Pergamon

0305-0548(94)E0024-2

Computers Ops Res. Vol. 22, No. 2, pp. 221–226, 1995

0305-0548/95 $7.00 + 0.00

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

MOHAMMAD KAYKOBAD,[1]†‡ Q. N. U. AHMED,[1]§ A. T. M. SHAFIQUL KHALID[1]¶ and REZWAN-AL BAKHTIAR[2]‖

[1]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh and [2]Onirbaan Group, Dhanmondi, Dhaka, Bangladesh

(Received June 1993; in revised form April 1994)

Scope and Purpose—The problem of ranking players in a round-robin tournament is to rank players according to their performance in the tournament. Similar problem also arises in soliciting consumer preferences regarding a set of products and establishing funding priorities. In this paper we have developed a new heuristic algorithm where the number of violations has been chosen as the criterion of optimality. Experimental results suggest that this algorithm outperforms the existing ones in solving the ranking problem.

Abstract—In this paper we present a new algorithm for solving the problem of ranking players in a round-robin tournament in which outcome of any match is a win or a loss. We also compare performance of our algorithm with that of ARRANGE and GIK algorithms for solving the same problem. Some theoretical properties of our algorithm have also been deduced.

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 environments, for example, in problems of soliciting consumer preferences regarding a set of products and establishing funding priorities for a set of projects [1]. It is known that the results of a tournament can be expressed in a diagraph $G=(V, A)$ known as *tournament digraph*, where vertices correspond to players, and arcs correspond to match results. A tournament result is said to be an *upset* (or, *violation*) if a lowly-ranked player has defeated a highly-ranked player. Goddard [2] among others has concentrated on the problem of determining ranks based on the results of a tournament. In Ref. [3] Ali *et al.* developed a heuristic known as *Iterated Kendall*, which provided a substantial improvement over the then existing procedures. They have also presented an enumeration algorithm for determining the minimum violations ranking. Recently in Ref. [1] Cook *et al.* have developed a new algorithm known as the *Generalized Iterated Kendall* (GIK), and it was shown that this algorithm outperforms algorithms, like *Hamiltonian Path* (HP), *Iterated Kendall* and ARRANGE in terms of number of violations. The HP algorithm finds a ranking of players such that a player always defeats a player ranked immediately below him, whereas ARRANGE algorithm starts with an arbitrary ranking and improves it by rearranging the ranking of a single player so that the total number of violations is

†To whom all correspondence should be addressed.

‡Dr M. Kaykobad is an Assistant Professor in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). He received his M.S. in Engineering Degree from OIME, Odessa, an M.Eng. Degree in Computer Applications Technology from AIT, Bangkok, and a Ph.D. Degree from The Flinders University of South Australia. He has published in the journals of *Economics and Management of Marine Transport, Linear Algebra and Its Application, Physical Review, Bangladesh Academy of Sciences, Bangladesh Mathematical Society, Bangladesh Computer Society*. His research interests are in the fields of design and analysis of algorithms, discrete optimization and graph theory.

§Q. N. U. Ahmed is a lecturer in the Department of Computer Science and Engineering. At present he is a graduate student in Japan. His interests lie in the area of design and analysis of algorithms.

¶A. T. M. Shafiqul Khalid is an undergraduate student in the Department of Computer Science and Engineering, BUET. His interests are in the fields of logic design, algorithms and combinatorial problems.

‖Rezwan-al Bakhtiar holds a B.Sc. E.E. Engineering Degree from Bangladesh University of Engineering and Technology. His interests lie in developing software packages and algorithms.

decreased. Algorithm GIK uses arrange on a good initial solution. For details of these algorithms the reader is referred to Ref. [1]. Poljak *et al.* [4] presented an algorithm of $O(n^3 \log n)$ which produces a ranking with at most $\frac{1}{2}(n/2)+c_1 n^{3/2}$ violations, where $c_1=\frac{1}{8}\pi^{-1/2}$. 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 the Minimum Feedback Arcset Problem, which is NP-hard [1]. This, in turn, virtually denies any possibility of existence of a good algorithm. For any problem of this complexity class since there is a computational explosion with the increase of the problem size, heuristic algorithms are used. However, quite often these heuristic algorithms are caught at a local minimum, and cannot come out. Sometimes, a significant amount of computational effort is needed to move out of such situation. In this paper we are introducing a new heuristic to be known as MST algorithm for solving this problem. This algorithm has the advantage of superimposability over the solution obtained by any other algorithm. This, together with some of its theoretical properties, makes it particularly useful in such situations. Section 2 is introductory, whereas in Section 3 we will develop the MST algorithm, and compare its performance with that of ARRANGE and GIK algorithms.

