Journal of Theoretical and Applied Information Technology
30th November 2012. Vol. 45 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A TABLE-FILLING SCHEDULING ALGORITHM FOR THE ROUND ROBIN CALENDAR PROBLEM WITH ARBITRARY COMPETITORS

[1]JIANYONG LI, [1]DAOYING HUANG, [2]BING HAN, [1]ANLIN ZHANG

[1] School of Computer & Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou, Henan, 450002, China

[2] Software Professional 2009 grade, School of Software at BIT, Beijing, 100081, China

ABSTRACT

A novel table-filling schedule algorithm is proposed for the round robin calendar problem with arbitrary competitors. Among schedules calendar, the number of rows is the ID of each competitor and elements of the schedule correspond to a certain round serial number of some two competitors. Competition schedule can be completed by the algorithm when the number of competitors is even. If the number of competitors is odd, the schedule will be filled through three steps, even transforming, table filling and even eliminating. It is proved that time and space complexity of the algorithm are all $O(n^2)$.

Keywords: *Round Robin Scheduling, Table-filling Algorithm, Complexity*

1 INTRODUCTION

Round robin scheduling algorithm can be traced back to the timetable arrangement of the tournament with multiple competitors. Limited by the number of venues or the increasing requirement for the spectator of sports, not all the competitors will compete with others during one round. So far, the dominant researches of the tournament scheduling mainly focus on how the athlete participating time effected on the physical distributions and the fairness[1-4]. Particularly, the fairness index system modeling and algorithm design is the most widely studied issue1. Foreign scholars focused most of their attention on the resource scheduling of computer operating system and the scheduling of the network routers and the data transformation[5-11].

Under the assumption that in each round, all the competitors participate in the competition, the round robin scheduling problem is actually the problem of how to complete the tournament in the shortest time, i.e., the minimum number of required round. Wang et al[12] provided a formal description of the problem and put forward a divide-and-conquer based algorithm for the timetable arrangement with 2k competitors. Cheng et al[13] verified the correctness of the algorithm. For the non-2k player round robin scheduling problem, Liu et al[14] proposed an extended algorithm. Luo et al[15] proposed an alternative algorithm by the means of line arranged transformation. For the problem with 2t athletes, the graph theory was used to construct the scheduling timetable[16, 17]. Li et al[18] put forward a novel divide-and-conquer algorithm to tackling the overlap problem in non-2k competitors scheduling.

When taking into consideration of the algorithm complexity, it is obvious that a possible simplest algorithm should guarantee the tournament complete in $O(n^2)$. In this paper, we consider the complexity of the algorithm and put forward a table-filling round robin algorithm. We also prove that the time and space complexity of the algorithm is less than $O(n^2)$.

2 EXAMPLE OF DIVIDE AND CONQUER BASED ROUND ROBIN ALGORITHM

We first provide an example for the divide-and-conquer based round robin scheduling for 2^k and non- 2^k competitors respectively.

2.1 Divide And Conquer Scheme With 2^k Competitors

Let $k = 3$, according to [12], we can get the scheduling timetable, as shown in Figure 1.

Journal of Theoretical and Applied Information Technology
30th November 2012. Vol. 45 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Competitor Number	Day Number						
	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	1	4	3	6	5	8	7
3	4	1	2	7	8	5	6
4	3	2	1	8	7	6	5
5	6	7	8	1	2	3	4
6	5	8	7	2	1	4	3
7	8	5	6	3	4	1	2
8	7	6	5	4	3	2	1

Figure 1: The Time Scheduling of 8 Competitors

2.2 Divide-And-Conquer Scheme With Non-2^k Competitors

According to [18], we can get the time schedule for 6 competitors (non-2^k competitors) shown in Figure 2.

Competitor Number	Day Number				
	1	2	3	4	5
1	2	3	4	5	6
2	1	5	5	6	4
3	6	1	6	4	5
4	5	6	1	2	3
5	4	2	2	3	1
6	3	4	3	1	2

Figure 2: The Time Scheduling of 6 Competitors

3 A TABLE-FILLING ALGORITHM FOR THE ROUND ROBIN SCHEDULE

3.1 Problem Description

Suppose there are n competitors in a round robin match and the match courts are adequate. We want to design a match schedule to meet the following requirements:

1) Each competitor must have a match with all the other competitors;

2) Each competitor has only one competition every day;

3) If n is even, the match must be completed in $n-1$ days. If n is odd, the match must be completed in n days.

