***Introduction to Sports Scheduling Problem***

|  |  |
| --- | --- |
| Siddesh L C (16CO144)  Department of Computer Science and Engineering  National Institute of Technology Karnataka  Mangalore  16co144.siddesh@nitk.edu.in  Shreyas Pandith (16CO142)  Department of Computer Science and Engineering  National Institute of Technology Karnataka  Mangalore  16co142.shreyas@nitk.edu.in |  |

***Abstract*—Scheduling of sports tournaments needs to fulfil several types of constraints or limitations such as stadium unavailability, fixed games, forbidden matches, minimum number of breaks. There is no solution that satisfies all the constraints of tournament Scheduling. In this work, we will analyse the existing scheduling algorithms and try to improve or optimize the algorithms to maintain fairness in tournament.**

***Keywords—tournament; Sports Scheduling Algorithms; Round Robin Tournaments;***

1. INTRODUCTION

Number of sports tournament (e.g., cricket, football, basketball) usually encounters a number of scheduling problems for tournament. These problems involve numerous clashing imperatives to complete and different objectives to optimize, like decreasing the traveling distances, fixed number of matches played by a team per day, place inaccessibility at specific dates, fixed time gap between a home game and away game, etc. Creating acceptable and satisfactory timetable or schedules that completes the given conditions or satisfies these constraints and objectives is therefore an exceptionally difficult problem to get proper solution.

Sports scheduling is an area of increasing interest in constraint programming. As amateur and professional sports leagues proliferate and grow in size and complexity, organizers are increasingly turning to computer assisted scheduling [Nemhauser and Trick]. The scientific literature in this area is also growing and one can begin to get a sense of the range and mathematical difficulty of the problems encountered. These can include classical challenges such as set covering problems and quadratic assignment problems. In this note we concentrate on a version of a core problem that invariably comes up: determining whether a set of constraints on the schedule is feasible. This is often called the ÒtimetablingÓ problem of the scheduling process.

It is laborious to arrange the schedule for all kinds of

sports. Overseas professional contests, such as NBA, NHL, MLB ... etc., usually face this problem. There are targets needed to be reached, however, the sporting schedule is constituted and allotted under limited resource. It is easy to cause the complicated problem and take much time to find the answer. Obviously, it is hard to make the schedule for complete season in a short time. The planner has to consider the regulation for the league and the teams, such as the number of games for a weekend, the fairness of allotting the stadiums, the postponement of rainy days, and so on. The conditions are limited and complex and the schedule arrangement costs a lot of time to settle.

Official sport tournaments will involve a huge number of fans and considerable amount of speculations in players, broadcast rights, marketing, and advertising, challenging optimizations problem. They also need coordination and logistical efforts due to the large number of matches and participants.

The main problem in finding the timetable or schedule for the sports tournaments includes in finalizing the date and the place in which each match or game of the tournament will be played. The use of optimal scheduling is found in the tournaments of sports such as cricket, baseball, basketball, football, hockey etc, as these types of tournaments involve the place and time constraints. These scheduling problems have been described and solved by different exact and approximate methods, including constraint programming, integer programming, metaheuristic, and hybrid.

One way of getting insight into a complex environment is to classify what one sees and study the objects in each of the categories separately as a way of simplifying things. In fact there are many types of tournaments:  
**round robin tournament** (each team plays exactly k games against every other team (player))  
Note: Very often the value of k = 1 is so each team or player gets to play exactly one game (match) against every other team or player.  
 **elimination tournament** (the tournament progresses n rounds where in each round some players are eliminated and the surviving players are paired in future rounds, where again losers are eliminated)

Constraints can be applied universally to the whole schedule or locally to part of the schedule. (only in June, weekends, etc.).

Some of the constraints that matters in finding an optimal schedule are listed below:

1. Place constraints: In tournaments, every game is either home game or away game. Home game means the city where game is conducting is home city of either of the teams and away mean playing city is away from the home city. In Scheduling, we have to make sure that each team will fair number home games as well as away game.
2. Break/Gap constraints: We can consider break as the time gap between the two consecutive home games or away games. Sufficient gap should be provided in such cases mentioned before. A solution to Scheduling should provide a schedule which manages this constraint.
3. Distance and Time constraints: A optimal schedule should take care of this constraint. Scheduling games in improper way may lead to the decrease in player efficiency. For example, scheduling a game at Friday 9PM and next match on net day that is Saturday 12 noon can be considered as a bad schedule. Also, the distance between the games venues should be reduced so that cost of travelling can be reduced. In addition to this, time zone of different places can also be considered for an optimal schedule.

Some other constraints like hierarchical constraints, carry-over constraints and preferential constraints can also be considered. But we are limiting our study only to the basic constraints which are explained in above points.

1. LITERATURE REVIEW

A Hamilton path tournament structure consisting n teams and n/2 places, is a round-robin timetable or schedule on n-1 rounds in which each team plays in each place at most 2 times [11], [12], [13], and the set of games played in each place produces a Hamilton path on n teams. Before, Hamilton path tournament structure were proved to exist for all even n not divisible by 4, 6, or 10. Here, we give an inductive way for the building of Hamilton path tournament design for n = 2p >= teams [11].