2. PRELIMINARIES

We consider only simple connected diagraphs $G=(V, A)$. Spanning trees of any digraph are denoted by T.

Definition 2.1. A directed cutset (V_i, V_j) is defined as

$$(V_i, V_j)=\{(k, l) \mid k \in V_i, l \in V_j\}$$

Definition 2.2. Weight $W\{(V_i, V_j)\}$ of a directed cutset (V_i, V_j) of a diagraph G is defined as

$$W\{(V_i, V_j)\}=\sum_{[k \in V_i, l \in V_j]} W_{kl}$$

where W_{kl} is the weight attached to the arc $(k, l) \in A$.

Definition 2.3. For each arc $(i, j) \in T$ a fundamental cutset $[V_i, V_j]$ is defined as $[V_i, V_j]=(V_i, V_j) \cup (V_j, V_i)$ where V_i and V_j are the vertices of the connected components of $T \setminus \{(i, j)\}$ containing vertices i and j respectively.

Definition 2.4. Weight of an undirected cutset $[V_i, V_j]$ is defined as

$$W\{[V_i, V_j]\}=W\{(V_i, V_j)\}-W\{(V_j, V_i)\}.$$

It may be noted here that although by our definition $[V_i, V_j]=[V_j, V_i]$ their weights are not equal excepting when they are 0.

Let us now introduce a new class of spanning trees to be called *Majority Spanning Trees* (MST).

Let $W: A \to R^A$ be a nonnegative weight function defined on the arc-set of a digraph $G=(V, A)$. Then

Definition 2.5. A spanning tree T of digraph $G=(V, A)$ is said to be a majority spanning tree if

$$\{(i, j) \in T\} \Rightarrow \{W\{[V_i, V_j] \geqslant 0\}$$

Now we present the following theorem proof of which can be found in Ref. [5].

Theorem 2.1. For every digraph $G=(V, A]$ and every non-negative weight function W there exists a majority spanning tree.

In the following section we present an algorithm for solving the problem of ranking players of a round-robin tournament using the concept of majority spanning trees.

3. MAJORITY SPANNING TREE ALGORITHM

We consider the problem of ranking players by the criterion of minimizing the number of violations (upsets). Let R be a ranking and $G_R=(V_R, A_R)$ be a subgraph of $G=(V, A)$ such that

$V_R = V$ and $A_R = \{(i, j)|$ rank of player corresponding to vertex j is immediately below player corresponding to vertex $i\}$. It is obvious that $G_R = (V_R, A_R)$ is a spanning tree of G, more accurately $G_R = (V_R, A_R)$ is a Hamiltonian semipath.

Theorem 3.1. Let R be any optimal ranking of a tournament represented by $G = (V, A)$. Then $G_R = (V_R, A_R)$ is an MST of G.

Proof: Suppose that $G_R = (V_R, A_R)$ does not correspond to an MST. Then there exists a violating cutset (a cutset is said to be violating if its weight is negative) $[V', V \backslash V']$. This means that the set of players corresponding to V' has lost more games to the remaining players than they have won from them. Therefore, if we rank players corresponding to $V \backslash V'$ first, without changing their relative ranking, and then rank players corresponding to V', the number of upsets will be decreased. Therefore, the result is true.

Let $G^{ij}(R)$ be the subgraph of G induced by the set of vertices corresponding to players ranked from i to j, and let G_R^{ij} be the subgraph of $G^{ij}(R)$ having the same set of vertices, and only those arcs that correspond to the results of matches between consecutively ranked players. Then

Theorem 3.2. For any optimal ranking R and $1 \leqslant i \leqslant j \leqslant n$, G_R^{ij} must be an MST of $G^{ij}(R)$.

We note here that if G_R^{ij} is not an MST of the associated subdigraph, then again the solution can be improved by the same arguments as in the proof of Theorem 3.1.

The algorithm MST finds a G_R^{ij} that is an MST of G. If it is not so, then the ranking can be improved by swapping the set of consecutively ranked players as has been noted in the proof of Theorem 3.1. A systematic search for a violating cutset is carried out by this algorithm through choosing subdigraphs induced by consecutively ranked players, and then checking all its possible cutsets. If no more violating cutset can be found for any of the subdigraphs $G^{ij}(R)$ for all possible values of i, j, the MST algorithm stops. However, the quality of solution can still be improved, as is done in ARRANGE, by swapping players around a cut-set with 0 weight, which will not deteriorate the current solution.

From Theorem 3.2 we have the following corollaries relating the properties of the MST algorithm with those of HP and ARRANGE algorithms.

