media provides insight into fan sentiment. A variety of commercial data providers such as Opta[1], who collect data on UK Premiership Games and then offer this data as a service to clients for a fee, have lowered the barriers to entry for new 'adopters' of sports analytics. Information from commercial providers is also used by bookmakers in pricing various bets such as game outcome or the score differential arising in a game. As the volume of game data collected has increased, there has been a corresponding increase in the range of betting products offered to the public.

An interesting aspect of the use of sports analytics for game / player management is that domain knowledge is clearly important. Each sport has its own rules, training methods and differing athletic requirements. Even within an individual sport, the game environment is dynamic as rules change over time, and game strategy formulation and athlete preparation bears parallel with an evolutionary arms-race as successful innovations are either copied or countered. Hence, we have a dynamic environment, where domain knowledge is important, and where the range of available data is increasing. This suggests that data mining methodologies which can incorporate domain knowledge, such as grammar-based GP, are likely to be of considerable utility.

In this paper, we undertake a pilot study which illustrates the application of one form of grammar-based GP, namely Grammatical Evolution, for the purposes of modeling the outcomes

[1] http://http://www.optasports.com/

of 6 Nations Rugby Championship matches. In particular, we explore the utility of a number of different grammars for this task and examine the strengths and weaknesses of the resulting models. While there have been a couple of papers that have applied GP to sports analytics applications [5], [6]), we are unaware of any previous applications of a grammar-based GP approach in Sports Analytics, and this a novel application to the domain of Rugby Union.

In the Section II, we provide a short introduction to the game of rugby union and the dataset used in this study. Section III introduces the grammatical encodings and experimental settings used in the study. The results are provided in Section IV followed by conclusions and some suggestions for future work in Section V.

II. Rugby Union

Rugby union is a team sport in which each team has 15 players, split between 8 'forwards' and 7 'backs'. The game is played on a rectangular field and is based on running with an oval ball. In play, each team defends their 'try line' (one end of the rectangular field) and if their opponents touch the ball down behind this line, they are awarded a 'try' (worth five points). If a team commits an infringement of the rules, a penalty may be awarded against it, and in this case, the non-penalised team can attempt to kick the ball between two upright goalposts which are located on each try line. If the kick is successful, three points are awarded to the kicking team. On the scoring of each try, the attacking team is also awarded a 'conversion attempt' whereby they can score an additional two points by kicking the ball between the goalposts. While the ball may be kicked forward, hand passes can only be made backwards. The team which scores the higher number of points over two, forty-minute periods of play wins the game.

According to myth, the game was invented in 1823 when a pupil of Rugby School (a private school dating from 1567 in the town of Rugby in Warwickshire) called William Webb Ellis picked up the ball during a game of football and ran with it. Although the veracity of this story is debated, the game came to prominence in England in the early nineteenth century with the first set of codified rules being written in 1845 by pupils of Rugby School.

By the 1880s, international games were being played between the 'home nations' (Ireland, England, Wales, and Scotland) and the first championship between these countries took place in 1883. In subsequent years, the game spread to many other countries, and the game is now played in over 120 countries worldwide, including Australia, New Zealand, South Africa, Italy, Argentina, USA, Canada, Japan and Russia to name but a few. The major annual international competitions include the modern incarnation of the Home Nations Championship, the 6 Nations Championship (see below), and the Rugby Championship (played annually between New Zealand, Australia, South Africa and Argentina). Every four years there is a 'Rugby World Cup' in which the top 20 countries participate.

Up to 1995, the game was an amateur sport but since then, a professional tier of the game has developed which encompasses professional club teams in most of the top ten countries. Over the past twenty years, top-level rugby sport has become big business. Its scale can be appreciated by considering that some 2.5 million spectators attended the recent (2015) Rugby world cup games. The only other global sporting events in history that have generated larger numbers of spectators have been Soccer world cups [7].

A. Six Nations Rugby Dataset

The Six Nations Championship is an annual international rugby union tournament between Ireland, Wales, Scotland, France, Italy, and England[2]. This tournament has existed in some format since 1883, and was the first international rugby tournament. The current iteration of the tournament has been is existence since 2000, when Italy joined. During the tournament each country will play the other five countries once. The venue for these games alternate between home and away fixtures on an annual basis, and the ordering of fixtures is varied each year. The country with the most points at the end of the five fixtures wins. In the event of a tie the winner is decided by the following tie breakers, the country with the higher points differential (points for minus points against) wins. In the event this also results in a tie the country with the highest number of tries scored is awarded the trophy. In the event this also results in a tie the championship is shared between the tied countries.