Studies on ways of scheduling sports tournaments are quite manty [13], [17], [18], [19]. In particular, the problem of creating a round-robin timetable or schedule consisting n teams and n/2 places and n is even is basic problem in the area of combinatorial design.

A balanced tournament design with n teams in a round-robin schedule on n-1 days in which

* n/2 matches, one in each place are played on each day,
* each team uses each place at most 2 times.

Recently, a timetable or schedule almost related to balanced tournament designs has been considered by several authors [19], [20], [21], [22]. As in balances tournament designs, we have n teams with n/2 places, but in this case the goal is to find a timetable or schedule on n days in which each team pair plays at least one time, each team uses a place exactly two times, and no 2 teams meet 2 times in the same place. For easiness, we will call such schedules place-balanced. Place-balanced timetables or schedules have been constructed for all even n >= 8 (powers of 2 [14], all remaining even numbers n in [19]). Remember that each team has exactly one opponent which it will meet at two times.

The ubiquity of combinatorial optimisation of problems in our day today life society is illustrated by the novel application fields for optimization technology, which range from production chain management to sports games timetabling or scheduling. Over the last 20 years, constraints programming has emerged as a basic methodology to find the different types of combinatorial problems, and rich constraint programming languages are developed for showing and including constraints and mentioning search methods at a high level of abstraction. Local search methods to combinatorial optimisation are able to differentiate optimal or close-optimal answer with acceptable time constraints [21]. This book gives a way for solving combinatorial optimisation problems that involves the constraint programming and local search. After an overview of local search including neighbour hoods, heuristics, and metaheuristics, the book presents the architecture and modelling and search components of constraint-based local search and describes how constraint-based local search is support in COMET [21], [22], [23]. The book explains a number of uses, organized by meta-heuristics. It gives a timetabling or scheduling applications, and with the background required to understand these challenging problems. The book also involves a variety of acceptable problems, demonstrating the power of constraint-based local search approaches to cope with both acceptable and optimization problems in a same manner.

There is a way for the heavily constrained problem of finding a seasonable timetable or schedule for the best Danish soccer league. The league differs from most sports leagues, because it plays a triple round-robin tournament which directs to an uneven distribution of home and away games. The solution method described here employs a logic-based Benders decomposition in which the master problem finds home-away pattern sets while the sub problem finds timetables [26], [27], [28], [29]. And also, column producing methods are used to increase the speed of master problem. The computational answers proves that the solution method is capable of solving the problem with in acceptable time and Danish Football Association has employed it for finding the timetable or scheduling the season [30], [31].

Nemhauser and Trick [32] gave the solution of getting a timetable or schedule for the 1997-98 ACC in basketball. Their answer, generated with a using the concepts of both of integer and exhaustive enumerations, was used by the ACC.

Finite-domain constraint programming is one more programming method which is able to be used for solving combinatorial search problems such as sports tournament timetabling or scheduling. This paper tries to give an answer of round-robin tournament organizing based on finite-domain constraint programming [32]. The approach produces a dramatic increase in the performance, which makes an integrated interactive software solution feasible [23], [34], [35].

Tournament structuring is of crucial importance in competitive sports. The important and first goal of effective tournament structuring is to provide incentives for the participants to increase their performance both during the tournament and in the time period leading up to the tournament. In spectator sports, a second important goal of tournament structuring is to also encourage interesting match ups that produces fan interest. Seeded tournaments, in generally, encourage both the first and second goals. Teams or individuals with very good performances ends up to a tournament receive higher seeds which increase their opportunity of progressing more in future in the tournament. And also, seeding guarantees that the strongest teams or players are most likely to meet in the last rounds of the tournament when fan interest is at its very high position. Under some distributions of team or player skill, however, a seeding system can induce anomalies which could affect incentives. Our study of the NCAA basketball tournament of men opens such an anomaly [33], [34]. **The seeding system in this tournament gives teams with more success in the regular season more acceptable first round match ups, but the tournament is not reseeded as the games continues. Hence, while higher seeds progress to the 2nd round of the tournament at same higher rates than lower seeds, this bonding breaks down in future rounds [34], [35]. We recognize that 10th and 11th seeds average more wins and more progress farther in the tournament than 8th and 9th seeds.** This recognization violates the intended incentive design of seeded tournaments.

1. PROBLEM DESCRIPTION

Scheduling problems in sports leagues may be divided into two main classes: temporally-constrained and temporally-relaxed problems. In the time-constrained case, the planning horizon consists of the minimum number of periods (so-called rounds) required to schedule all the games and, hence, each team has to play exactly one game in each round. Tournaments following this pattern are said to be compact. On the other hand, in the time-relaxed case the number of periods is generally larger than the minimum number of rounds needed for scheduling all games. In this situation not every team necessarily plays in each round and thus teams may have some periods without a game.

The basic temporally-constrained problem for scheduling a sports league may be formulated as follows. The league consists of an even number n of different teams indexed by i ∈ {1, . . . , n}, where each team has to play against each other team exactly ` ≥ 1 times. The number of rounds available to schedule these n 2 ` = n(n − 1)`/2 games is equal to (n − 1)` , where each team has to play exactly one game in each round. Thus, one has to determine which teams i, j ∈ {1, . . . , n} play against each other in each round t = 1, . . . ,(n − 1)` and, for each of these pairings, whether it is played in the home stadium of team i (home game for team i) or in the home stadium of team j (away game for team i).

