International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184, Volume 7, Number 2 (April 2018), pp.22-28
© MEACSE Publications
http://www.meacse.org/ijcar

Generation and Enumeration of Final Table States of Football Tournaments Using a Blind Adaptive Filtering Algorithm

Mouslem Damkhi and Hüseyin Pehlivan
Department of Computer Engineering, Karadeniz Technical University, Trabzon, Turkey

Abstract

Football tournaments always hold the standings of teams in a table, such as the numbers of wins (*W*), draws (*D*) and losses (*L*), as well as the number of points (*P*). The final state of the table is obviously constructed from the eventual standings that point out the resulting positions of the teams in the table. Using the team standings, it is quite possible to compute possible states of a tournament table. This paper focuses on generating and enumerating all possible states of a final table for football tournaments according to the number of participating teams. The use of a blind search algorithm yields many table states, most of which are not valid in terms of the standings. An efficient algorithm is proposed to filter out such invalid states. The proposed algorithm evaluates the validity of a final table state by performing the exploration of a result-based tournament graph derived from that state. One can use the algorithm to determine which team standings are required to attain a certain position in the table. The paper also presents and discusses the experimental results of the algorithm on the tournaments with up to eight teams.

Keywords: tournament, graph, enumeration, number partition, combinatorics

Introduction

Sports events such as the FIFA World cup and the Olympic Games have been attracting a lot of interest among people more recently more than any other time. This interest has not only been restricted to the general public, but it has also spread rapidly to scientists and researchers. In the literature, most of studies related to sports activities have heavily focused on round-robin tournaments [2, 9, 10], as some others have done on the automation scheduling of games, referees and venues for some sports like football [1], cricket [4], ice hockey [7] and basketball [11]. A few of the previous studies have approached the round-robin tournaments from a different angle other than scheduling. For instance, a study has been done by Eggar [5] addressing the number of players with a high average of wins in a round-robin tennis tournament. The study of McSherry [8] explained a way to conclude the numbers of wins, draws and losses of teams from their points in the final tournament's table. Charon [3] proposed a method to solve the linear ordering problem for any weighted tournament. An algorithm was proposed by Hemasinha [6] which generates the score sequence of every tournament up to a given size. In this paper, we propose an algorithm which determines the possible states of final tables for football tournaments where the teams compete in a round-robin fashion by filtering the outputs of a blind search algorithm.

The incomes generated from some footballing events and local leagues play a major role in the economies of some countries through international tourism, TV networks and advertisements by making many companies and people to invest in football teams. The demotion or promotion of the teams to a lower or higher league would significantly influence their incomes. Hence it would be of vital importance to predict the final position of a team at the end of a tournament or league. Our approach provides a convenient way to determine what *WDL* value would be adequate to reach the desired position for a team.

Problem description

During a round-robin tournament every team must play against all the other teams, i.e. in a tournament between n team, every team plays n-1 games. If we denote the number of games played by each team as g, we can write:

$g = n - 1$ (1)

The total number of the games G that is played in the tournament can be calculated by the combination C(n, 2) as following:

$G= C(n, 2) = n!/2!(n - 2)! = n(n-1)/2$ (2)

LetTWDL denote a table that is derived from the W, D and L columns of a tournament's final table. Table 1 shows an instance of such a table, which is obtained from the final table of Group E in the 2006 FIFA World Cup [12].

Table 1. The T_{WDL} table derived from the final table of Group E in the 2006 FIFA World Cup

W	*D*	*L*
2	1	0
2	0	1
1	0	2
0	1	2

Table 1 can also be represented as TWDL=((2, 0, 1), (2, 0, 1), (1, 1, 1), (0, 1, 2)). The values of each g and G can be calculated from a TWDL table as following:

$g = (\sum_{k=1}^{n} W\mathrm{k})+ (\sum_{k=1}^{n} D\mathrm{k})/2 = (\sum_{k=1}^{n} L\mathrm{k})+ (\sum_{k=1}^{n} D\mathrm{k})/2$, where $1 \leq k \leq n$ (3)

$G = Wk + Dk+ Lk$, where $1 \leq k \leq n$ (4)

Each team Tk at the kth position gains Pk points based on its values of Wk and Dk, which can be calculated by the following formula:

$Pk = 3Wk + Dk$, where $1 \leq k \leq n$ (5)

The WDL values of the teams in a TWDL table must be in a descending order depending on their points, where:

$Pk \geq Pk+1$, where $1 \leq k \leq n- 1$ (6)

The set of all the possible WDL values that can take place in the final table of a tournament between 4 teams is:

SWDL(4) = {(3, 0, 0), (2, 1, 0), (2, 0, 1), (1, 2, 0), (1, 1, 1), (1, 0, 2), (0, 3, 0), (0, 2, 1), (0, 1, 2), (0, 0, 3)}.

