Constraint Programming and Machine Learning for Interactive Soccer Analysis

Robinson Duque[1], Juan Francisco Díaz[1], and Alejandro Arbelaez[2]

[1] Universidad del Valle, Cali, Colombia
{robinson.duque, juanfco.diaz}@correounivalle.edu.co
[2] Insight Centre for Data Analytics, University College Cork, Ireland
{alejandro.arbelaez}@insight-centre.org

Abstract. A soccer competition consists of n teams playing against each other in a league or tournament system, according to a single or double round-robin schedule. These competitions offer an excellent opportunity to model interesting problems related to questions that soccer fans frequently ask about their favourite teams. For instance, at some stage of the competition, fans might be interested in determining whether a given team still has chances of winning the competition (i.e., finishing first in a league or being within the first k teams in a tournament to qualify to the playoff). This problem relates to the elimination problem, which is NP-complete for the actual FIFA pointing rule system (0, 1, 3), zero point to a loss, one point to a tie, and three points to a win. In this paper, we combine constraint programming with machine learning to model a general soccer scenario in a real-time application.

1 Introduction

Soccer fans usually have questions related to their favourite teams and most of the time they are subject to media speculations that are sometimes proved wrong. Many domestic leagues use a two-stage tournament structure with a single or double round-robin tournament for the regular season and a final knockout stage (aka playoffs). The first stage of the league typically features between 16 and 30 teams, each team faces each other team once or twice with *home* and *away* matches distributed evenly in the regular season. Depending on the results of the matches, every team is awarded some points under the FIFA three-point-rule (three points for a victory, one point for a draw, and zero points for a defeat), and the top k teams (typically eight) qualify for the playoffs. The elimination problem is well-known in sports competitions, particularly from baseball [1,2] and consists in determining whether at some stage of the competition a given team still has the opportunity to be within the top teams to qualify for playoffs. The complexity of the problem depends on the score system for the results of the matches. In [3,4] the authors showed that the elimination problem is NP-complete for the current FiFA score system (0, 1, 3). However, interestingly [4] pointed out that with the old FIFA score system (0, 1, 2) from the 90's, the elimination problem could be solved in polynomial time using a network

flow algorithms as first proposed by [5]. In this paper we attempt to present a general model to simulate scenarios and problems where fans can formulate queries about the positions of the teams at the end of a tournament, e.g., Will R. Madrid be in a better position than 3. To this end, we propose a combination of constraint programming (CP) with machine learning (ML) to answer soccer related queries.

2 CP Model for Soccer Queries

Constraint programming (CP) is a powerful technique to solve combinatorial problems which combines backtracking with constraint propagation. At each step a value is assigned to some variable. Each assignment is combined with a look-ahead process called constraint propagation which can reduce the domains of the remaining variables. In the following, we describe a CP formulation for soccer competitions, we start by offering a list of variables and notations for a basic soccer model.

- n: number of teams in the competition;
- T: set of team indexes in the competition;
- i, j: team indexes, such that $(i, j \in T)$;
- p_i: initial points of team i. If i has not played any games, then $p_i = 0$;
- F: number of fixtures left to be played in the competition. A fixture consists of one or more games between competitors;
- k: represents a fixture number, $(1 \leq k \leq F)$;
- G: set that represents the schedule of the remaining games to be played. Every game is represented as a triple $ng_e = (i, j, k) \wedge 0 \leq e \leq |G|$, where k is the fixture when both teams (i and j) meet in a game;
- pt_{ik}: represents the points that team i gets in fixture k, $(1 \leq k \leq F$ and $pt_{ik} \in \{0, 1, 3\})$. If team i is not scheduled to play fixture k, then $pt_{i,k} = 0$.
- tp_i: total points of team i at end of the competition;
- geq_{ij}: boolean variable indicating if team j has greater or equal total points as i: if $tp_j \geq tp_i$ then $geq_{ij} = 1$; otherwise $geq_{ij} = 0, \quad (\forall i, j \in T)$;
- eq_{ij}: boolean variable indicating if two different teams i and j tie in points at the end of the competition: if $tp_j = tp_i$ and $i \neq j$ then $eq_{ij} = 1$; otherwise $eq_{ij} = 0, \quad (\forall i, j \in T)$.
- pos_i: position of team i at the end of the competition;
- $worstPos_i$: upper bound for pos_i;
- $bestPos_i$: lower bound for pos_i;

Position in Ranking Queries: we use this set of variables to represent queries about positions of the teams at the end of the competition (e.g., R.Madrid will be in position 3).