In most cases we have ` = 1 (single round robin tournament, SRRT) or ` = 2 (double round robin tournament, DRRT). For double round robin tournaments the season is often partitioned into two half series, where each pairing has to occur exactly once in each half (with different home rights). The second half series is usually not scheduled independently from the first. In a so-called mirrored schedule, the second series is planned complementarily to the first, i.e. the pairings of round t = 1, . . . , n − 1 in the second half are exactly the same as in round t of the first half, but with exchanged home rights. Another possibility is the so-called “English” system, where the pairings in the first round of the second half are the same as in the last round of the first half and rounds 2, . . . , n − 1 in the second half equal rounds 1, . . . , n − 2 of the first half.

If the league consists of an odd number of teams, in each round one team has a bye, i.e. does not play. This situation may be reduced to the previous case with an even number of teams by adding a dummy team n + 1. Then, in each round the team playing against n + 1 has a bye.

Usually, a schedule for a sports league is described by a so-called opponent schedule and a so-called home-away pattern. An opponent schedule may be represente by an n × (n − 1)-matrix where the entry oppit ∈ {1, . . . , n} \ {i} specifies the opponent of team i in round t. If this matrix is enlarged by an additional column containing the teams 1, . . . , n, its structure is a latin square (each column and each row is a permutation of 1, . . . , n such that in any column or row no number occurs twice). Additionally, the latin squares fulfill the symmetry condition that oppit = j if and only if oppjt = i (i.e. if team i plays against team j in round t, also team j plays against team i in that round).

A home-away pattern (HAP) is defined as n×(n−1)-matrix H = (hit), where hit equals “H” (resp. “A”) when team i has a home (resp. away) game in round t. If two consecutive entries hi,t−1 and hit in a row i are equal for some t = 2, . . . , n − 1, then team i has a so-called break in round t (i.e. the alternating sequence of home and away games is broken).

1. SOLUTION PROPOSED

In sports Scheduling there are many constraints such that every team play with others twice, one on owns home town and another on opponent home town. And our problem is define one most important constraint, i.e. sufficient gap between previous and next matches for all teams. So we are introducing an algorithm which satisfy all previous constraints and also provide the at least two matches gap between previous and next matches for all teams and this is our constraint on IPL (Indian Premier League) tournament scheduling. There are 6 previous schedules of IPL but no one satisfy this constraint, so we have design an schedule who follows all previous and our constraint.

Our main objective is providing sufficient gap of matches to each and every team to perform better in tournament. And providing equal chance to all team for relaxes. So, we have design an algorithm which provides many schedules with constraint of at least two matches gap between previous and next matches for each team.

First, we have making a simple combination of pairs in which we have n teams, n stadium and each team played two matches with all other teams i.e. n\*(n-1) matches played in tournament and also n\*(n-1) pairs for one combination of this tournament. Now, we will start to rearrange the simple pairs of combination according to constraint. And our constraint is at least two matches different between previous and next match for each and every team. If team t1 played 5th match then in 6th and 7th matches t1 will not allowed to play and same procedure for all teams. We start from first pair and then choose the second according to given constraint and same for all others. And delete the selected pairs from the simple pairs of combination and if there is any pair left in the simple pairs of combination which is/are not chosen from the simple pairs of combination then we will use backtracking algorithm to put the left pairs of the simple combination. In backtracking algorithm, we have to choose pairs from pairs of simple combination one by one and start to put it into IPL combination with same constraint from bottom to top checking. And if any pair is not feet in IPL combination then we will rearrange the pairs of simple combination, we will use one left shift operation (first pair replace with second, second pair replace with third and so on for all others, and at last place first pair will come to complete the n\*(n-1) pairs), and this pairs of combination is called pairs of simple combination then we will restart our algorithm to make the IPL combination again and this whole process will applied for until we will not got the n\*(n-1) pairs in IPL combination or until we will reaches n\*(n-1) one left shift operations in pairs of simple combination.

1. EXPERIMENTAL RESULTS AND ANALYSIS

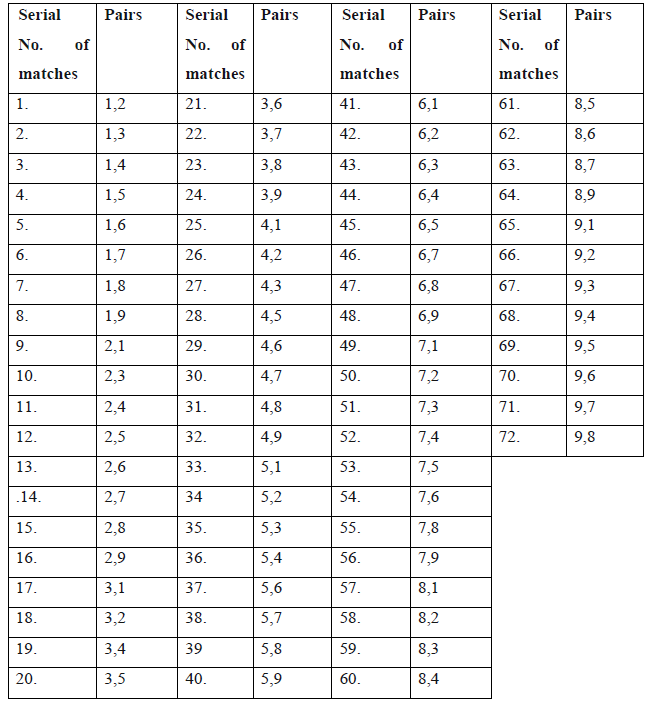
Table 4.1 indicate the pairs of simple combination. In which we have n\*(n-1) pairs and each team play to all other exact two matches. This is a simple combination of pairs for the 9 teams, it means we have considered n = 9.