In general, the values in the set SWDL can be calculated by formula (7) as follows:

$SWDL(n) = \{(w, d, l) \mid \forall w \in \{0,1, \dots , n -1\}, \forall d \in \{0,1, \dots , n -1\},$
$\forall l \in \{0,1, \dots , n -1\}, w + d + l = n -1\}$ (7)

The number of elements of SWDL is given by:

$N(SWDL(n)) = n(n+1)/2$ (8)

Based on the number of elements of SWDL, the blind search algorithm generates a very large number of states given by the formula (n(n + 1)/2)n.The following algorithm shows the pseudo code which implements the blind search algorithm:

<u>Blind Search Algorithm</u>
Intput: number of teams n
Output: the states of TWDL (STWDL[(n(n+1)/2)^n][n][3])

```
SWDL[n*(n+1)/2][3] : the set of all the possible WDL values
TWDL[n][3]         : the tournament table

Main Procedure: blindSearch(n)
1:  form SWDL based on formula (7)
2:  statesGenerating(n, 0, TWDL)

Procedure: statesGenerating(n, index, TWDL[][])
1:  if index <n then
2:     for i ← 0 to (n*(n+1)/2)-1 do
3:        TWDL[ind][0] ← SWDL[i][0]
4:        TWDL[ind][1] ← SWDL[i][1]
5:        TWDL[ind][2] ← SWDL[i][2]
6:        statesGenerating(n, ind+1, TWDL)
7:  else
8:     sort TWDL into STWDL
```

Most of the resulting states are not valid, because they do not respect the relations (3) and/or (6). For example, in the case of 4 teams the state T_{WDL} = ((2, 0, 1), (3, 0, 0), (1, 0, 2), (1, 1, 1)) can be generated by the blind search algorithm, but it doesn't respect any of the relations (3) and (6). In this paper we propose an algorithm which distinguishes the valid states from those that can be generated by the blind search algorithm.

The Proposed Approach

A tournament graph is a complete graph (every pair of nodes is connected by a unique edge), where each of its nodes represents a team and each of the edges represents a game between two teams. The edge representing a win or a loss is directed, as a draw is represented by an undirected edge. A directed edge starts from the winner to the loser. Figure 1 represents such a tournament graph that leads to the T_{WDL} in Table 1. Our proposed approach aims at determining the possibility of finding a tournament graph based on a state of T_{WDL} table, where the teams must be in a descending order depending on their number of points. Table 2 illustrates how to find the graph shown in Figure 1 from the T_{WDL} table given in Table 1 by determining the edges of the graph and their directions based on the WDL values of the teams.

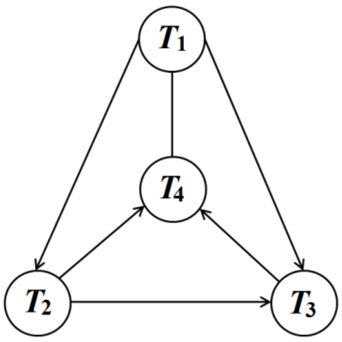

Fig. 1. The tournament graph which leads to the T_{WDL} table shown in Table 1

The steps of how to construct a tournament graph are illustrated in Table 2. Each selection of the edges is done by comparing the WDL values of the related two teams. Note that the related digits of the WDL values from which the edge selection is made are underlined at each step. The edges selections are conducted in accordance with the ascending order of the related W, D and L values for each team. For example, the draw of T_1 is first sought, because its value of D is 1. Among the other teams, only T_4 has a value of D different from 0. So the first edge selection for T_1 is made for the game against T_4 at step 3. After a valid edge selection is made, the relevant *WDL* values are gradually reduced to show the consequences of the selection. For example, with respect to the selected edge for the game between T_1 and T_2 at step 4, a win is taken out of the *WDL* values of T_1 and a corresponding loss is taken out of the *WDL* values of T_2. At step 9 the values of each of *W*, *D* and *L* of all the teams are equal to 0. In this case the process reaches the end, and the current state of T_{WDL} is considered as a valid state.