B. Dataset

The dataset for this study was obtained from Statsguru, ESPN's Rugby statistics database[3] using the Team Records search utility[4]. We retrieved records covering the period of the 21st Century (2001-2015) for the Trophy "Five/Six Nations" with the View Format set to "Match list". The dataset contains 12 variables as displayed in Table I.

TABLE I
ESPN Rugby StatGuru Dataset Variables

Variable	Description
`Team`	Name of the `Team`
`Result`	Result of the match (`win`, `lost`, `draw`)
`For`	Number of points For the `Team`.
`Aga`	Number of points scored against the `Team`.
`Diff`	Points difference (i.e., `Diff=For-Aga`)
`Tries`	Number of Tries scored by the `Team` (worth 5 points).
`Conv`	Number of conversions scored by the `Team` (worth 2 points).
`Pens`	Number of penalties scored by the `Team` (worth 3 points).
`Drop`	Number of drop goals scored by the `Team` (worth 3 points).
`Opposition`	Opposition team name.
`Ground`	Ground in which the match was played.
`MatchDate`	Date on which the match was played.

Note that there are two entries for every match, so that we have the number of tries, conversions, penalties and drop goals for both sides in a match. In this study we exclude three variables

[2]http://www.rbs6nations.com/

[3]http://stats.espnscrum.com/statsguru/rugby/stats/.

[4]http://stats.espnscrum.com/statsguru/rugby/stats/index.html?class=1;type=team

from the generated models. Diff is excluded as it is effectively a proxy for Result. As there are two entries for each match (one for each team that plays) we exclude Opposition. We do not consider the temporal nature of the dataset, and the possibility that the performance of teams may ebb and flow over the years depending on factors such as team composition and motivation, as such, MatchDate is also excluded.

The dataset is divided into training and test sets. The training set is comprised of all matches from 2001 up to and including the 2013 tournament. The test set is the remaining data for the last two tournaments which took place during 2014 and 2015. Summary statistics of the numerical features are provided in Tables II & III for the training and test datasets respectively. For the 390 matches in the training set, 10 resulted in a draw, and for the 60 test set matches every game resulted in a winner.

III. Grammatical Encodings for Match Prediction

The objective of this study is to evolve decision tree-like models in Python using grammar-based GP [8]. Specifically, Grammatical Evolution is adopted, which has an extensive literature including method development and application (e.g., [9], [10], [11], [12]). To this end, the following approach is taken to encode all the required variables in the grammar, whilst ensuring that solutions generated are in a decision tree-like format. This produces a classification-type problem and we note that there is a substantial exant literature on classification in GP (e.g., [13], [14], [15], [16], [17], [18], [19]).

A particular strength of grammatical evolution is that the modeller can easily specify, via the design of the grammar, the structure of the model (classifier in this case) which is to be produced. In this section we describe the various grammars used and the resulting classifers which they can output.

The grammar in Fig. 1 generates a fixed form solution (in Python) based on an if-elif-else conditional statement. Working from the start symbol (<code>), the conditional statement is embedded in a for loop, which serves to execute the conditional statements over each element of the training set.

The second line of the grammar, encodes the if-elif-else statement which is rooted in the non-terminal <line>. The return values in this case are themselves evolved to take on any of the values won, lost, or draw. The general form of a solution can be seen in Fig. 2.

The non-terminal <cond> specifies that a condition can be generated from either an expression utilising problem-specific variables of type string (<strexpr>) or from an expression generated from problem-specific numerical variables (integers) rooted in the non-terminal <numexpr>.

Conditional statements based on problem-specific strings include the six nations participating in the tournament (i.e., Ireland, England, Scotland, Wales, France and Italy) and the stadia in which the matches are played (i.e., Landsdowne Road, Twickenham, Murrayfield, Millenium Stadium, Stade de France and Rome). Each of these string values is provided as a constant in the grammar through the non-terminals <ground> and <team>. Conditional statements using these string constants are then formed by comparing team and ground values from records in the dataset to these constants. For example, team[i]==<team> tests whether or not a constant <team> is equivalent to the team name in record i of the training dataset. Multiple string conditionals can be combined into a more complex boolean expression using either and's and or's as specified in the rule <strexpr><boolop><strexpr>.

Similarly, numerical conditional statements can be complex boolean expressions as specified by the rule <numexpr><boolop><numexpr>. In this case, the statements can utilise predefined constants (from the rule <const>) and problem-specific values from the variables f0r,aga,tries,conv,pens and drop, which encode the number of points scored by the team, the number of points scored against the team, the number of tries, conversions, penalities and drop goals scored by the team, respectively. The relational operators less-than-or-equal-to and greater-than are provided through the non-terminal <relop>.