Every match is played in a stadium and first element of the pair indicate the stadium, such as suppose first pair is (1, 2), it means team t1 and team t2 play with each other on the t1 home town stadium. First element of each pair is not only indicating the team number but also indicate about the stadium. Through which we can satisfy the constraints of IPL (Indian Premier League) tournament, i.e. each team play with all other exact two matches, one match on owns home town stadium and another on opponent home town stadium.

Table 4.1 show the simple combination of the pairs which is not satisfying our constraint, which is gap constraint. So, we will rearrange the above schedule according to our constraint of gap i.e. at least two matches gap between previous and next matches played by each and every team.

First of all, we start to select one by one pairs and put on another table with our restriction. Such as, select pair (1, 2) and put in to another table and remove from the table4.1. Then select another i.e. (1, 3), but team t1 has played in last match of table4.2, so we can’t put it into table4.2, so we will consider next pair from the table4.1. As we are seeing pair (1, 3), (1,4)... (2, 9), (3, 1) and (3, 2) is not satisfy our constraint, so we will select next pair i.e. (3, 4) and this pair satisfy the constraint show in table4.2.

Table 4.1



Now, we will continue to choose such pairs, and if we will reach on the last then again start from the starting off the table until we are unable to select any one from the remaining pairs.

There may be opportunity that there is no any pairs left in the simple combination of pairs schedule, that means we have a successful schedule of the IPL according to constraints. And there may be possibility that there are some pairs left in the simple combination of pairs schedule, i.e. show in table4.4.