3.2 Problem Reformulation

For a given matrix A, denote the elements as $a[i, j]$, where i is the index number of the team and j is the index number of the competition team. Then the round robin scheduling problem for arbitrary competitor n can be reformulated as the table-filling problem of a $n \times n$ matrix. To be detailed, the problem can be reformulated as follows:

1) The elements in each row of the matrix cannot have duplicate values,

2) The elements in each column of the matrix cannot have duplicate values;

3) If n is even, the available filling number is $0 \sim n-1$. If n is odd, the available filling number is $0 \sim n$, where $a[i, j] = 0$ means that there is no compete between i and j team.

3.3 The Feasibility Of The Solution

Form [12-15, 18] we can see that there exist solutions for the round robin scheduling problem of 2^k and non-2^k competitors. Matrix A can be transformed from the time schedule, and there must be exists a solution for the round robin table-filling algorithm.

3.4 Table-Filling Algorithm

Let the element $a[i, j] = k$, which means that the competition between the i^{th} team and the j^{th} team will be arranged on the k^{th} day. $k = 0$ means that there is no competition between the i^{th} team and the j^{th} team.

The table-filling algorithm can be described as follows:

1) Fill the first line with use $0 \sim n-1$;

2) From the 2nd line, let $a[i, j] = a[i-1, j+1]$ $(i = 2, ..., n; j = 1, ..., n-1)$. For the last element in each row, let $a[i, n] = a[1, i]$;

3) Modify the elements in the last row and last line as following: $a[i, n] = a[i, i]$, $a[n, i] = a[i, i]$, where $i = 2, ..., n$;

4) Fill the diagonal blocks with 0.

3.5 The Pseudo-Code Description Of The Table-Filling Algorithm

The algorithm can be realized by the pseudo code shown in figure 3.