The fitness of a solution is determined by simply counting the number of training cases mis-classified, and dividing by the total number of training cases (as we are minimising fitness). The Python code used to evaluate a solution is provided in Fig. 3.

```
<code> ::= for i in range(0, TOTAL): <lines>
<lines> ::= if <cond> : guess[i] = <result>
            elif <cond> : guess[i] = <result>
            else : guess[i] = <result>
<cond> ::= <strexpr> | <numexpr>
<strexpr> ::= team[i] == <team>
            | ground[i] == <ground>
            | <strexpr> <boolop> <strexpr>
<numexpr> ::= <numvar> <relop> <numvar>
            | <const> <relop> <numvar>
            | <numexpr> <boolop> <numexpr>
<relop> ::= <= | >
<boolop> ::= and | or
<biop> ::= + | *
<const> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
          | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19
          | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30
<result> ::= "won" | "lost" | "draw"
<numvar> ::= f0r[i] | aga[i] | tries[i] | conv[i] | pens[i] | drop[i]
<ground> ::= "Lansdowne Road"  | "Twickenham" | "Murrayfield"
           | "Millennium Stadium" | "Stade de France" | "Rome"
<team> ::= "Ireland" | "England" | "Scotland" | "Wales" | "France" | "Italy"
```

Fig. 1. A grammar to generate rules to predict the outcome of a Six Nations match.

```
# Each evolved individual iterates over the training set
# TOTAL is the number of training examples

for i in range(0, TOTAL):
        if <cond> :
                guess[i] = <result>
        elif <cond> :
                guess[i] = <result>
        else :
                guess[i] = <result>
```

Fig. 2. The general form of a solution generated by the grammar in Fig. 1. The two conditional statements (<cond>'s) and match outcomes (<result>'s) are evolved.

TABLE II
TRAINING DATASET STATISTICS COVERING 390 MATCHES PLAYED BETWEEN 2001 AND 2013

Statistic	For	Aga	Tries	Conv	Pens	Drop
N	390	390	390	390	390	390
Mean	21.426	21.426	1.992	1.459	2.649	0.200
St.Dev.	11.589	11.589	1.770	1.470	1.596	0.482
Min	0	0	0	0	0	0
Median	19.5	19.5	2	1	2	0
Max	80	80	10	9	7	3

TABLE III
TEST DATASET STATISTICS COVERING 60 MATCHES PLAYED BETWEEN 2014 AND 2015 SIX NATIONS TOURNAMENTS

Statistic	For	Aga	Tries	Conv	Pens	Drop
N	60	60	60	60	60	60
Mean	21.050	21.050	2.050	1.550	2.467	0.100
St.Dev.	13.445	13.445	1.987	1.641	1.546	0.303
Min	0	0	0	0	0	0
Median	20	20	2	1	2	0
Max	61	61	8	7	6	1

```
# cmd is a list containing the evolved solution code
# TOTAL is the number of training cases

def calculateSixNationsDifference(cmd):
        guess = [ ("draw") for ii in range(0, TOTAL)]
        exec cmd
        difference = 0.0
        for ii in range(0, TOTAL):
                if guess[ii]=="won":
                        if result[ii]!="won":
                                difference+=1
                        else:
                                difference+=0
                elif guess[ii]=="lost":
                        if result[ii]!="lost":
                                difference+=1
                        else:
                                difference+=0
                else:
                        if result[ii]!="draw":
                                difference+=1
                        else:
                                difference+=0
         return difference / TOTAL
```

Fig. 3. Python code of the objective function.

A. Experimental Settings

As indicated earlier, the dataset is divided into training and test sets. The training set is comprised of all matches from 2001 up to and including the 2013 tournament. The test set is the remaining data for the last two tournaments which took place during 2014 and 2015. The evolutionary parameter settings used for our experiments are shown in Table IV.

IV. RESULTS AND DISCUSSION

In this section we outline our results, discussing in turn the outputs from the initial grammar as well as those from the reduced variables and restricted conditions grammars. 30 independent runs on the training set are performed for each grammatical encoding.

A. Initial Grammar

The first grammar explored is presented earlier in the paper (Fig 1) and allows the return value of each component of `if-elif-else` to be evolved.

TABLE IV
EVOLUTIONARY PARAMETER SETTINGS.

