A MODIFIED ALGORITHM FOR RANKING PLAYERS OF A ROUND-ROBIN TOURNAMENT

AVIJIT DATTA, MOAZZEM HOSSAIN AND M. KAYKOBAD

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka-1000, Bangladesh.

e-mail: **kaykobad@cse.buet.ac.bd , datta_avijit@hotmail.com**

Abstract: The problem of ranking players in a round-robin tournament, in which outcome of any match is a win or a loss, is to rank players according to their performances in the tournament. In this paper we have improved previously developed MST (Majority Spanning Tree) algorithm for solving this problem, where the number of violations has been chosen as the criterion of optimality. We have compared the performance of our algorithm with that of MST algorithm.

Keyword: Ranking; Round-robin tournament; Upset; Digraph

C.R. Category: F2.2

1. INTRODUCTION

The problem of ranking players in a tournament has been the subject of various research investigations. This tournament structure also arises in other environment, for example, the problems of soliciting customer preferences regarding a set of products, establishing funding priorities of a set of projects [5], establishing searching priorities for a set of search engines in the internet. It is known that the results of a tournament can be represented in a digraph, G=(V, A) known as tournament graph, where vertices correspond to players and

arcs correspond to match results. A tournament result is said to be upset (or violation) if a lowly-ranked player has defeated a highly-ranked player. Ali [1], Cook [2] Goddard [4], Poljak [6] and many others have concentrated on the problem of determining ranks based on the results of the tournament. A constructive lower bound on the tournament ranking function was obtained in [7]. In [5], a heuristic solution to optimize the number of violations has been developed. This paper presents a modified version of that algorithm which reduces the time and space complexity significantly and gives the same output. The problem of minimizing the number of upsets is equivalent to finding the minimum number of arcs in a digraph deletion of which results in an acyclic digraph. This problem is known as Minimum Feedback Arc set Problem, and is NP-hard for general digraphs [1]. A fast and effective heuristic for the Feedback Arcset Problem has been developed in [3].

2. **THE ALGORITHM**

In this section we propose modification to MST algorithm that results in improvement of both space and time complexity compared to the MST algorithm for ranking players in a round-robin tournament [5]. Before describing the Modified_MST algorithm, we present here a brief discussion on MST algorithm.

MST: For ease of discussion we recapitulate some of the definitions used in MST algorithm.

1. cutset (i, k, j) – is the difference between the numbers of outgoing arcs from set (i, k) to set (k+1, j) and outgoing arcs from set (k+1, j) to set (i, k),
   where set (i, k) is the set of vertices corresponding to players ranked from i to k.
2. maxwin (i, j) – is the maximum number of wins of a player in set (i, j).
3. n – is the number of players in the tournament.
4. SetSwap (i, k, j) – Swaps the set (i, k) with set (k+1, j).
5. UpdateMaxWin() – updates the maxwin (i, j) whenever the SetSwap (i, k, j) is done.
6. UpdateCutSet() – updates the cutest (i, k, j) whenever the SetSwap (i, k, j) is done and it does this in the following incremental approach :

   cutset(i,k,j) = cutset(i,k-1,j) + sum

   where,

$$sum = \sum_{l=k+1}^{j} Arcs\ (k, l) - \sum_{l=i}^{k-1} Arcs\ (l, k)$$

$$Arcs(k, l) = \begin{cases} 1; & \text{if player ranked 'k' defeats player ranked 'l'} \\ -1; & \text{if player ranked 'l' defeats player ranked 'k'} \\ 0; & \text{if } k = l \end{cases}$$

here each of, i, j and k loops is run at most *n* times, and computation of sum takes another O(n) complexity. So, the total complexity of UpdateCutSet() is $O(n^4)$.