- P: set of possible position in ranking queries, defined as a set of triples $np_b = (i, opr_i, ptn_i)$ and $0 \leq b \leq |P|$;
- opr_i: logical operator ($opr_i \in \{<, \leq, >, \geq, =\}$) to constrain team i;
- ptn_i: denoting the expected position for team i; $1 \leq ptn_i \leq n$;

2.1 CP Model Formulation

Basic Soccer Model: Constraints (1), (2), and (3) represent a valid game point assignment (0,3), (3,0) or (1,1) for each game $ng_e \in G$ between two teams i and j in a fixture k:

$$(0 \leq pt_{ik} \leq 3) \wedge (0 \leq pt_{jk} \leq 3) \quad \forall ng_e \in G \wedge ng_e = (i,j,k) \tag{1}$$

$$(pt_{ik} \neq 2) \wedge (pt_{jk} \neq 2) \quad \forall ng_e \in G \wedge ng_e = (i,j,k) \tag{2}$$

$$2 \leq pt_{ik} + pt_{jk} \leq 3 \quad \forall ng_e \in G \wedge ng_e = (i,j,k) \tag{3}$$

Constraint (4) corresponds to the final points tp_i of a team i. It is the addition of the initial points p_i and the points pt_{ik} obtained in every fixture k:

$$tp_i = p_i + \sum_{k=1}^{F} pt_{ik} \quad \forall i \in T \tag{4}$$

Constraints (5) to (8) are used to calculate final positions. All the final positions must be different and every position is bounded by $bestPos_i$ and $worstPos_i$:

$$\begin{aligned} geq_{ij} &= \begin{cases} 1, & \text{if } tp_j \geq tp_i \\ 0, & otherwise \end{cases} \quad \forall i,j \in T \\ worstPos_i &= \sum_{j=1}^{n} geq_{ij} \quad \forall i,j \in T \end{aligned} \tag{5}$$

$$\begin{aligned} eq_{ij} &= \begin{cases} 1, & \text{if } tp_j = tp_i \text{ and } i \neq j \\ 0, & otherwise \end{cases} \quad \forall i,j \in T \\ bestPos_i &= worstPos_i - \sum_{j=1, j\neq i}^{n} eq_{ij} \quad \forall i,j \in T \wedge i \neq j \end{aligned} \tag{6}$$

$$bestPos_i \leq pos_i \leq worstPos_i \quad \forall i \in T \tag{7}$$

$$alldifferent(pos_1, ..., pos_n) \tag{8}$$

Position in Ranking Queries: involves a set of constrained teams and indicates whether a given team can be above, below, or at a given position ptn_i, constraint (9) depicts the five possibilities:

$$\forall np_b \in P \wedge np_b = (i, opr_i, ptn_i) \begin{cases} pos_i = ptn_i, & \text{if } opr_i \text{ is } = \\ pos_i < ptn_i, & \text{if } opr_i \text{ is } < \\ pos_i \leq ptn_i, & \text{if } opr_i \text{ is } \leq \\ pos_i > ptn_i, & \text{if } opr_i \text{ is } > \\ pos_i \geq ptn_i, & \text{if } opr_i \text{ is } \geq \end{cases} \tag{9}$$

2.2 Variable/Value Selection

In this section we propose some query based heuristics for variable/value selection. First we introduce a set of required variables in order to describe a priority mechanism to select the team variables constrained in queries P:

- $spos_i$: starting position of team i before any branching strategy is applied;
- pri_i: denoting the priority of team i to be selected during branching, If team i does not appear in any query, then $pri_i = 0$;
- str_i: denoting the global branching strategy for the variables pt_{ik} of a particular team i in every fixture k. str_i starts with "tie" as a default value.