Parameter	Value
Population size	1000
Generations	100
Selection	Fair Tournament
Tournament size	0.01% of the population size
Replacement	Generational with Elitism (0.1% of population size)
Crossover probability	0.9
Mutation probability	0.01
Initialisation	ramped-half-and-half
Initial Max depth	15
Raw fitness	sum of misclassifications over the training set
Trials per treatment	30 independent runs for each value

Perfect predictive accuracy is achieved using the above grammar as highlighted in Fig. 4. In plain english, if the score against the home team is greater than their own score they have lost. Else if their score is greater than the score against them they have won, else the outcome is a draw. While this model has perfect predictive accuracy, it is stating the obvious, and has no practical use in prediction of match outcome.

```
Generation 7      Fitness 0

for i in range(0, TOTAL):
        if aga[i] > f0r[i] :
                guess[i] = "lost"
        elif f0r[i] > aga[i] :
                guess[i] = "won"
        else : guess[i] = "draw"
```

Fig. 4. Example evolved solution for the initial grammar (Fig. 1).

B. Reduced Variables Grammar

In order to predict match outcomes prior to match start we remove match score statistics from the grammar, so models can now only be constructed using teams and stadia (see Fig. 5). Inferior predictive training accuracy is produced of approximately 72%, with the dominant models including rules which state that Italy and Scotland will always lose, everyone else wins but with certain caveats. For example, in the generation 50 sample individual from a typical run (Fig. 6)

we see that France and Wales tend to lose at Twickenham (the home ground of England), England and Wales tend to lose at Landsdowne Road (the home ground of Ireland), England lose in the Millenium Stadium (Wales), and Wales, England and Ireland lose in the Stade de France.

```
<code> ::= for i in range(0, TOTAL): <lines>
<lines> ::= if <cond> : guess[i] = <result>
            elif <cond> : guess[i] = <result>
            else : guess[i] = <result>
<cond> ::= <strexpr>
<strexpr> ::= team[i] == <team>
            | ground[i] == <ground>
            | <strexpr> <boolop> <strexpr>
<boolop> ::= and | or
<result> ::= "won" | "lost" | "draw"
<ground> ::= "Lansdowne Road"  | "Twickenham" | "Murrayfield"
           | "Millennium Stadium" | "Stade de France" | "Rome"
<team> ::= "Ireland" | "England" | "Scotland"
         | "Wales" | "France" | "Italy"
```

Fig. 5. A grammar to produce models which can be used prior to a match (i.e., numerical variables on match score statistics have been removed).

```
Generation 10      Fitness 0.312821

for i in range(0, TOTAL):
        if team[i] == "Scotland" or team[i] == "Italy" :
                guess[i] = "lost"
        elif ground[i] == "Twickenham" and team[i] == "France" :
                guess[i] = "lost"
        else :
                guess[i] = "won"
```

```
Generation 50     Fitness 0.276923

for i in range(0, TOTAL):
        if team[i] == "Scotland" or team[i] == "Italy" :
                guess[i] = "lost"
        elif
            team[i] == "France" and ground[i] == "Twickenham" or
            team[i] == "Wales" and ground[i] == "Twickenham" or
            team[i] == "Wales" and ground[i] == "Lansdowne Road" or
            team[i] == "England" and ground[i] == "Lansdowne Road" or
            team[i] == "England" and ground[i] == "Millennium Stadium" or
            ground[i] == "Stade de France" and team[i] == "Wales" or
            ground[i] == "Stade de France" and team[i] == "England" or
            ground[i] == "Stade de France" and team[i] == "Ireland" :
                guess[i] = "lost"
        else :
                guess[i] = "won"
```

Fig. 6. Example evolved solutions for the reduced variables grammar (Fig. 5). The generation 50 sample individual has been simplified by reordering and removing redundant components from the `elif` condition.

C. Restricted Conditions Grammar

With this grammar we wish to find a more accurate predictive model which might be used as a game progresses by generating models which include match statistics for the home team. These include the score for the home team, score against them, the number of tries, conversions, penalties and drop goals scored. The grammar adopted is presented in Fig. 7, and examples of evolved solutions are provided in Fig. 8 achieving a training accuracy of approximately 96%.

```
<code> ::= for i in range(0, TOTAL): <lines>
<lines> ::= if <cond> : guess[i] = <result>
            elif <cond> : guess[i] = <result>
            else : guess[i] = <result>
<cond> ::= <strexpr> | <numexpr>
<strexpr> ::= team[i] == <team>
            | ground[i] == <ground>
            | <strexpr> <boolop> <strexpr>
<numexpr> ::= <const> <relop> <numvar>
            | <numexpr> <boolop> <numexpr>
<relop> ::= <= | >
<boolop> ::= and |or
<biop> ::= + | *
<const> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
          | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19
          | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30
<result> ::= "won" | "lost" | "draw"
<numvar> ::= f0r[i] | aga[i] | tries[i] | conv[i] | pens[i] | drop[i]
<ground> ::= "Lansdowne Road"  | "Twickenham" | "Murrayfield"
           | "Millennium Stadium" | "Stade de France" | "Rome"
<team> ::= "Ireland" | "England" | "Scotland"
         | "Wales" | "France" | "Italy"
```