Table 4.2

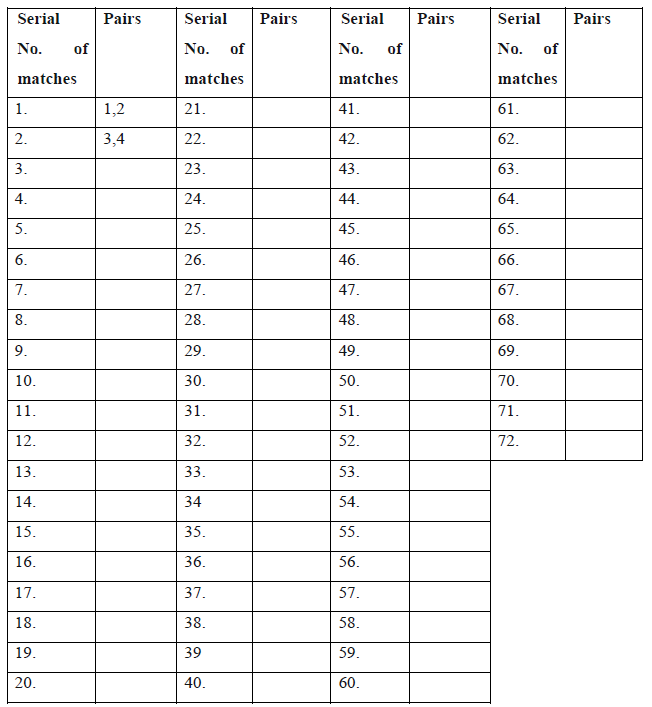


Table 4.3

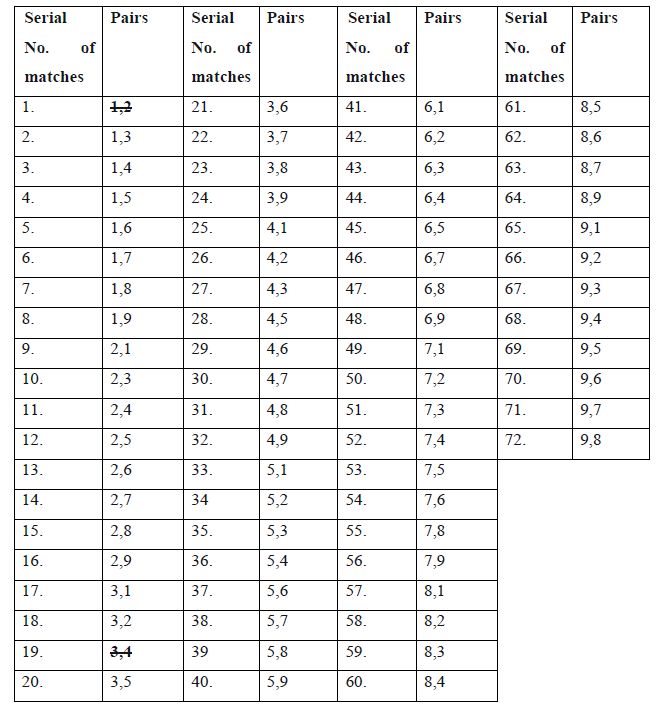
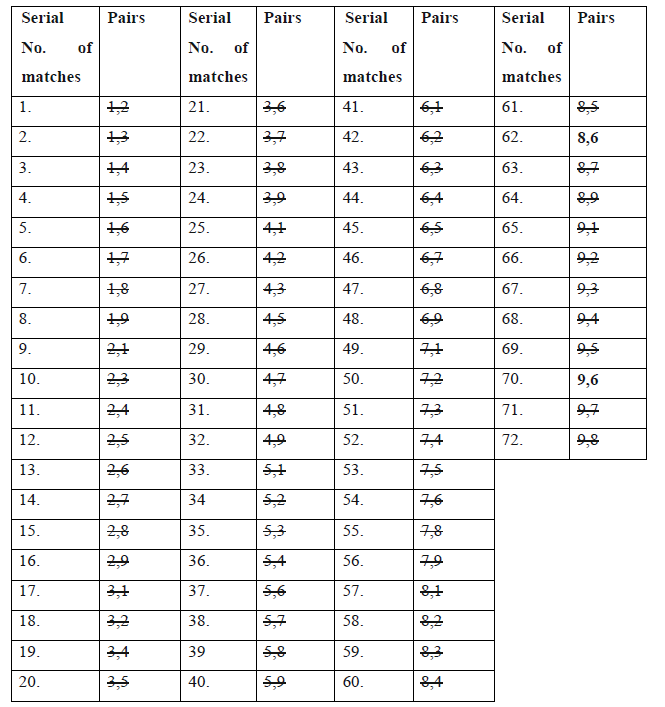
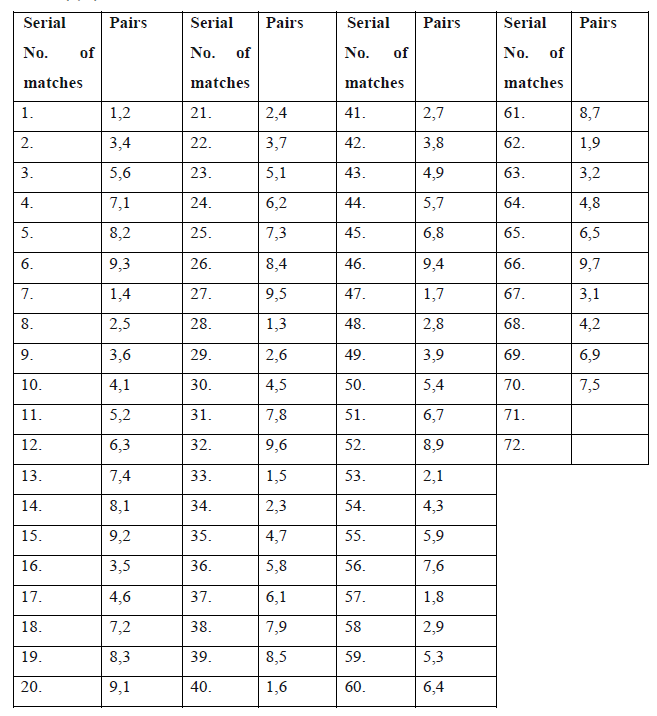


Table 4.4



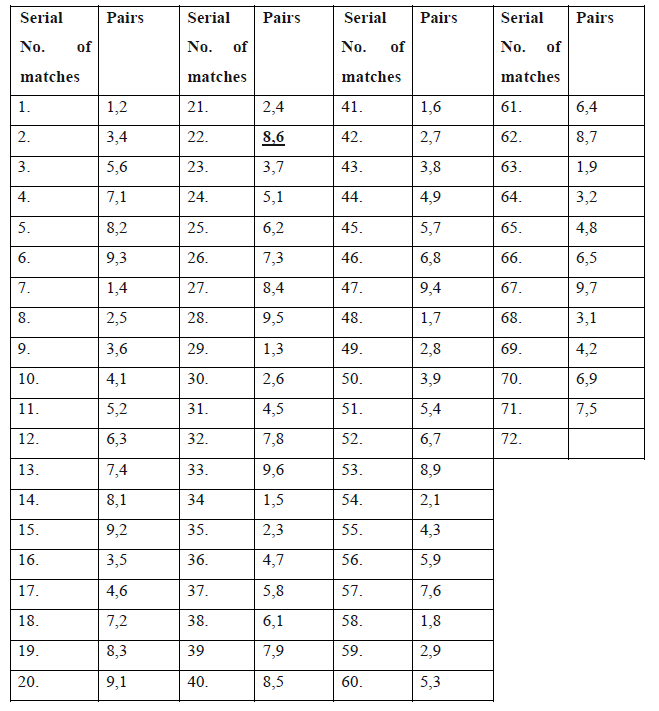
And the table 4.5 indicates the possible schedule of 70 matches, there is 2 pairs are not satisfying our constraints. So for remaining pairs we will use iterative backtracking method to find the appropriate schedule for IPL.

