An Improvement to MST Algorithm for Round-Robin Tournament Ranking

M Kaykobad, Pradipta Prometheus Mitra\textsuperscript{1} and Irwin King\textsuperscript{2}
\textsuperscript{1}Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh, email: mkbd1234@yahoo.com
\textsuperscript{2}Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong, email: king@cse.cuhk.edu.hk

Abstract - This paper improves the complexity of a newly introduced MST (Majority Spanning Tree) algorithm for ranking players of a round-robin tournament previously published.

Keywords: Ranking, Algorithm, Graphs.

1. INTRODUCTION

The problem of ranking players in a round-robin tournament has been investigated for many different areas of applications. This tournament structure arises in problems of soliciting consumer preferences regarding a set of products and establishing funding priorities for a set of competing projects [2]. Recently with the introduction of internet and its growing popularity, it is becoming all the time more important for search engines to present documents against a search in order of its desirability, again indicating necessity of ranking a set of documents based upon their desirability. Now search engines are capable of coming up with thousands of documents from billions in seconds. But it is impossible for the users to look into even a few hundred, thus making the retrieval operation largely redundant. It has been thought of evoking several search engines for the same search request and then to rank the retrieved documents giving consideration to their ranking by different search engines.

It is known that the results of a tournament can be expressed in a digraph $G=(V,A)$, 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 [3] among others has concentrated on the problem of ranking based upon the results of a tournament. Ali et al. [1] developed a heuristic known as Iterated Kendall, which provided a substantial improvement over the then existing procedures. Recently Cook et al. [2] 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 decreased. Algorithm GIK uses ARRANGE on a good initial solution. For details of these algorithms the reader is referred to [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 known to be NP-hard in general [2]. This, in turn, virtually denies any possibility of existence of a good algorithm. As a result only alternative is to try with heuristic algorithms, which are inevitably caught at a local optimal solution. Sometimes, a significant amount of computational efforts are needed to move out of such a situation. In [5] Kaykobad et al introduced a new heuristic algorithm, called MST algorithm for solving this problem. This algorithm enjoys 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. It was reported that this algorithm outperformed all the existing algorithms in terms of quality of solution. In the following section we introduce some definitions and describe some theoretical results that are combined to obtain MST algorithm. In section 3 we present the MST algorithm and its complexity. In section 4 we show how this algorithm can be improved to obtain an algorithm of lower complexity.
2. MST ALGORITHM: SOME PROPERTIES

We consider only simple connected digraphs \( G=(V,A) \). Spanning trees of underlying graphs are denoted by \( T \). Each edge of a spanning tree uniquely partitions the vertex set, and corresponds to a fundamental cutset that contains all the edges running from one subset of vertices to another and vice versa.

Definition 2.1
Weight of a cutset defined by edge \((i,j)\in A\) is defined as weights of edges of cutset in the orientation of \((i,j)\) minus the weights of edges in the cutset in the opposite orientation.

Definition 2.2
A spanning tree of a digraph \( G=(V,A) \) is said to be a majority spanning tree if weight of each fundamental cutset defined by an edge of the spanning tree is non-negative.

The following theorem can be found in [4].

Theorem 2.1
For every digraph \( G=(V,A) \) and every non-negative weight function defined on the edges there exists a majority spanning tree.

3. THE MST ALGORITHM

We consider the problem of ranking players of a round robin tournament by minimizing the number of violations or 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 \( A = \{(i,j)\mid \text{rank of player corresponding to vertex } j \text{ is immediately below player corresponding to vertex } i\} \). It is obvious that \( G_R \) is a spanning tree.

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

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

Theorem 3.2
For any optimal ranking \( R \) and \( 1 \leq i \leq j \leq n \), \( G^{(i,j)}_R \) must be a MST of \( G^{(i)}(R) \).

4. IMPROVEMENT OF MST ALGORITHM

In the algorithm below the following definitions have been used.

\( \text{cut}(i,k,j) \) - is the difference between the number 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 \( k \).

\( \text{maxwin}(i,j) \) - is the maximum number of wins of a player in set \((i,j)\).

\( \text{pair}(i,j) \) - corresponds to an upset if the player ranked \( j \) defeats the player ranked \( i \).

\( \text{size} \) - is the number of players in the tournament.

Majority_Spanning_Tree

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

Assuming the number of players 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 \( k \) loops 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. For values of \( n \) exceeding thousand the calculation is definitely prohibitive if we want to apply it in the internet application for ranking documents obtained by different search engines.

O\(n^4\) computation may be needed. Since reduction of number of upsets can be O\(n^2\) the worst case complexity of the algorithm is O\(n^6\).

Now let us define the following special cutset with value Kut\((i,j)\)=cutset\((1,i,j)\). Now it can be verified that

\[
\text{cutset}(i,j,k) = \text{Kut}(k,j) - \text{Kut}(i-1,j) + \text{Kut}(i-1,k),
\]

Where, Kut\((0,j)\) is assumed to be 0 for all \(j\).

This inequality allows us to compute only Kut values and derive cutset values from there by only two additions/subtractions. We can start the algorithm by computing the Kut values initially and recompute them with every swapping. Since number of Kut values is O\(n^2\), and computing each of them will require O\(n\) computational effort, so each recomputation of Kut values will require O\(n^3\) which can be repeated at most O\(n^3\) times thus resulting in O\(n^5\) algorithm. Avoiding recalculation of Kut values that remain unaffected can also further minimize the computation. The quality of solutions in terms of number of violations remains as before but order of computation goes down by 1 thus producing ranking in lesser efforts which is important in case of presenting huge search results according to priority and thus make the efforts of search engine more useful.

5. CONCLUSION

While the present modification reduces the complexity of the MST algorithm, for purposes of Internet application where thousands of documents need to be ranked after search engines have retrieved them the order of the algorithm must be lowered further in order for its effective use. However, an online algorithm can always rank a small subset of the documents and produce them to the user in acceptable time. While the user is busy looking into these documents the algorithm can continue to rank the remaining documents and produce the next subset for the user if he/she is interested to go further.

Acknowledgement- One of the authors gratefully thanks his PhD supervisor Dr F.J.M. Salzborn of the University of Adelaide.

REFERENCES