Fig. 7. A grammar to restrict conditions to take the form of decision tree-like linear decision surfaces. The "restriction" over earlier grammars is that conditions cannot contain comparisons between a problem variable and another problem variable, instead problem variables must be compared to elements of `<const>`.

```
Generation 79     Fitness 0.0358974
for i in range(0, TOTAL):
        if
           23 <= aga[i] and 25 > f0r[i] or
           30 <= aga[i] and 1 > drop[i] or
           19 > f0r[i]  and 28 > tries[i] and 18 <= aga[i] or
           12 <= aga[i] and 14 > f0r[i] and 25 > tries[i] :
                guess[i] = "lost"

        elif
        # this condition is effectively redundant given return value of else
        # and has therefore been removed for simplicity of exposition
                guess[i] = "won"

        else :
                guess[i] = "won"
```

```
73      0.0358974
for i in range(0, TOTAL):
        if 20 <= conv[i] :
                guess[i] = "lost"
        elif
           14 > aga[i] and 12 <= f0r[i] or
           17 <= f0r[i] and 4 <= conv[i] or
           17 <= f0r[i] and 18 <= pens[i] and 10 <= drop[i] or
           19 > aga[i] and 18 <= f0r[i] or
           16 <= tries[i] or
           29 <= f0r[i] and 17 <= aga[i] and 26 > aga[i] or
           16 <= conv[i] or
           10 > aga[i] or
           5 <= tries[i] and 20 > pens[i] or
           19 > aga[i] and 18 <= f0r[i] or
           18 <= drop[i] or
           22 <= f0r[i] and 25 > aga[i] :
                guess[i] = "won"
        else :
                guess[i] = "lost"
```

Fig. 8. Example evolved solutions for the restricted conditions grammar.

D. Including Variable Statistics in the Grammar

We also extend the restricted conditions grammar (Fig. 7) to include statistical analysis of the problem variables through the incorporation of the <stats> non-terminal as outlined in Fig. 9. A <const> can now become either an <int> or <stats>. <int> encodes the integer constants from the earlier grammars, and <stats> encodes ten score statistics. Specifically, the mean and max values of f0r,aga,tries,conv,pens and drop in the training dataset are provided as constants. An example solution is given in Fig. 10 also exhibiting a training classification accuracy of 96%.

```
<code> ::= for i in range(0, TOTAL): <lines>
<lines> ::= if <cond> : guess[i] = <result>
           elif <cond> : guess[i] = <result>
           else : guess[i] = <result>
<cond> ::= <strexpr> | <numexpr>
<strexpr> ::= team[i] == <team>
            | ground[i] == <ground>
            | <strexpr> <boolop> <strexpr>
<numexpr> ::= <const> <relop> <numvar>
            | <numexpr> <boolop> <numexpr>
<relop> ::= <= | >
<boolop> ::= and |or
<biop> ::= + | *
<const> ::= <int> | <stats>
<int> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
          | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19
          | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30
<stats> ::= <__mean_f0r_aga> | <__max_f0r_aga>
          | <__mean_tries> | <__max_tries>
          | <__mean_conv> | <__max_conv>
          | <__mean_pens> | <__max_pens>
          | <__mean_drop> | <__max_drop>
<__mean_f0r_aga> ::= 21.43
<__maxf_0r_aga> ::= 80
<__mean_tries> ::= 1.99
<__max_tries> ::= 10
<__mean_conv> ::= 1.46
<__max_conv> ::= 9
<__mean_pens> ::= 2.65
<__max_pens> ::= 7
<__mean_drop> ::= 0.2
<__max_drop> ::= 3
<result> ::= "won" | "lost" | "draw"
<numvar> ::= f0r[i] | aga[i] | tries[i] | conv[i] | pens[i] | drop[i]
<ground> ::= "Lansdowne Road"  | "Twickenham" | "Murrayfield"
           | "Millennium Stadium" | "Stade de France" | "Rome"
<team> ::= "Ireland" | "England" | "Scotland"
         | "Wales" | "France" | "Italy"
```

Fig. 9. An extension of the restricted conditions grammar presented in the last subsection (Fig. 7) to include statistics calculated on match scores over the training set.