Procedure: MST

```
repeat
      done = false
      for i =1 to n – 1 do
            for j = i +1 to n do
                  for k = i to j – 1 do
                        if cutset (i, k, j) < 0 then
                              done = true
                              SetSwap(i, k, j)
                              UpdateCutSet ()
                              UpdateMaxWin ()
                        else if cutset (i, k, j)  = 0 then
                              if( maxwin (i, k) < maxwin (k+1, j) then
                                    done = true
                                    SetSwap (i, k, j)
                                    UpdateCutSet ()
                                    UpdateMaxWin ()
                              End if
                        End if
                  End for ( k-loop )
            End for ( j-loop )
      End for ( i-loop )
Until not done
```

Assuming the number of players in the tournament to be n, complexity of the MST algorithm can be derived as follows: execution of UpdateCutSet() to update the cutset (i,k,j) value requires computation of at most $O(n^4)$ complexity. Execution of UpdateMaxWin() to compute the maxwin (i, j) value requires computation of at most $O(n^3)$ complexity. Each of the i, j and k-loop will be done at most n times for a single swap, which will reduce the number of violations by 1. So, the amount of computation is at most $O(n^4)$. Since there can be at most $O(n^2)$ violations initially, the algorithm requires at most $O(n^6)$. Space requirement of cutset (i,k,j) is $O(n^3)$.

Modified_MST: For modification we introduce the following functions:

1. kut (i, k) = cutset (i, k, n), the difference between the numbers of outgoing arcs from set (i, k) to set (k+1, n) and outgoing arcs from set (k+1, n) to set (i, k),
   where set (i, k) is the set of vertices corresponding to players ranked from i to k.
2. UpdateKut () – updates the kut (i, k) whenever the SetSwap (i, k, j) is done and it does this in the following incremental approach :

   kut(i,k) = kut(i,k-1) + sum

   where,

$$\text{sum} = \sum_{l=k+1}^{n} \text{Arcs}(k, l) - \sum_{l=i}^{k-1} \text{Arcs}(l, k)$$

$$\text{Arcs}(k, l) = \begin{cases} 1; & \text{if player ranked 'k' defeats player ranked 'l'} \\ -1; & \text{if player ranked 'l' defeats player ranked 'k'} \\ 0; & \text{if } k = l \end{cases}$$

each of i and k loops is run at most *n* times, and computation of sum takes another O(n) complexity. So, the total complexity of UpdateKut() is $O(n^3)$.

To derive the relationship between cutset (i, k, j) and kut (i, k), we consider the set (i, n) as a union of 3 disjoint sets: set (i, k), set (k+1, j) and set (j+1, n).

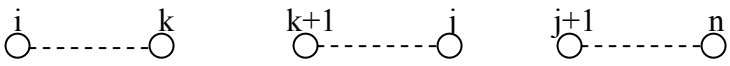
Let, A denote the set (i, k)

B denote the set (k+1, j)

C denote the set (j+1, n)

Then if we express AB = cutset (i,k,j)

We get, A(B+C) = cutset (i, k, n) = kut (i, k)



A B C

here, Cutset(i,k,j) = AB

= A(B+C) + BC - (A+B)C

= kut(i, k) + kut(k+1, j) - kut(i, j)

So cutset-values can be calculated through kut-values; thus reducing both space requirements and computation overhead.

Procedure : Modified_MST

```
repeat
      done = false
      for i =1 to n – 1 do
            for j = i +1 to n do
                  for k = i to j – 1 do
                        if kut(i,k) + kut(k+1,j) – kut(i,j) < 0 then
                                    done = true
                                    SetSwap (i, k, j)
                                    UpdateKut ()
                                    UpdateMaxWin ()
                        else if kut(i,k) + kut(k+1,j) – kut(i,j) = 0 then
                              if( maxwin(i,k) < maxwin(k+1,j)  then
                                    done = true
                                    SetSwap (i, k, j)
                                    UpdateKut ()
                                    UpdateMaxWin ()
                              End if
                        End if
                  End for ( k-loop )
            End for ( j-loop )
      End for ( i-loop )
Until not done
```

Assuming the number of players in the tournament to be n, complexity of the MST algorithm can be derived as follows: execution of UpdateKut() to update the kut (i,k) value requires computation of at most $O(n^3)$ complexity. Execution of UpdateMaxWin() to compute the maxwin(i, j) value requires computation of at most $O(n^3)$ complexity. Each of the i, j and k-loop will be done at most n times for a single swap, which will reduce the number of violations by at least 1. So, the amount of computation is at most $O(n^3)$. Since there can be at most $O(n^2)$ violations initially, the algorithm requires at most $O(n^5)$ computation. Space requirement of kut (i,k) is $O(n^2)$.