Table 4.5



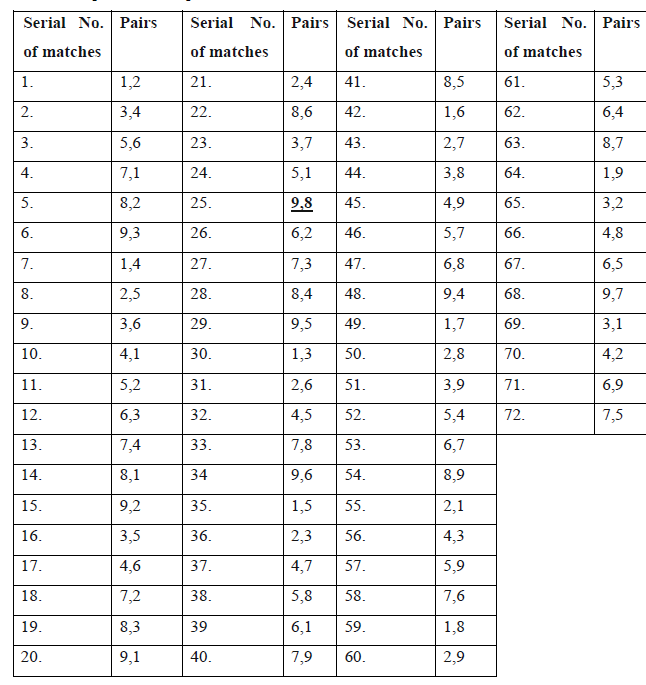
Here we have left two pairs, i.e. (8, 6) and 9, 8) and to insert these pairs in IPL tournament schedule we have used iterative backtracking algorithm on all remain pairs, i.e. after inserting pair (8, 6) we have indicated in table 4.6.Table 4.6 indicate all pairs of tournament except pair (9, 8).

Table 4.6



There is only one pair is left in simple tournament schedule to insert in the IPL tournament schedule. So we have applied backtracking on another remain pair, i.e. pair (9, 8). And table 4.7 indicate the one schedule of combination of the pairs of the IPL tournament after inserting the pair (9, 6).

Table 4.7



Each simple combination of pairs is not necessary to produce IPL tournament schedule. There may be some arrangements those are not producing any schedule. If this type of simple combination is founded then the algorithm drops the schedule and applied backtracking approach on simple combination of pairs for the next simple combination of pairs.

Table 4.8

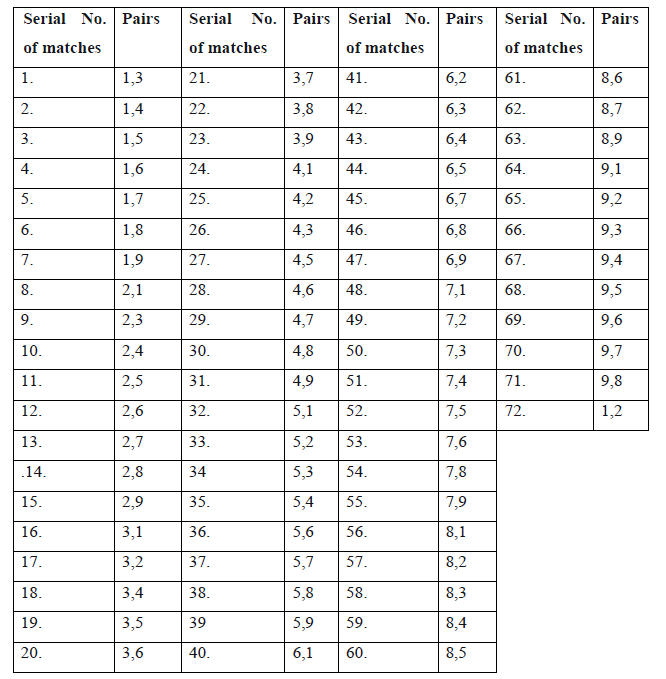


Table 4.8 shows such type of simple pairs generated during the backtracking on the simple combination of pairs. This table indicate the simple combination of pairs after producing the successful IPL schedule with starting pair (1, 2), and after that it shift all pairs up with just one position and first pair shift to last position in combination. It means after shifting the starting pair of the simple combination is (1, 3) and the last pair is (1, 2). This type of shifting is done for all pairs once. Table 4.8 indicate such simple combination of pairs whose IPL tournament scheduling is not possible with given constraints.

Table 4.9 shows all arrangement of table 4.8 without applying backtrack approach. There are some pairs, i.e. pairs (1, 7), (2, 9), (3, 8), (5, 9), (6, 4), (8, 7) and (9, 7) are not inserted showing in table 4.10, because of given constraints. So we applied backtracking on all remaining pairs one by one. And remaining pair (1, 7) is not fulfil given constraints and not got any position in IPL tournament schedule. So, the algorithm drops this simple combination of pairs and move for the next one using the sifting down to up with one shift.

This is the main demerit of this algorithm and to solve that we can modify in this algorithm and then we can solve these type of problem. Actually we can get number of combination may be n \* (n-1) IPL tournament schedule. And we are working on it to improve this algorithm to get all the possible combination of the IPL tournament schedule.