Fixing return values: We restrict the grammar outlined in Fig. 9 to return a fixed set of values, namely "won", "lost" and "draw" respectively for the if,elif,else conditions as per the non-terminal <lines> in Fig. 1. The best model found is presented in Fig. 11 and again achieves training accuracy of 96%.

E. Summary

In summary, we can capture the relative performance of each grammar encoding using fitness plots calculated over 30 replications of each grammar in Fig. 12 (top). We can see three clear clusters of performance. The initial grammar consistently produces a global optimum model, but as discussed earlier this has no predictive value as it simply captures the facts that if a team scores more than their opponents they win, if scores for both teams are the same it is a draw, otherwise the team lost. The model with weakest predictive accuracy (approximately 70%) is produced by the "reduced variables" grammar (Fig. 5), which excluded numerical variables (match score statistics). These models had the advantage that they could be used prior to a match to predict the winning team. The best of these models found that Italy and Scotland generally lost while the remaining teams generally won given a set of caveats based on what grounds they were playing in.

The third cluster of models based on predictive performance, utilised numerical variables to produce in-game predictive models, and enforced decision tree-style models with boolean composition of linear decision surfaces at each node. Some of these models allowed the use of variable statistics, but it was found that these did not provide a clear performance advantage. However, superior predictive accuracy of approxi-

```
Generation 70     Fitness 0.0358974
for i in range(0, TOTAL):
        if
          13 > aga[i] and 11 <= f0r[i] or
          3 <= pens[i] and 3 <= tries[i] and 21.43 <= aga[i] or
          5 <= tries[i] or
          1 <= drop[i] and 1.99 <= tries[i] and 21.43 > aga[i] or
          23 > aga[i] and 19 <= f0r[i] :
                guess[i] = "won"

       elif 2.65 > drop[i] :
                guess[i] = "lost"

       else :
                guess[i] = "draw"
```

Fig. 10. Example evolved solutions for the grammar enriched with variable statistics calculated on the training data (see Fig. 9). Interestingly this example individual exploits the constants 21.43 and 1.99 in their "correct" contexts, as the former is the mean score for/against the home team and the latter (1.99) is the mean number of tries scored.

```
Generation 84     Fitness 0.0384615
for i in range(0, TOTAL):
        if
          28 <= f0r[i] or
          10 <= f0r[i] and 13 > aga[i] or
          19 <= f0r[i] and 23 > aga[i] :
                guess[i] = "won"

        elif
          13 > f0r[i] and 80 > aga[i] and 23 > conv[i] and 14 <= drop[i] or
          21.43 > tries[i] and 21.43 <= pens[i] and 13 > f0r[i]
             and 80 > aga[i] and 23 > drop[i] and 14 <= drop[i] or
          10 <= drop[i] and 21 <= aga[i] or
          19 <= f0r[i] and 80 > aga[i] and 1.99 <= tries[i] or
          10 <= f0r[i] and 13 > aga[i] or
          18 <= f0r[i] and 23 > aga[i] or
          21.43 <= aga[i] or
          13 > f0r[i] or
          7 <= drop[i] or
          7 <= conv[i] or
          18 <= aga[i] or
          13 <= pens[i] :
                guess[i] = "lost"

        else :
                guess[i] = "draw"
```

Fig. 11. The best evolved model from the fixing return values restriction to Fig. 9.

TABLE V
CLASSIFICATION ERROR OF THE BEST SOLUTIONS FOUND DURING TRAINING ACROSS ALL GRAMMARS INVESTIGATED, BOTH THEIR TRAINING AND TEST SET PERFORMANCE IS REPORTED.

Grammar	Training Error	Test Erro
Initial Grammar	0.00	0.00
Reduced Variables	0.28	0.20
Restricted Conditions	0.04	0.03
Restricted Conditions + Variable Statistics	0.04	0.08
Restricted Conditions + Variable Statistics + Fixed Returns	0.04	0.1
zeroR (`won` as the majority class)	0.51	0.50
J48 (Weka)	0.02	0.05

mately 96% is achieved over the pre-match models.

Table V captures the training and test classification errors for the best models found during training on each grammar. As can be seen from the data, the best model evolved from the restricted conditions grammar produces the best test set classification accuracy. Even though the initial grammar model produces perfect predictive accuracy it is not useful in practice. The model generated by the restricted conditions grammar can be used mid-game to provide a better classification accuracy than the model generated by the reduced variables grammar, which is used pre-game. When compared against the baseline methods of zeroR (majority class) and J48 (C4.5-like decision tree learning from Weka) on both training and test datasets it can be seen that all models outperform the majority class, and the best evolved models, in particular the restricted grammar, outperform J48 on the test dataset.