Table 2. The steps of finding a tournament graph for T_{WDL} = ((2, 0, 1), (3, 0, 0), (1, 0, 2), (1, 1, 1))

Step	T_1	T_2	T_3	T_4	Game	Edge
1	$2\underline{1}0$	$2\underline{0}1$	102	012	T_1- T_2	-
2	$2\underline{1}0$	201	$1\underline{0}2$	012	T_1- T_3	-
3	$2\underline{1}0$	201	102	$0\underline{1}2$	T_1- T_4	Undirected
4	$\underline{2}00$	$20\underline{1}$	102	002	T_1- T_2	From T_1 to T_2
5	$\underline{1}00$	200	$10\underline{2}$	002	T_1- T_3	From T_1 to T_3
6	000	$\underline{2}00$	$10\underline{1}$	002	T_2- T_3	From T_2 to T_3
7	000	$\underline{1}00$	100	$00\underline{2}$	T_2- T_4	From T_2 to T_4
8	000	000	$\underline{1}00$	$00\underline{1}$	T_3- T_4	From T_3 to T_4
9	000	000	000	000	-	-

Table 3 shows the steps involved in trying to find a tournament graph based on the state T_{WDL}=((2, 1, 0), (2, 0, 1), (2, 0, 1), (0, 1, 2)). At step 8 the process reaches a situation where the value of *W* for T_2 is equal to 1 and only T_4 has a value of $L>0$, but the edge between T_2 and T_4 is already selected at step 7. In this case we can say that there is no tournament graph based on the current state of T_{WDL}, and therefore the state is not valid.

Table 3. The steps of finding a tournament graph for T_{WDL} = ((2, 1, 0), (2, 0, 1), (2, 0, 1), (0, 1, 2))

Step	T_1	T_2	T_3	T_4	Game	Edge
1	$2\underline{1}0$	$2\underline{0}1$	201	012	T_1- T_2	-
2	$2\underline{1}0$	201	$2\underline{0}1$	012	T_1- T_3	-
3	$2\underline{1}0$	201	201	$0\underline{1}2$	T_1- T_4	Undirected
4	$\underline{2}00$	$20\underline{1}$	201	002	T_1- T_2	From T_1 to T_2
5	$\underline{1}00$	200	$20\underline{1}$	002	T_1- T_3	From T_1 to T_3
6	000	$\underline{2}00$	$20\underline{0}$	002	T_2- T_3	-
7	000	$\underline{2}00$	200	$00\underline{2}$	T_2- T_4	From T_2 to T_4
8	000	$\underline{1}00$	200	001	-	-

The following algorithm shows the pseudo code which implements the proposed approach:

```
TWDL State Validity Checking Algorithm

Intput: the number of teams (n) and a state of TWDL table (TWDL[n][3])
Output: true or false

tracker[n][n] : tracks the conflicts

Main Procedure: isValid(n, TWDL[][])
1:  set all tracker[i][j] to 0
2:  trackingInds[] ← {1, 1, 1}
3:  return stateValidity(n, 0, trackingInds)
```

```
Procedure: stateValidity(n, k, trackingInds_p[])
1:  valid← false
2:  if the points of the teams are in a descending order then
3:      trackingInds[] ← trackingInds_p[]
4:      find the non-zero minimum of W, D and L values for k-th team and set an index variable (minInd) to 0, 1 or 2 accordingly, or −1 if all is 0
5:      if minInd = −1 then
6:          if k = n-1 then
7:              if TWDL elements are all 0 then
8:                  valid ← true
9:          else
10:             trackingInds[] ← {k+2, k+2, k+2}
11:             valid ← stateValidity(n, k+1, trackingInds)
12:     else
13:         for i ← trackingInds[minInd] to n-1 do
14:             if TWDL[i][2−minInd] > 0 and tracker[k][i] = 0 then
15:                 TWDL[k][minInd] ← TWDL[k][minInd] − 1
16:                 TWDL[i][2−minInd] ← TWDL[i][2−minInd] − 1
17:                 tracker[k][i] ← 1
18:                 tracker[i][k] ← 1
19:                 trackingInds[minInd] ← i
20:                 valid← stateValidity(n, k, trackingInds)
21:                 if valid= true then
22:                     go to 28
23:                 else
24:                     tracker[k][i] ← 0
25:                     tracker[i][k] ← 0
26:                     TWDL[k][minInd] ← TWDL[k][minInd]+1
27:                     TWDL[i][2−minInd] ← TWDL[i][2−minInd]+1
28: return valid
```

By applying TWDL State Validity Checking Algorithm on every generated state from the Blind Search Algorithm we obtain the algorithm that generates all the valid final table states of football tournaments. The Final Table States of Football Tournaments Generating Algorithm is as follows:

Final Table States of Football Tournaments Generating Algorithm

Intput: the number of teams (n)
Output: the states of TWDL (STWDL[][n][3])