Corollary 3.1. For any ranking R obtained by the MST algorithm, G_R^{ij} is a directed Hamiltonian Path of $G^{ij}(R)$ for $1 \leqslant i \leqslant j \leqslant n$.

Corollary 3.2. For any ranking R obtained by the MST algorithm, R cannot be improved by applying Hamiltonian Path or ARRANGE algorithms.

By virtue of Corollary 3.1 MST ranking cannot be improved by the HP algorithm, whereas if a ranking can be improved by ARRANGE algorithm then one can still find a violating cutset for the MST algorithm to get an improved solution. Hence ARRANGE algorithm cannot improve any solution obtained by the MST algorithm.

Below we give some explanations of what the algorithm does, and the functions that are used in the 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, j), where set (i, j) is the set of vertices corresponding to players ranked from i to j.
2. *maxwin(i, j)* —is the maximum number of wins of a player in set (i, j).
3. *pair(i, j)* —corresponds to an upset if the player ranked j defeats the player ranked i.
4. *size* —is the number of players in the tournament.

In the algorithm i-loop selects the first vertex of $G^{ij}(R)$, j-loop selects the last vertex of $G^{ij}(R)$, whereas k-loop selects the position of cutset. In case of the absence of any violating cutset, players, around a cutset with zero weight, are swapped, just as is done in ARRANGE. However, if even such a swapping does not guarantee an improvement of the solution the subset of players containing player corresponding to *maxwin*(i, j) is given better ranking by swapping.

Majority (MST)

```
repeat
  swap ← false
  for i = 1 to size − 1 do
    for j = i + 1 to size do
      for k = i to j − 1 do
        if cutset(i, k, j) < 0
          swap ← true
        elseif cutset(i, k, j) = 0 then
          if pair(i, j) or (i − 1, k + 1) or (k, j + 1) is upset then
            swap ← true
            swap respective pair
          else
            if maxwin(i, k) < maxwin(k + 1, j)
              swap ← true
              swap respective pair
            endif
          endif
        endif
        if swap = true then
          swap set ({i, k}, {k + 1, j})
          break i-loop
        endif
      endfor (k-loop)
    endfor (j-loop)
  endfor (i-loop)
until not swap
```

Assuming the number of players in the tournament to be n, complexity of the MST algorithm can be derived as follows: In the k-loop, calculation of cutset value requires $O(n)$ operations. 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. The amount of computation for this is at most $O(n^4)$. Since there can be at most $O(n^2)$ violations initially, the algorithm requires at most $O(n^6)$ calculations. If we permit $O(n^3)$ space for storing the values of *cutset*(i, k, j), then values of next cutsets can be calculated from the previous ones using $O(1)$ computation, which makes the algorithm $O(n^5)$. Our statistical experiments also suggest that average computational time is of lower order.

4. EXPERIMENTAL RESULTS

For comparing performance, three algorithms, namely ARRANGE, GIK and MST have been considered. The basis for comparison of the above-mentioned heuristics is a set of randomly generated tournaments of sizes ranging from 15 to 100 players. All heuristics have been programmed in C and runs were made on 80486 machines. A Random number generator available with TURBO C has been used. Performance of the heuristics has been measured both in terms of violations and computational time. In order to obtain better statistics on computational time, a group of 5 examples

Table 1. Computational time required by different heuristics

Size	ZRRANGE		GIK		MST	
15	0.024	0.141	0.006	0.01	0.39	0.17
20	0.06	0.16	0.03	0.01	0.10	0.21
30	0.33	0.172	0.14	0.05	0.575	0.34
40	1.20	0.21	0.53	0.12	3.76	0.86
50	3.14	0.25	1.33	0.17	9.70	1.94
60	6.75	0.28	3.08	0.31	24.82	4.10
70	14.68	0.74	6.27	0.48	63.44	10.85
80	28.81	1.20	11.23	0.57	112.36	27.50
100	183.36	12.71	30.90	2.90	321.40	76.8

have been solved before both these statistics have been noted. For each size 10 groups of problems have been run to obtain values of statistical parameters. Average numbers of violations for random ranking and ranking based on sorted scores have been computed. Then each algorithm has been run on the initial ranking obtained from sorted score on all the randomly generated tournaments. Statistics on both computational times and number of violations have been kept. These statistics together with standard deviations of these quantities have been shown in the following tables.