Heuristics for Position in Ranking Queries (P): Recall from (9) that we use the position pos_i to constrain a team to a wanted position ptn_i. Suppose we have the query ($pos_i < 8$). It's natural to try $str_i = win$ and assign a priority using the position ptn_i from the query. We depict in (10) some general rules for variable value selection:

$$\forall np_b \in P \wedge \atop np_b = (i, opr_i, ptn_i) \begin{cases} \text{if } opr_i \text{ is } < \text{ or } \leq, & pri_i = n - ptn_i \wedge str_i = win \\ \text{if } opr_i \text{ is } > \text{ or } \geq, & pri_i = ptn \wedge str_i = lose \\ \text{if } opr_i \text{ is } =, & pri_i = ptn \wedge \begin{cases} str_i = lose, \text{ if } ptn_i > n/2 \\ str_i = win, \text{ otherwise} \end{cases} \end{cases} \tag{10}$$

Interestingly the defined heuristics in (10) for queries with the "=" operator seem to fail quite often (see section 3). Suppose a scenario with a query ($pos_i = 7$) where $spos_i = 9$ with $F = 8$ fixtures to play. Given that the starting position is 9 and we have to reach position 7, the global branching strategy $str_i = win$ causes that pos_i overshoots position 7 and would require many backtracks of the search algorithm in order to reach such position, therefore, it might be useful to perform a bias search and in the following section we tackle this problem by using machine learning.

Machine Learning for Value Selection: For teams constrained with the "=" operator, we decided to assign a high priority ($pri_i = |spos_i - ptn_i| \cdot n$) for variable selection and to avoid position overshooting, we trained a classifier that selects among 9 branching strategies: S1= [1,0,0], S2= [0,1,0], S3= [0,0,1], S4= [0.5,0.5,0], S5= [0.5,0,0.5], S6= [0,0.5,0.5], S7= [0.5,0.25,0.25], S8= [0.25,0.5,0.25], S9= [0.25,0.25,0.5]. Each strategy defines probabilities to select among [win, tie, lose] respectively, e.g, S7 means that for a team i, every variable pt_{ik} will be assigned win with a probability of 0.5, tie and $lose$ with a probability of 0.25 each. We use the selected strategy with a restart-based search; therefore we restart the algorithm when some cutoff in the execution time is met. (3 secs in this paper). Notice that that we excluded a strategy [1/3, 1/3/, 1/3] as preliminarily tests showed a poor performance for this alternative.

Training the classifier: In order to train this classifier, we created a total of 500 P queries with the equality operator at different stages of a tournament (fixture 7, 9, 11, 14 and 16) with 18 teams, scheduled in a single round robin.

We ran every query with each of the 9 branching strategies in order to get the strategy that solved the instance in the shortest time and created a data set with the following features: starting position ($spos_i$), wanted position (ptn_i), direction and distance ($spos_i - ptn_i$), fixtures to play (F), range rate ($|spos_i - ptn_i|/n$), best executing strategy $\in \{S1, S2, S3, S4, S5, S6, S7, S8, S9\}$. The classifier was trained with Weka (V3.6.12) using the J48 decision tree and the objective was that given a P query with the equality operator "=", the classifier assigns one of the 9 branching strategies to the constrained team.

		P Queries						Running Times					
Fixture	Test \ Sup.	2	3	4	5	7	9	2	3	4	5	7	9
Fixture 7	CP	24	28	38	46	50	52	**.22**	**.48**	**.31**	**.41**	**.77**	**1.10**
	CP-ML	**6**	**9**	**14**	**21**	**27**	**35**	1.19	1.34	1.83	2.40	3.28	3.09
Fixture 9	CP	23	28	35	46	49	44	**.08**	**.41**	**.23**	**.65**	**.31**	**.72**
	CP-ML	**1**	**4**	**9**	**18**	**27**	**31**	1.68	1.48	2.25	2.81	2.78	1.66
Fixture 11	CP	25	27	31	44	49	47	**.56**	**.10**	**.75**	**.69**	**.86**	**.45**
	CP-ML	**2**	**7**	**11**	**18**	**31**	**34**	1.53	1.09	2.01	2.67	2.56	2.46
Fixture 14	CP	23	33	38	41	51	45	**.42**	**.13**	**.49**	**1.01**	1.53	**.74**
	CP-ML	**8**	**22**	**25**	**34**	**44**	**42**	1.17	1.29	1.46	1.48	**1.08**	1.47
Fixture 16	CP	23	31	29	**31**	**25**	**13**	**.12**	**.04**	**.89**	.55	.45	.65
	CP-ML	**19**	**25**	**25**	33	28	17	.56	.50	.93	**.05**	**.11**	**.03**
CP Results		Unsolved: 1069						Avg: 0.51 sec					
CP-ML Results		Unsolved: 627						Avg: 1.60 sec					