We also report the expressed genome lengths for each grammar in Fig. 12 (bottom). Interestingly all have similar genome growth behaviour rapidly converging towards solutions which utilise approximately 25-50 codons on average, followed by a growth phase towards approximately 200 codons, with the exception of the fixed output grammar, which demonstrates much more explosive growth. The fixed output grammar (a version of which is illustrated earlier in the paper in Section 2.3) forces the `if<cond>` to return `won`, the `elif<cond>` to return `lost`, and `else` to return `draw`. Based on what we observe in the expressed genome lengths plot this causes a rapid growth in the number of expressed codons relative to the other grammars, which results in a reduced predictive accuracy of the evolved models. When comparing training and test classification accuracies it would also appear that model overfitting has occured.

V. CONCLUSIONS AND FUTURE WORK

In this paper we explored a series of grammatical encodings that could generate decision tree-like models capable of predicting the outcome of international rugby matches. Several variations of the grammar were examined and comparisons to other methods made. Comparable performance was observed. Overfitting was also observed in the evolved solutions however this could be attributed to the limited number of games played in the six nations (5 matches per team per season). Looking to the future we would like to scale up the data set, and hope to gain access to data from one of the larger rugby competitions, such as the English Rugby Premiership (22 matches per team), France's Top 14 (22 matches per team), or The Pro 12 Championship (22 matches per team). This commercially available data contains a larger set of recorded variables, as well as having a larger sample size than the data used for this initial study. The temporal aspects of the data noted above are also of interest going forward, as would incorporating meteorological data in the dataset. It is hoped that with the increased dataset, and strategies to prevent overfitting that we will generate more powerful models. More generally, we note that the importance of incorporation of domain knowledge into many sports analytics applications suggests that grammar-based methods of genetic programming are likely to have particular utility. This paper introduces this methodology into this application domain.

ACKNOWLEDGMENTS

This research is based upon works supported by the Science Foundation Ireland under Grant No. 13/IA/1850.

REFERENCES

[1] G. R. Lindsey, "Statistical data useful for the operation of a baseball team," *Operations Research*, vol. 7, no. 2, pp. 197–207, 1959.

[2] M. Lewis, *Moneyball: The art of winning an unfair game*. WW Norton & Company, 2004.

[3] T. Sawchik, *Big Data Baseball*. Flatiron Books, 2015.

[4] *Journal of Quantitative Analysis in Sports. http://www.degruyter.com/view/j/jqas*. [Online]. Available: http://www.degruyter.com/view/j/jqas

[5] C. Graham, "Baseball enigma: The optimal batting order," in *Proceedings of MIT Sloan Sports Analytics Conference*, 2012. [Online]. Available: http://www.sloansportsconference.com/wp-content/uploads/2012/02/1-Baseball-Enigma-MIT-Sloan-2012-C-Graham.pdf

[6] A. Tsakonas, G. Dounias, S. Shtovba, and V. Vivdyuk, "Soft computing-based result prediction of football games," in *The Ist International Conference on Inductive Modelling (ICIM'2002)*, V. Hrytsyk, Ed., Lviv, Ukraine, 20-25 May 2002, pp. 15–23. [Online]. Available: http://www2.ba.aegean.gr/members/tsakonas/Lvov2002.pdf

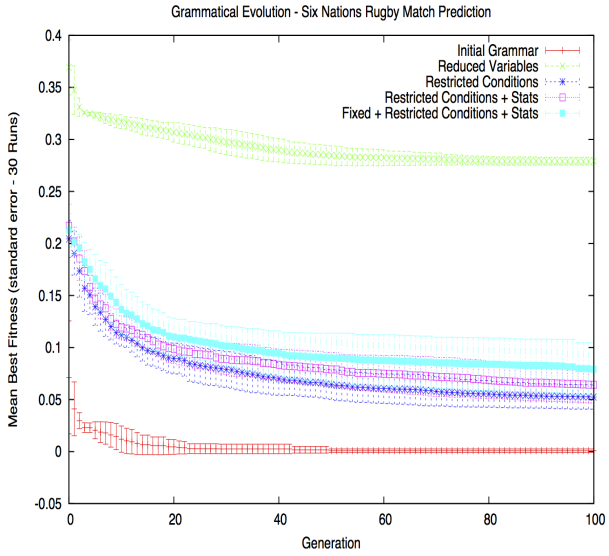
[7] EY, "The economic impact of rugby world cup 2015," Tech. Rep., Accessed 6 November 2015 2015. [Online]. Available: http://www.ey.com/Publication/vwLUAssets/EY-rugby-world-cup-final-report/\$FILE/EY-rugby-world-cup-final-report.pdf

[8] R. I. McKay, N. X. Hoai, P. A. Whigham, Y. Shan, and M. O'Neill, "Grammar-based genetic programming: a survey," *Genetic Programming and Evolvable Machines*, vol. 11, no. 3/4, pp. 365–396, Sep. 2010, tenth Anniversary Issue: Progress in Genetic Programming and Evolvable Machines.