In Table 1 the first entry in each cell corresponds to average computational time (in seconds), whereas the second one is standard deviation. In Table 2 the corresponding figures of number of

Table 2. Number of violations for different heuristics

	Number of violations				
Size	Before sorting	After sorting	ARRANGE	GIK	MST
15	54.00	38.48	26.04	26.34	25.84
	3.22	3.34	2.27	1.91	2.14
20	95.16	69.98	50.94	52.50	50.72
	4.65	3.60	2.37	2.41	2.22
30	217.22	170.86	132.62	134.90	131.66
	4.60	4.90	2.39	2.49	2.40
40	389.46	321.20	254.40	256.28	251.80
	8.69	6.32	2.70	3.42	2.62
50	613.12	522.12	422.12	424.80	419.72
	8.61	6.42	3.79	4.04	3.95
60	884.70	753.46	624.60	629.18	620.7
	10.26	9.96	5.86	5.70	5.08
70	1208.18	1068.64	875.74	879.38	849.98
	12.74	10.71	5.74	4.19	4.96
80	1577.63	1459.60	1171.86	1177.50	1165.53
	18.57	12.61	6.09	6.69	4.52
100	2474.00	2360.25	1894.35	1899.45	1887.85
	32.87	14.90	12.33	18.42	12.95

Table 3. Performance of heuristics (figures are in percentages)

Size		GIK		MST		1st	2nd	3rd
15	ARRANGE	16	38	20	0	74	26	0
	GIK	0	8	26	0	52	48	0
	MST	6	38	0	0	94	8	0
20	ARRANGE	10	40	14	0	80	16	4
	GIK	0	8	30	0	46	54	0
	MST	2	46	0	0	96	4	0
30	ARRANGE	24	70	48	0	42	44	14
	GIK	10	14	58	0	18	48	34
	MST	12	80	0	0	82	18	0
40	ARRANGE	36	56	76	0	12	60	28
	GIK	22	16	70	0	14	38	48
	MST	16	74	0	0	84	14	2
50	ARRANGE	32	64	74	0	16	60	24
	GIK	18	22	68	0	20	32	48
	MST	20	76	0	0	76	24	0
60	ARRANGE	22	72	86	0	10	68	22
	GIK	32	36	90	0	8	26	66
	MST	6	90	0	0	88	10	2
70	ARRANGE	42	54	92	0	2	60	38
	GIK	26	50	82	0	20	28	52
	MST	22	78	0	0	78	22	0
80	ARRANGE	22	78	92	0	6	78	16
	GIK	48	42	86	0	12	16	72
	MST	8	92	0	0	90	10	0
100	ARRANGE	20	60	92	0	0	70	30
	GIK	50	45	92	0	15	25	60
	MST	15	75	0	0	85	15	0

violations have been placed one under the other. In Table 3 we compare relative performances of the heuristics in terms of percentage of times each of them produced best, second best and the third best solutions. We have also superimposed each of these algorithms over all others in order to compare their superimposability. In this case column algorithms are superimposed on row algorithms. The columns with headings GIK and MST contain two entries in each cell; the first one shows the percentage of times improvement has been obtained, whereas the second figure shows the percentage of time the initial solution has been deteriorated by the superimposition.

Table 2 shows that in all the 9 sizes ARRANGE has outperformed GIK in terms of number of violations. This was not the case when we started with a random ranking instead of ranking on the sorted score. It is our observation that since GIK runs ARRANGE on a good initial solution for a random initial ranking most often it outperforms ARRANGE. However, when initial ranking is a good one, as can be obtained by ranking players according to score, it loses its superiority. Amount of computational time required for GIK is less than that of ARRANGE, which also indicates that it cannot rearrange the initial data in a more potential way and gets stuck to some local optimal solution quickly. Table 3 shows that most of the time the MST algorithm improves solutions obtained by other algorithms, and in no case does it deteriorate the initial solution, which is not the case with the GIK algorithm.

Acknowledgements- We would like to thank Professor Shamsher Ali, Vice Chancellor of Bangladesh Open University, Dhaka and Dr F. J. M. Salzborn of the University of Adelaide. We thank the referees of an earlier version whose valuable comments have enriched presentation of our results.

REFERENCES

1. W. D. Cook, I. Golan and M. Kress, Heuristics for ranking players in a round-robin tougnament. *Computers Ops Res.* **15**, 135–144 (1988).
2. S. T. Goddard, Ranking tournaments and group decision making. *Mgmt Sci.* **29**, 1384–1392 (1983).
3. I. Ali, W. D. Cook and M. Kress, On the minimum violations ranking of a tournament. *Mgmt Sci.* **32**, 660–674 (1986).
4. S. Poljak, V. Rodl and J. Spencer, Tournament ranking with expected profit in polynomial time, *SIAM Jl Disc. Math.* **1**, No. 3 (1988).
5. M. Kaykobad, Minimum Connection Time and Some Related Complexity Problems. Unpublished Ph.D. Thesis, The Flinders University of South Australia (1986).