Table 4.9

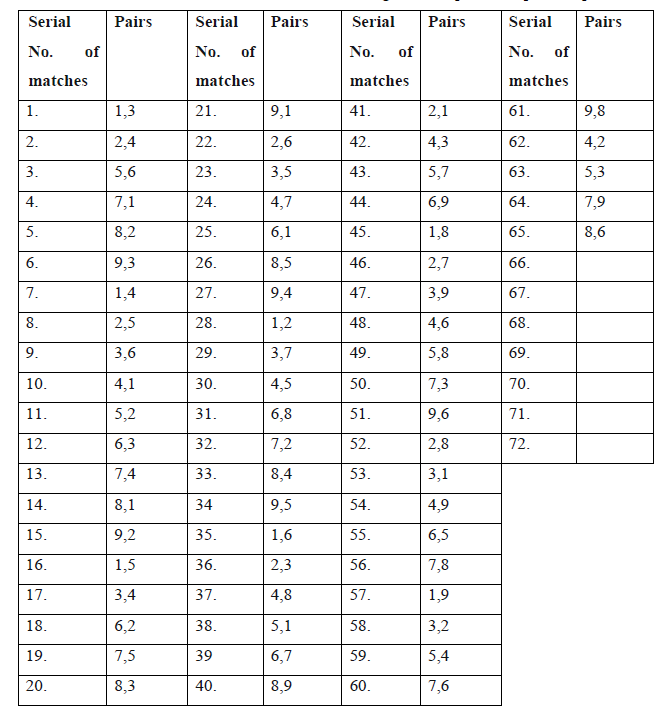
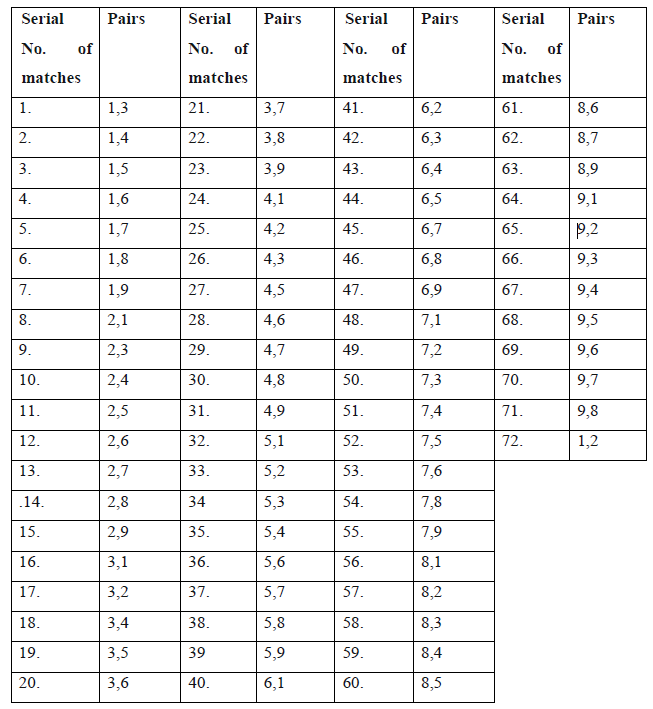


Table 4.10



1. CONCLUSION
2. REFERENCES

[1] K. Coolsaet, J. Degraer, “A computer assisted proof of the uniqueness of the Perkel graph, Designs Codes Cryptogr,” 34(2-3) (2005) 155-171,2005.

[2] Bing JIANG, Yongmin MU, Zhihua Zhang, “Research of Optimization Algorithmfor Path-Based Regression Testing Suit: Proceedings of Second International Workshop on Education Technology and Computer Science,” *IEEE,* 2010.

[3] Leung H. K. N, White L, “Insights into regression testing: Proceedings of International conference on Software Maintenance,” LOS Alamipos: *IEEE* *Computer Society, 1989, pp. 60-69*.

[4] David G. Sullivan, ” Recursion and Recursive Backtracking*,” Computer Science E-119Harvard Extension School,* 2012.

[5] David S. Johnson, Michael A. Trick, “Cliques, Coloring, and Satisfiability: Second Dimacs Implementation Challenge,” *American Mathematical Soc., 01-Jan-* *1996,October 11-13, 1993 (Google eBook).*

[6] M. Melcher and K. B. Reid.,” Monochromatic sinks in nearly transitive arccolored Tournaments,” *Discrete Math., 310(20):2697–2704,* 2010.

[7] H.A. Priestley, M.P. Ward,” A Multipurpose Backtracking Algorithm,” *Ph.D. thesis, Mathematical Institute 24/29, St. Giles Oxford OX1 3LB,*January 17, 2003.

[8] Ambjörn, Seatrack,” Web forecasts and backtracking of oil spills-an efficient tool to find illegal spills using AIS,” *US/EU-Baltic International Symposium, 2008* *IEEE/OES, pp 1-9, C,* 2008.

[9] Gerhart, S. L. & Yelowitz, L., “Control Structure Abstractions of the Backtracking Programming Technique,” *IEEE Trans. Software Eng. SE 2, 4, pp. 285{292,* 1976.

[10] Ward, M.,” Derivation of Data Intensive Algorithms by Formal Transformation,” *IEEE trans. Software Eng. 22, 9, pp. 665{686.* *hhttp://www.dur.ac.uk/»dcs0mpw/martin/papers/sw-alg.ps.gzi,* 1996.

[11] Yoshiko T. Ikebe , Akihisa Tamura, “ Construction of Hamilton Path Tournament Designs,” in *Springer, Graphs and Combinatorics (2011) 27:703–711, DOI*

*10.1007/s00373-010-0998-6,* 2011.