3. EXPERIMENTAL RESULTS

The Modified_MST Algorithm has been compared with MST Algorithm on the basis of a set of randomly generated tournaments of sizes ranging from 20 to 200 players. All heuristics have been programmed in C and runs were made on a Pentium III. Performances of the heuristics have been measured both in terms of violations and computational time. Here both the MST and Modified_MST return the same result with respect to number of violations.

FIGURE 1 Graphical Comparison between MST and Modified_MST

TABLE I Comparison between MST and Modified_MST

n	Number of initial upsets	Number of optimized upsets	Execution time for MST (ms)	Execution time for Modified_MST (ms)	% of savings in execution time
20	93	52	31	8	74
40	392	246	805	187	77
60	901	621	7116	1304	82
80	1579	1171	28304	4695	83
100	2492	1872	101648	14578	86
120	3621	2797	257812	34594	87
140	4836	3853	632844	73296	88
160	6460	5153	1298786	127650	90
180	8013	6589	2682684	232395	91
200	9982	8225	4545292	436796	90

We have also done the experiment with initial ranking based on sorted position of the sports.

TABLE II Comparison between Sorted-MST and Sorted-Modified_MST

n	Number of initial upset	Number of optimized upset	Execution time for Sorted-MST (ms)	Execution time for Sorted-Modified_MST (ms)	% of savings in execution time
20	97	51	31	0	100
40	395	240	375	31	92
60	912	613	4984	187	96
80	1572	1157	20860	781	96
100	2478	1870	60687	1531	97
120	3621	2799	182609	3718	98
140	4836	3840	414000	7250	98
160	6460	5148	864710	14188	98
180	8013	6584	1687880	22828	99
200	9982	8228	3055800	35906	99

From Table 1 and Figure 1 it is clear that the performance of Modified_MST is better than that of MST. If in both cases ranking of players is initially done according to their number of wins; that is, players are sorted with respect to their performance then performance of both MST and Modified_MST improved. This finding has been listed in Table 2.

We have also performed regression analysis to establish an approximate relation between size of tournament and computational time. Taking the data for n = 50 … 200 and performing a linear regression, we got the following relations between computational time, t and size of tournament, n. Results have been presented in the following table:

TABLE III Regression Analysis

Algorithm	Corresponding Equation
MST	$t = 0.000001551966\, n^{5.407932304917}$
Modified_MST	$t = 0.000004816414\, n^{4.736315310870}$
Sorted-MST	$t = 0.000002017904\, n^{5.274688725106}$
Sorted-Modified_MST	$t = 0.000002801777\, n^{4.394431753903}$

4. CONCLUSION

Experimental results comply with the theoretical complexity analysis. It also shows how superimposition of MST or Modified_MST algorithms on the initial ranking based on sorting improves computational efforts significantly.

References

1. I. Ali, W.D.Cook, M.Kress, *On the minimum violations ranking of a tournament,* Mgmt Sci., 32, 660-674 (1986).

2. W.D. Cook, I.Golan, M. Kress, *Heuristics for ranking players in a round-robin tournaments,* Computers Ops. Res., Vol. 15, pp. 135-144, 1988.

3. P. Eades, X. Lin, W.F. Smyth, *A fast and effective heuristic for the feedback arcset problem* [http://citeseer.ist.psu.edu/156876.html]

4. S.T.Goddard, *Ranking tournament and group decision making*, Mgmt Sci. 29, 1384-1392 (1983).

5. M. Kaykobad, Q.N.U. Ahmed, A.T.M. Khalid, R. Bakhtiar, *A New Algorithm for ranking players of a round-robin tournament,* Computers Ops. Res., Vol. 22, No. 2, pp. 221-226, 1995.

6. S. Poljak, V.Rodl, J. Spencer, *Tournament ranking with expected profit in polynomial time*, SIAM Jl Disc. Math., No. 3 (1988).

7. S. Poljak, A. Czygrinowy, V. Rodl, *Constructive quasi-ramsey numbers and tournament ranking,* SIAM J. Discrete Math., Vol. 12, No.1, pp. 48-63.