Journal of Theoretical and Applied Information Technology
30th November 2012. Vol. 45 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
for (int j = 1; j <= n)
{
//Algorithm 1)
      a[1][j] = j - 1;
}
//Algorithm 2)
for (int i = 2; i <= n; i ++)
{
      for (int j = 1; j <= n - 1; j ++)
      {
            a[i][j] = a[i - 1][j + 1];
      }
      a[i][j] = a[1][i];
}
// //Algorithm 3)
for (int i = 2; i <= n - 1, i + +)
{
      a[i][n] = a[i][i];
      a[n][i] = a[i][i];
}
//Algorithm 4)
for (int i = 2; i <= n - 1, i + +)
      a[i][i] = 0;
```

Figure 3: Pseudo Code Table-Filling Algorithm

3.6 Example

Let $n = 8$.

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	1
2	3	4	5	6	7	1	2
3	4	5	6	7	1	2	3
4	5	6	7	1	2	3	4
5	6	7	1	2	3	4	5
6	7	1	2	3	4	5	6
7	1	2	3	4	5	6	7

Figure 4: The Results After Executed Step 1) And 2)

According to the table-filling algorithm, the results after Step 1) and 2) can be shown in Figure 4.

The results after executed step 3) are shown in Figure 5.

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	2
2	3	4	5	6	7	1	4
3	4	5	6	7	1	2	6
4	5	6	7	1	2	3	1
5	6	7	1	2	3	4	3
6	7	1	2	3	4	5	5
7	2	4	6	1	3	5	7

Figure 5: The Results After Executed Step 3)

The results after executed step 4) are shown in Figure 6.

0	1	2	3	4	5	6	7
1	0	3	4	5	6	7	2
2	3	0	5	6	7	1	4
3	4	5	0	7	1	2	6
4	5	6	7	0	2	3	1
5	6	7	1	2	0	4	3
6	7	1	2	3	4	0	5
7	2	4	6	1	3	5	0

Figure.6: The Results After Executed Step 4)

Let $n = 6$. According to the table-filling algorithm, the results are shown in Figure 7.

0	1	2	3	4	5
1	0	3	4	5	2
2	3	0	5	1	4
3	4	5	0	2	1
4	5	1	2	0	3
5	2	4	1	3	0

Figure.7: The Table-Filling Algorithm Results For 6 Competitors

3.7 Conversion Form Filling Algorithm Results

In order to get the full time scheduling and also to give a comparison with the conventional divide-and-conquer based algorithm, we transform the results to a scheduling table similar to Figure1 and Figure2.

Journal of Theoretical and Applied Information Technology
30th November 2012. Vol. 45 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

First, according to Figure 6, the matrix can be transformed to Figure 8, where the competitor number is eight.

Competitor Number	Day Number						
	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	1	8	3	4	5	6	7
3	7	1	2	8	4	5	6
4	6	7	1	2	3	8	5
5	8	6	7	1	2	3	4
6	4	5	8	7	1	2	3
7	3	4	5	6	8	1	2
8	5	2	6	3	7	4	1

Figure 8: Eight Competitors Filling Algorithm Program

According to Figure 7, the matrix results for the scheduling of 6 competitors can be transformed to Figure 9.

Competitor Number	Day Number				
	1	2	3	4	5
1	2	3	4	5	6
2	1	6	3	4	5
3	5	1	2	6	4
4	6	5	1	2	3
5	3	4	6	1	2
6	4	2	5	3	1

Figure 9: Six Competitors Filling Algorithm Program

4 CORRECTNESS VERIFICATION

4.1 Matrix Decomposition And Combination

Denote the shaded triangular sub-matrix in the matrix shown in Figure 3, as A_1 and the non-shaded part as A_2. From the proposed table-filling approach, in each row and column of A_1 and A_2, there are no repeated elements. So both A_1 and A_2 satisfy the demand of the scheduling requirement.

Now combine the triangular matrix A_1 and A_2. For the triangular matrix A_1, the i^{th} round corresponds to the $1 \sim i+1$ line; for the triangular matrix A_2, the i^{th} round corresponds to the $i+1 \sim n$ line. After the combination of matrix A_1 and A_2, the i^{th} round repeated only one time in the $i+1$ line.

4.2 Elimination Of The Repeated Rounds

It is obvious that the only possible repeat of each round appears in the $(i+1)^{th}$ line. In the $(i+1)^{th}$ line, $a[i+1, i+1]$ indicates the competition round number between the $(i+1)^{th}$ competitors in the two teams. According to the principle of the scheduling, the number should be 0. Exchange the elements $a[i+1, n]$ and $a[i+1, i+1]$. Let the round value in the diagonal line as 0. Because 0 never appears, this treatment will not result in the duplication of the elements of any row.

After the above combination step, in each row of matrix A, there are no duplicated elements. Further noting that it is a symmetric matrix, there are also no repeating elements in each column. So matrix A satisfies the demand.

4.3 Construction Of Odd-Scale Matrix

When n is odd, we can fill the matrix with and ensure that there are no repeating elements in each row and each column. Thus we can get the round robin schedule of $n+1$ team. Remove the elements in the last row and the last column, and we can get the time schedule of the n teams.

0	1	2	3	4	5	6	7
1	0	3	4	5	6	7	2
2	3	0	5	6	7	1	4
3	4	5	0	7	1	2	6
4	5	6	7	0	2	3	1
5	6	7	1	2	0	4	3
6	7	1	2	3	4	0	5
7	2	4	6	1	3	5	0

Figure.10: The Results Of Delete The 8^{th} Team

For example $n=7$, thus we will get the round robin schedule by fill the table of $n+1$ teams. According to the filling algorithm above, the result shows as Figure 6. Since there are 7 teams, it should delete the 8^{th} team arrangement. The result shows in Figure 10.