[12] de Werra, D.,”Some models of graphs for scheduling sports competitions,”, *Discrete Appl. Math. 21,47–65,* 1988.

[13] de Werra, D., Ekim, T., Raess, C.,” Construction of sports schedules with multiple venues,”, *Discrete Appl. Math. 154, 47–58,* 2006.

[14] Ikebe, Y.T., Tamura, A.,v” On the existence of sports schedules with multiple venues,”, *Discrete Appl.Math. 156, 1694–1710,* 2008.

[15] Lamken,E.R.,Vanstone, S.A.,” Balanced tournament designs and related topics,*” Discrete Math. 77, 159–176*, 1989.

[16] Nemhauser, G., Trick,M.,” Scheduling a major college basketball conference,*” Oper. Res. 46, 1–8,* 1998.

[17] Russell, R.A., Urban, T.L.,” A constraint-programming approach to the multiple venue sport-scheduling problem,” *Comput. Oper. Res. 33, 1895–1906,* 2006.

[18] Urban, T.L., Russell, R.A.,” Scheduling sports competitions on multiple venues*,” European J. Oper. Res. 148, 302–311*, 2003.

[19] LeRoy B. Beasley, Richard A. Brualdi, and Bryan L. Shader,” *Combinatorial orthogonality, Combinatorial and Graph-Theoretical Problems in Linear Algebra,” Discrete Math. 77, 159–176*, 1993.

[20] Richard A. Brualdi, Shmuel Friedland, and Victor Klee, eds.,” The IMA Volumes in Mathematics and its Applications,” *vol. 50, Springer- Verlag, New York, 1993,*

*pp. 207{218*.

[21] Lowell W. Beineke and K. B. Reid, ” Tournaments, Selected Topics in Graph Theory,” *Academic press,New york, pp. 169{204.,* 1978.

[22] Alfred Brauer and Ivey C. Gentry,” On the characteristic roots of tournament matrices,” *Bulletin of the American Mathematical Society 74 (1968), 1133{1134,* 1968.

[23] Ezra Brown and K.B. Reid,” Doubly regular tournaments are equivalent to skew hadamard matrices*,” Journal of Combinatorial Theory, Series A 12 (1972),* *332{338.198,* 1972

[24] Richard A. Brualdi, Rachel Manber, and Je\_ery A. Ross,” On the minimum rank of regular classes of matrices of zeros and ones,” *Journal of Combinatorial Theory,* *Series A 41 (1986), 32{49*, 1986.

[25] D. deCaen, D.A. Gregory, S.J. Kirkland, and N.J. Pullman, ” Algebraic multiplicity of the eigen values of a tournament matrix, Linear Algebra and its Applications,” *169, 179{193.,* 1992.

[26] Faun C. C. Doherty, J. Richard Lundgren, and Daluss J. Siewert,” Biclique covers and partitions of bipartite graphs and digraphs and related ranks of (0; 1)- matrices,” *Congressus Numerantium 136 (1999), 73{96,* 1999.

[27] Carolyn Eschenbach, Frank Hall, Rohan Hemasinha, Stephen J. Kirkland, Zhongshan Li, Bryan L. Shader, Je\_ery L. Stuart, and James R.Weaver,” On almost regular tournament matrices, Linear Algebra and its Applications,” *306* *(2000), 103{121.199*, 2000.

[28] Bryan Shader, ” On tournament matrices, Linear Algebra and its Applications,” *162-164 (1992), 335{368., 1992*.

[29] Daluss J. Siewert, “ Biclique covers and partitions of bipartite graphs and digraphs and related ranks of f0; 1g-matrices,” Ph.D. thesis, *University of Colorado at* *Denver, Denver, Colorado*, May 2000.

[30] J. H. vanLint, ” f0; 1; \_g distance problems in combinatorics, Surveys in Combinatorics,” 1985 (Ian Anderson, ed.), *London Mathematical Society Lecture*

*Note Series, vol. 103, Cambridge University Press, Cambridge, pp. 113{135.,* 1985.

[31] Bing JIANG, Yongmin MU, Zhihua Zhang, “Research of Optimization Algorithm for Path-BasedRegression Testing Suit: Proceedings of Second *International* *Workshop on Education Technology and Computer Science,” IEEE, DOI*

*10.1109/ETCS.2010.365*, 2010.

[32] Leung H K N, White L, “Insights into regression testing: Proceedings of International Conference on Software Maintenance,” LOS Alamipos: *IEEE* *Computer Society, pp. 60-69.,* 1989.

[33] Rasmus V. Rasmussen, “Scheduling a triple round robin tournament for the best Danish soccer league,” *Department of Operations Research, University of Aarhus,* *Ny Munkegade, Building 1530, 8000 Aarhus C*, Denmark.

[34] Martin Henz, “Scheduling a Major College Basketball: Proceedings of national conference on Computer Science,” *School of Computing, National University of* *Singapore*, 1999.

[35] Robert Baumann, Victor A. Matheson and Cara A. Howe, “Anomalies in Tournament Design,” in *Journal of Quantitative Analysis in Sports, Volume 6,* *Issue 2, Pages –, ISSN (Online) 1559-0410, DOI: 10.2202/1559-0410.1233*, April 2010.