International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184, Volume 7, Number 3 (June 2018), pp.38-42
© MEACSE Publications
http://www.meacse.org/ijcar

A Point-Based Algorithm to Generate Final Table States of Football Tournaments

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

Abstract

Sports tournaments commonly place the points and scores of particing teams or players in tables. For football tournaments, the state of a final table is based basically upon the numbers of wins (W), draws (D) and losses (L) together with the sum of points (P) corresponding to team standings at the end of a tournament. This paper describes a point-based algorithm to generate and enumerate all possible states of final tables that can take place in a football tournament with a variable number of teams. The algorithm attempts to determine the possible final *WDL* values of the teams using their sums of points. The position of a team in the final table can lead to either that team's qualification to the next stage or its elimination from the tournament, which thus makes it important to predict the final position of the team at the end of the tournament. The study provides an elegant way to estimate the position of a team with a particular value of WDL in the final table. In the paper, the experimental results of the described algorithm are also presented and discussed, by comparison with those from a blind search algorithm, evaluating a set of tournaments with the number of teams ranging from two to eight.

Keywords: tournament, enumeration, number partition, combinatorics

Introduction

Recently, studies with great interests on sports have considerably increased due to the wide popularity of some sporting events such as the Olympic Games and the FIFA World Cup. Of the sports-related studies, the most recent ones seem to be those that mainly focus on the scheduling of round-robin tournaments [2, 8, 9] for some sports like football [1], basketball [10], ice hockey [7] and cricket [4]. Some others deal with round-robin tournaments from different perspectives; namely generating the score sequence of every tournament up to a given size [6], solving the linear ordering problem for the weighted tournaments [3] and addressing the number of players with a high average of wins [5]. In this paper, we propose an algorithm which determines the possible states of final tables for football tournaments, based on the final points of the teams.

The incomes generated in some countries from sporting events like football (TV rights and advertisements) also have a positive impact on the incomes of the participating teams. Hence the demotion or promotion of teams to a lower or higher league will significantly influence their incomes. The generation of the possible states of a tournament final table will help to predict whether a team can reach a specific position at the end of the tournament based on the current state of the table or not.

Problem description

A football tournament with n teams involves $g = n(n\text{-}1)/2$ games and each team plays n-1 games. Let $T_{WDL}(n)$ represent a final state of a tournament with n teams, which contains the final WDL values of participating teams with respect to the order of their number of points. For example the final table of Group E in the 2006 FIFA World Cup [11] can be represented as $T_{WDL}(4)$=((2, 0, 1), (2, 0, 1), (1, 1, 1), (0, 1, 2)). For a tournament with n-teams, the general set of possible WDL values that might be gained by a team can be generated as follows:

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

In this case, the number of WDL values in the set would be determined by the relation:

$N(S_{WDL}(n)) = n(n+1)/2.$ (2)

For instance, there are 10 distinct values of WDL for $n = 4$, as seen in the set:

$S_{WDL}(4) = \{(3, 0, 0), (2, 1, 0), (2, 0, 1), (1, 2, 0), (1, 1, 1),$
$(1, 0, 2), (0, 3, 0), (0, 2, 1), (0, 1, 2), (0, 0, 3)\}$ (3)

The variety of WDL values of teams leads to the occurrence of many different table states. To enumerate all possible states of a tournament table, the number of states based on the relation $N(S_{WDL}(n))$ that a blind search algorithm must evaluate would be:

$B(n) = (n(n+1)/2)^n$ (4)

In fact, not every evaluated state is valid. For example, the state T_{WDL} = ((2, 0, 1), (3, 0, 0), (1, 0, 2), (1, 1, 1)) is not valid because the WDL values are not in descending order with respect to the points of the teams. Also, there is no consistency between the WDL values, for instance, where the WDL=(1, 1, 1) contains a D value of 1 as all D values for the other teams are equal to 0. To generate the states of the final tournament table only the valid states must be taken into account, that is the validity of each generated state must be checked. When the number of the invalid states gets bigger, a larger time is spent on checking the states which will not be taken into account as final states. In this paper we propose an algorithm which minimizes the generation of the invalid states and generates all the possible final states based on the final points of the teams.

The Proposed Approach

In a football tournament with n teams, a team gains 3 points for each win and 1 point for each draw. At the end of the tournament, the number of points P_k of a team T_k at the kth position is:

$P_k = 3W_k + D_k$, where $1 \le k \le n$ (5)

Our approach is based basically on calculating the possible WDL values of teams through their number of points. Formula (6) defines the set of all the possible WDL values that can be calculated based on a specific number of points:

$S_{WDL}(P)=\{(w, d, l) \mid w = \lfloor P/3\rfloor\text{-}x, \forall x \in \{0, ..., \lfloor P/3\rfloor\}, d = P\text{-}3w, w+d\le n\text{-}1, l = n\text{-}1\text{-}w\text{-}d\}$(6)