The result shows in Figure 6 is proved to be correct, the difference from Figure 6 to figure 5 is only delete the last row and column, and there are no duplicated elements in each row.

4.4 Complexity Analysis

The main task of the proposed table-filling algorithm is to fill the elements in matrix A.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Because there are n^2 elements, so the time complexity of the algorithm is $O(n^2)$. Furthermore, during the construction of timetable, we only use matrix A and no other auxiliary space is used, so the space complexity is also $O(n^2)$.

5 CONCLUSION

When the player number is even, we can get the schedule according to the table-filling algorithm. When the player number is odd, after even transforming, table filling and even eliminating, the time schedule can also be got. We also shown that the time and space complexity of the proposed algorithm is $O(n^2)$.

Compare with the algorithm proposed in [15], for the time scheduling of 6 competitors, the results is same. But in this paper, we proposed a symmetric matrix to construct the timetable, the correctness of the algorithm can be easily proved. Furthermore, making use of the symmetry properties, we can only fill the upper triangular matrix and reduce the time and space algorithm complexity.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant 60974005 and 60704004, the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20094101120008, the Nature Science Foundation of Henan Province under Grant 092300410201 and Zhengzhou Science and Technology Research Program under Grant 0910SGYN12301-6.

REFERENCES:

[1] Zhang Jia, Xie Chunhe, Liu Yuangui, "Answes to the Model of the Arrangements of Ball-games' Processes", Journal of Engineering Mathematics, Vol. 20, No. 3, 2003, pp. 124-129.

[2] Rong Zhenliang, Wen Xiaoyong, Guo Yixin, "Optimization Design for the Schedule Arrangement, Journal of Shaoyang University (Natural Science)", Vol.2, No. 4, 2003, pp. 25-29.

[3] Cheng Feng, Liang Fangchu, Cai Junwei, "A Mathematical Model of Arranging the Game Schedule for Single Round Robin", Journal of Ningbo University (NSEE), Vol. 17, No. 3, 2004, pp. 70-73.

[4] Wu Xiumei, Jiang Jing, Wang Shaohua, Wen Jinghe, "The Recursive and Non-recursive Solution for Calendar of Round Robin", Computer Knowledge and Technology, Vol. 3, No. 9, 2008, pp. 1445-1448.

[5] Cena Gianluca, Valenzano Adriano, "Achieving Round-robin Access in Controller Area Networks", IEEE Transactions on Industrial Electronics, Vol. 49, No. 12, 2002, pp. 1202-1213.

[6] Chaskar Hemant M., Madhow Upamanyu, "Fair Scheduling with Tunable Latency: a Round-robin Approach", IEEE/ACM Transactions on Networking, Vol. 11, No. 8, 2003, pp. 592-601.

[7] Fattah Hossam, Leung Cyril, "An Improved Round Robin Packet Scheduler for Wireless Networks", International Journal of Wireless Information Networks, Vol. 11, No. 1, pp. 2004, 41-54.

[8] Mitchell Paul D., Grace David, Tozer Tim C, "Analytical Model of Round-robin Scheduling for a Geostationary Satellite System", IEEE Communications Letters, Vol. 7, No. 11, 2003, pp.546-548.

[9] Oki Eiji, Jing Zhigang, Rojas-Cessa Roberto, Chao H. Jonathan, "Concurrent Round-robin-based Dispatching Schemes for Clos-network Switches", IEEE/ACM Transactions on Networking, Vol. 10, No. 12, 2002 pp.830-844.

[10] Zhang Xiao, Bhuyan Laxmi N., "Deficit Round-robin Scheduling for Input-queued Switches", IEEE Jounal on Selected Areas in Communications, Vol. 21, No. 5, 2003, pp.584-594.

[11] Lenzini Luciano, Mingozzi Enzo, Stea Giovanni, "Eligibility-based Round Robin for Fair and Efficient Packet Scheduling in Wormhole Switching Networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 3, 2004, 244-256.

[12] Wang Xiaodong, Computer Algorithm Design and Analysis,. Beijing: Electronic Industry Press, 2001.

[13] Cheng Guozhong, "Divide and Conquer Algorithm on the Calendar Problems of the Round Robin", Journal of China West Normal University (Natural Sciences), Vol. 25, No. 9, 2004, 279-281, 297.

[14] Liu Chao, Liu Jianhui, Linsen, "Study on

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Extending of Round Robin Calender Algorithm", Journal of Liaoning Technical University, Vol. 23, No. 6, 2004, pp. 50-52.

[15] Luo Yu, Hui Xiaowei, Li Enli, Luo Gang, "The Application of Horizontal Variance Arrangement Algorithm in the Round Robin", Journal of Liaoning University (Natural Sciences Edition), Vol. 32, No. 3, 2005, 61-65.

[16] Chou Wanxi, "2t Competitors Round-robin Tournment and Duplet Sets Partitions", Journal of Anhui University of Science and Technology (Natural Science), Vol. 26, No. 3, 2006, pp. 64-69.

[17] Chou Wanxi, "Round-robin Tournaments and a New Algorithm of the Pefect Matchings," Journal of Jinggangshang University (Natural Sciences), Vol. 27, No. 8, 2006, pp. 5-7.