SWDL[n*(n+1)/2][3] : set of all the possible WDL values
TWDL[n][3]   : tournament final table

```
Main Procedure: validStatesGenerator(n)
1:  form SWDL based on formula (7)
2:  validStateGenerating(n, 0, TWDL)
```

```
Procedure: validStateGenerating(n, index, TWDL[][])
1:  if index <n then
2:      for i ← 0 to (n*(n+1)/2)-1 do
```

```
3:      TWDL[ind][0] ← SWDL[i][0]
4:      TWDL[ind][1] ← SWDL[i][1]
5:      TWDL[ind][2] ← SWDL[i][2]
6:      validStateGenerating(n, ind+1, TWDL)
7:  else
8:    valid ← isValid(n, TWDL)
9:    if valid = true then
10:     sort TWDL into STWDL
```

Computation Results

The proposed approach is programmed using the Java programming language and compiled by JDK 1.6 compiler. The machine on which our experiments are performed runs Windows 7 Ultimate 64-bit operating system and is equipped with an Intel core i5 1.60GHz CPU, 6GB of RAM and a WDC WD10JPVX-55JC3T0 ATA hard drive with 1 TB of memory space. A set of tournaments with the number of teams ranging between two and eight is examined during the experiments. Each experiment aims to determine the number of valid final table states and their execution times. The results of the experiments are shown in Table 4. Table 5 shows the percentages of the valid and invalid states of T_{WDL} that are generated by the Blind Search Algorithm. As seen in Table 5, the percentage of the generated invalid states by the blind search algorithm is very large compared to that of the valid states.

Table 4. Experimental results

n	Number of the valid states	Execution time
2	2	0
3	7	0
4	40	0
5	404	0
6	6317	3 seconds
7	131288	8 minutes
8	3366444	81 hours

Table 5. The percentage of each valid and invalid states

n	Number of all the states$(n(n+1))^n$	Valid states (%)	Invalidstates (%)
2	9	22.22	77.77
3	216	3.24	96.75
4	10000	0.40	99.60
5	759375	0.05	99.94
6	85766121	0.00	99.99
7	13492928512	0.00	99.99
8	2821109907456	0.00	99.99

Conclusion

In this paper we focus on generating and enumerating the final table states of football tournaments. The basis of our idea is to determine the numbers of won (W), drawn (D) and lost (L) games for each participanting team in a round-robin tournament by filtering the states generated by a blind search algorithm. In order to filter the generated states, the algorithm seeks to construct a tournament graph based on a given state. The proposed approach is experimented with up to eight teams, where the number of the final table states of the tournament and their execution times are measured.

The scope of the study can be extended to cover a competitive sports league environment. Since the demotion or promotion of the teams to a lower or higher league significantly influence their incomes, it is important for a team to predict its final position at the end of a tournament or league. The proposed study facilitates the future predictions of the teams by generating every possible state of the football tournaments.

References

[1] Atan, T., Huseyinoglu, O. P.: Simultaneous scheduling of football games and referees using Turkish league data. International Transactions in Operational Research. 24, 465- 484 (2017)

[2] Carlsson, M., Johansson, M., Larson, J.: Scheduling double round-robin tournaments with divisional play using constraint programming. European Journal of Operational Research. 259, 1180-1190 (2017)

[3] Charon, I., Hudry, O.: A branch-and-bound algorithm to solve the linear ordering problem for weighted tournaments. Discrete Applied Mathematics. 154, 2097-2116 (2006)

[4] Duckworth, F. C., Lewis, A. J.: A Fair method for resetting the target in interrupted oneday cricket matches. In: Wright, M. (ed.) OR Essentials, Operational Research Applied to Sports, 128-143. Palgrave Macmillan, England (2015)

[5] Eggar, M. H.: A tournament problem. Discrete Mathematics. 263, 281-288 (2003)

[6] Hemasinha, R.: An algorithm to generate tournament score sequences. Mathematical and Computer Modeling. 37, 377-382 (2003)

[7] Kyngas, J., Nurmi, K.: Scheduling the Finnish 1st Division Ice Hockey League. In: Lane, H. C., Guesgen, H. W. (eds.) Proceedings of the Twenty-Second International Florida Artificial Intelligence Research Society Conference, 195-200. AAAI Press, California (2009)

[8] McSherry, D.: Inferring wins, draws and losses from points scored in a sports tournament. Irish Math. Soc. Bulletin. 42, 48-53 (1999)

[9] Perez-Ceceres, L., Riff, M. C.: Solving scheduling tournament problems using a new version of CLONALG. Connection Science. 27, 5-21 (2015)

[10] Suksompong, W.: Scheduling asynchronous round-robin tournaments. Operations Research Letters. 44, 96-100 (2016)

[11] Westphal, S.: Scheduling the German Basketball League. Interfaces. 44, 498-508 (2014)

[12] 2006 FIFA World Cup Germany. FIFA. http://www.fifa.com/worldcup/archive/germany2006/groups/index.html. Accessed 22 January 2018