The maximum number of points that can appear in the final table of a tournament with n teams is 3n-3. This value appears when the first team wins all its games against the other teams. Also, the minimum number of points is 0, which appears when the last team loses all its games. So, generally we can write:

$0 \le P \le 3n\text{-}3$ (7)

Since P = 3n-3 can be gained only by the first team and only the last team can have 0 points, formula (7) can be described in detail as follows:

$$\begin{cases} 0 < P_1 \le 3n\text{-}3 \\ 0 < P_k < 3n\text{-}3 \text{ , where } 1 < k < n \\ 0 \le P_n < 3n\text{-}3 \end{cases} \quad (8)$$

As $P_k \le P_{k\text{-}1}$ when $1 < k \le n$, formula (8) can be written as:

$$\begin{cases} 0 < P_1 \le 3n\text{-}3 \\ 0 < P_k < P_{k\text{-}1} \text{ , where } 1 < k < n \\ 0 \le P_n < P_{n\text{-}1} \end{cases} \quad (9)$$

Formula (9) represents the search space which our approach is based on during the generation of the final states. The used search space ensures that the generated states are in descending order depending on the points of the teams. The following algorithm shows the pseudo code which implements the proposed approach:

Point-Based Algorithm to Generate Final Table States of Football Tournaments

```
Intput: the number of teams (n)
Output: the states of TWDL (STWDL[][n][3])
TWDL[n][3]          : tournament final table
Main Procedure: validStatesGenerator(n)
1:  validStateGenerating(n, 0, TWDL)
Procedure: validStateGenerating(n, k, TWDL[][])
1:  if k≠n then
2:     maxP ←TWDL[k-1][0]*3+TWDL[k-1][1]
3:  else
4:     maxP ←3*n-3
5:  if k≠n-1then
6:     minP ←1
7:  else
8:     minP ←0
9:  for P ←maxP to minP do
10:    calculateWDLvalues(n, k, TWDL, P)
Procedure: calculateWDLvalues(n, k, TWDL[][], P)
1:  w ←⌊P/3⌋
2:  d ←P-3*w
3:  while w ≥0and w+d ≤ n-1do
4:     TWDL[k][0] ←w
5:     TWDL[k][1] ←d
6:     TWDL[k][2] ←n-1-w-d
7:     if k+1 <n then
8:        validStateGenerating(n, k+1, TWDL)
9:     else
10:       if the current state of TWDL is valid then
11:          sort TWDL into STWDL
12:    w ←w-1
13:    d ←P-3*w
```

A T_{WDL} state is considered to be valid if it leads to at least one case of the games results for a tournament, otherwise, it can not be valid. The following algorithm checks the possibility of finding a case of the games results based on a T_{WDL}state:

T_{WDL} 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
```

Computation Results

The proposed approach is programmed using the Java programming language, compiled by JDK 1.6 compiler and ran on Windows 7 Ultimate 64-bit operating system. A set of tournaments with the number of teams ranging between two and eight is examined during our experiments, where each of the numbers of the valid states, invalid states and the execution time are measured. The machine on which our experiments are performed 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.. The obtained experimental results are shown in Table 1.

Table 2 shows the number of the checked invalid states in the case of our proposed algorithm and the ones which must be checked during the blind search algorithm. Comparing with the blind search algorithm, we find that the proposed approach reduces the number of the checked invalid states up to 99.99% there by considerably optimizing the execution time especially when the number of teams gets bigger.

Table 1. Experimental results

n	Number of valid states	Number of invalid states	Execution time
2	2	3	0
3	7	43	0
4	40	698	0
5	404	13514	0
6	6317	308191	2 seconds
7	131288	8582812	4 minutes
8	3366444	275675646	54 hours

Table 2. The invalid states in our proposed algorithm versus those in the blind search algorithm

n	The proposed algorithm	Blind search algorithm
2	3	7
3	43	209
4	698	9960
5	13514	758971
6	308191	85759804
7	8582812	13492797224
8	275675646	2821106541012

Conclusion

This study focuses on generating the possible final table states of football tournaments, using the final points of the teams. Comparing with a blind search algorithm, an optimized search space which is based on the final points of the teams is proposed. For the proposed search space, an algorithm is developed to calculate the possible *WDL* values of each team from its final number of points. Also, an algorithm to check the validity of each of the generated states is presented. The proposed approach is expermented with a set of tournaments with number of teams ranging between two and eight. The results of the experiments are discussed, where our proposed approach proves its effectiveness compared to the blind search algorithm regarding the number of the generated invalid states.

The scope of the study can be extended to cover a competitive sports league environment. Since the position of a team in the final table is the main reason of its qualification to the next stage or elimination from a tournament, generating the states of the tournament's final table will be helpful in predicting the chances of a team to gain a specific position at the end of the tournament based on the current state of the table.

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

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

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

[11] 2006 FIFA World Cup Germany. FIFA. http://www.fifa.com/worldcup/archive/germany2006/groups/index.html. Accessed 22 January 2018