[9] A. Brabazon, M. O'Neill, and S. McGarraghy, *Natural Computing Algorithms*. Springer, 2015.

[10] M. O'Neill and C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language*, ser. Genetic programming. Kluwer Academic Publishers, 2003, vol. 4. [Online]. Available: http://www.wkap.nl/prod/b/1-4020-7444-1

[11] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments*, ser. Studies in Computational Intelligence. Springer, Apr. 2009, vol. 194. [Online]. Available: http://www.springer.com/engineering/book/978-3-642-00313-4



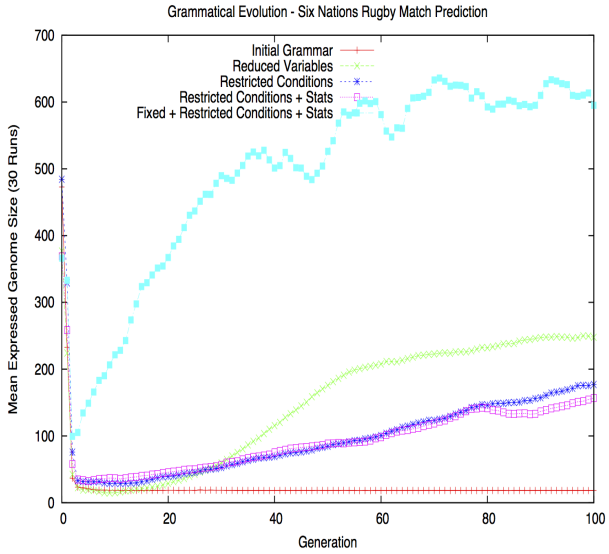


Fig. 12. Fitness and length plots averaged over 30 replications on the training dataset.

[12] A. Brabazon and M. O'Neill, *Biologically Inspired Algorithms for Financial Modelling*, ser. Natural Computing Series. Springer, 2006.

[13] P. G. Espejo, S. Ventura, and F. Herrera, "A survey on the application of genetic programming to classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 2, pp. 121–144, Mar. 2010.

[14] J. R. Koza, "Concept formation and decision tree induction using the genetic programming paradigm," in *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, ser. Lecture Notes in Computer Science, H.-P. Schwefel and R. Männer, Eds., vol. 496. Dortmund, Germany: Springer-Verlag, 1-3 Oct. 1991, pp. 124–128. [Online]. Available: http://www.genetic-programming.com/jkpdf/ppsn1990.pdf

[15] H. Iba, H. de Garis, and T. Sato, "Genetic programming using a minimum description length principle," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. MIT Press, 1994, ch. 12, pp. 265–284. [Online]. Available: http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/etl-tr-93-15.pdf

[16] D. J. Montana and S. Czerwinski, "Evolving control laws for a network of traffic signals," in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Stanford University, CA, USA: MIT Press, 28–31 Jul. 1996, pp. 333–338. [Online]. Available: http://vishnu.bbn.com/papers/gp96.pdf

[17] N. I. Nikolaev and V. Slavov, "Inductive genetic programming with decision trees," in *9th European Conference on Machine Learning*, ser. Lecture Notes in Computer Science, M. van Someren and G. Widmer, Eds., vol. 1224. Prague, Czech Republic: Springer, 23-26 Apr. 1997, pp. 183–190.

[18] A. Agapitos, M. O'Neill, A. Brabazon, and T. Theodoridis, "Maximum margin decision surfaces for increased generalisation in evolutionary decision tree learning," in *Proceedings of the 14th European Conference on Genetic Programming, EuroGP 2011*, ser. LNCS, S. Silva, J. A. Foster, M. Nicolau, M. Giacobini, and P. Machado, Eds., vol. 6621. Turin, Italy: Springer Verlag, 27-29 Apr. 2011, pp. 61–72.

[19] M. Sprogar, "Prudent alignment and crossover of decision trees in genetic programming," *Genetic Programming and Evolvable Machines*, online first.