Table 1. Unsolved instances and average running times of CP and CP-ML

3 Empirical Evaluation

Tests Configuration: We evaluated our models using Mozart-Oz (V 1.4.0) as our CP reference solver. All the experiments were performed in a 4-core machine, featuring an Intel Core i5 processor at 2.3 Ghz and 4 GB of RAM. We focus our attention in the Colombian league (liga Postobón 2014-I) with 18 teams and 18 fixtures to play in a single round-robin schedule (17 fixtures + 1 extra fixture for the derbies). We provided five experimental scenarios (i.e., fixtures 7, 9, 11, 14, and 16). We also created a series of instances for each fixture (100 with 2 suppositions, 100 with 3 suppositions, and the same for 4, 5, 7, and 9 suppositions). Each instance (3000 in total) was executed with a time limit of 30 seconds. We recall that our models are implemented in SABIO, a Web based application where long answer times are not desirable. We experimented two scenarios: the basic CP implementation using the heuristics for position in ranking queries and the CP-ML implementation configured with 10 restarts (i.e., 3 seconds per restart), featuring the basic heuristics and the machine learning classifier for equality constraints.

Tests Results: Table 1 shows the number of unsolved instances and the average runtimes of the solved ones in our experiments. We observe 1069 unsolved instances with the CP model and we attribute this to 2 main reasons: first, the position bounds (i.e., $bestPos_i$ and $worstPos_i$) can only be computed

after finding the total points (tp_i) for all the teams in the competition. As a result, position in ranking constraints standing are validated only when the search algorithm performs a complete game points assignment for all teams. Second, we observed that our variable/value selection heuristics struggle with queries related to the "=" operator and the lack of a biased search causes position overshooting. We also observed that our CP-ML implementation seems to perform better and the classifier improves the effectiveness of the algorithm by reducing the number of unsolved instances from 1069 to 627 while displaying a small trade-off in running time.

4 Conclusions

In this paper we have combined the use of constraint programming and machine learning to solve general soccer fan queries at different stages of a competition and presented 2 alternative solutions CP and CP-ML.Our computational experiments showed that our CP-ML model improves CP effectiveness since it performs a query biased search. We also plan to extend our models to deal with more queries such as determining the maximum number of games that a team can afford to lose and still qualify to the playoffs and also explore the implementation of a MIP model, based on the work of Ribeiro and Urrutia [6].

Acknowledgments

We would like to thank Luis Felipe Vargas, María Andrea Cruz and Carlos Martínez for developing early versions of the CP model under the supervision of Juan Francisco Díaz. Robinson Duque is supported by Colciencias under the PhD scholarship program. Alejandro Arbelaez is supported by the DISCUS project (FP7 Grant Agreement 318137), and Science Foundation Ireland (SF) Grant No. 10/CE/I1853.

References

1. Schwartz, B.L.: Possible winners in partially completed tournaments. SIAM Review **8**(3) (1966) 302–308
2. Hoffman, A., Rivlin, T.: When is a team mathematically eliminated?, Princeton, NJ, Princeton Symposium on Mathematical Programming (1967) 391–401
3. Kern, W., Paulusma, D.: The new fifa rules are hard: complexity aspects of sports competitions. Discrete Applied Mathematics **108**(3) (2001) 317–323
4. Bernholt, T., Gälich, A., Hofmeister, T., Schmitt, N.: Football elimination is hard to decide under the 3-point-rule. In: MFCS. (1999) 410–418
5. Wayne, K.D.: A new property and a faster algorithm for baseball elimination. SIAM Journal on Discrete Mathematics **14**(2) (2001) 223–229
6. Ribeiro, C.C., Urrutia, S.: An application of integer programming to playoff elimination in football championships. ITOR **12**(4) (2005) 375–386