Scheduling Court-Constrained Sports Tournaments

Michael Trick

Graduate School of Industrial Administration, Carnegie Mellon, Pittsburgh, PA USA, 15213 trick@cmu.edu

Abstract. We examine sports tournaments where the number of games that can be played in a time slot is limited. It is always possible to schedule such tournaments, and we show how to do so while minimizing the gap between consecutive games for any team.

1 Introduction

Consider creating a sports league schedule for $2n$ teams over k time slots, where the schedule must satisfy the following:

(P1) Teams play in pairs.
(P2) No team plays more than once in any time slot.
(P3) Every pair of teams plays exactly once over the course of the schedule.

If we add the restriction that every team plays in every time slot, then we have the standard round-robin scheduling problem, dating back to at least Kirkman's 1847 paper [4]. But it is common in practice for there to be limits on the number of games that can be played in each slot. For instance, there may be a limited number of courts on which to play, or the availability of courts may vary depending on the day of the week the time slot occurs. If these limits allow less than n games in a time slot, standard round-robin schedules do not necessarily apply.

Suppose slot i has a limit of $b_i \leq n$ games to be played, where $\sum_{i=1}^{k} b_i = \frac{n(n-1)}{2}$. The *Court Constrained Tournament Array Problem* is a schedule satisfying P1, P2, and P3 and satisfying P4:

(P4) Exactly b_i games are scheduled in each slot i.

The case where $b_i = n$ for all i is the standard round-robin scheduling problem, while the case where $b_i = c$ for a fixed constant c is called the Tournament Array problem TA($2n$,c) [5]. In that paper, Mendelsohn and Rodney give a construction that not only creates the tournament, but also balances the appearance of each team on each court (where each of the c games in a slot is interpreted as being on a particular playing court). Since our problem is a generalization of the Tournament Array problem, we will denote instances of our problem as TA($2n$,b) where b is the vector of game limits.

Example: TA(4,$[1, 2, 1, 2]$) has solution

$$\begin{array}{r|ll}
1 & (1,2) & \\
2 & (1,3) & (2,4) \\
3 & (3,4) & \\
4 & (1,4) & (2,3)
\end{array}$$

One important characteristic of a tournament array is the "evenness" of play: does any team go a long period without playing? We say that a tournament array T has *gap* g if, for some team j, there are consecutive time slots $[s..t]$ such that (1) j does not play in $[s..t]$, and (2) $\sum_{i=s}^{t} b_i = g$. The gap for a tournament array is the maximum such g for which this occurs.

We show that there is a solution to the Court Constrained Tournament Array Problem for every choice of b, and we construct a solution with gap no more than n, which is the best possible result.

In Section 2, we outline some methods for solving the standard round robin scheduling problem, and, based on these, give two methods for solving the Tournament Array problem. In section 3 we analyze the gaps that result from these constructions. The final section gives some future directions for this work.

2 Constructing Tournament Arrays

2.1 Standard Round Robin Tournaments

The first construction of round robin tournaments appears to be due to Kirkman [4]. To schedule the standard round robin tournament (TA($2n$,n) in our notation), his approach was to take teams 1,2,. . . ,$2n$ and order the pairs (i, j) in their lexicographic order, so that (i, j) is before (i, k) for $j < k$ and (i, j) before (k, l) for $i < k$. The slots are then considered in cyclic order. Games are assigned "greedily" in the cyclic order: start by assigning $(1, 2)$ to the first slot. Subsequent games are assigned to the next slot (considered cyclically) to which the game can be legally assigned. So, for $n = 6$, we get the tournament (reading left to right, top to bottom in the order the algorithm assigns the games):

$$\begin{array}{r|llllll}
1 & (1,2) & & & (3,5) & & (4,6) \\
2 & (1,3) & & (2,6) & & (4,5) & \\
3 & (1,4) & (2,3) & & & & (5,6) \\
4 & (1,5) & (2,4) & & (3,6) & & \\
5 & (1,6) & (2,5) & (3,4) & & &
\end{array}$$

Andersen[2] gives a nice proof that this approach does give a correct round robin tournament.

Kirkman's approach does not work, however, for the tournament array problem. Consider TA(4,$[1, 2, 1, 2]$). The greedy approach leads to the partial schedule:

$$\begin{array}{r|ll}
1 & (1,2) & \\
2 & (1,3) & (2,4) \\
3 & (1,4) & \\
4 & (2,3) & \text{X}
\end{array}$$

where the remaining game $(3, 4)$ cannot be assigned to the remaining position "X".

Andersen [1] proved that Kirkman's construction is equivalent to what is now the "standard" construction: Label the $2n$ teams 1,2,..., 2n and create the first round schedule

$(2n,1)$, $(2,2n-1)$, $(3,\ 2n-2)$,... $(n,n+1)$

For subsequent rounds, add 1 (mod $2n-1$) to each number except $2n$ (treating 0 as $2n-1$). So the next round would be

$(2n,2)$, $(3,1)$, $(4,\ 2n-1)$,... $(n+1,n+2)$

For this construction, slot i consists of the games ($2n$,i) and (k, l) where $k + l \equiv 2i \bmod 2n-1$. For $2n = 8$ we get the schedule

1	(8,1)	(2,7)	(3,6)	(4,5)
2	(8,2)	(3,1)	(4,7)	(5,6)
3	(8,3)	(4,2)	(5,1)	(6,7)
4	(8,4)	(5,3)	(6,2)	(7,1)
5	(8,5)	(6,4)	(7,3)	(1,2)
6	(8,6)	(7,5)	(1,4)	(2,3)
7	(8,7)	(1,6)	(2,5)	(3,4)

While this schedule is equivalent to Kirkman's construction (after suitable permutations of the rows and teams, see [1]), there exist multiple non-equivalent tournaments. For instance, starting with

1	(8,1)	(2,4)	(3,7)	(5,6)

results in an inequivalent tournament [3] (check!).

2.2 Tournament Array Scheduling

With a round robin schedule, we can now construct a tournament array TA($2n$,b) for any vector b of game limits. We give two constructions: the first only works when starting with the "standard" round robin schedule, while the second works with any round robin schedule.

Kirkman Tournament Array Given the standard round robin schedule with initial slot

$(2n,1)$, $(2,2n-1)$, $(3,\ 2n-2)$,... $(n,n+1)$

we say that a game (k, l) is game m in slot i if it occurs in the mth column of slot i. For instance, $(3,2n-2)$ is game 3 in slot 1. The key insight is that if team j plays in game m in slot i, then it plays in either slot $m-1$, m, or $m+1$ in slot $i+1$. This implies that if we order all the pairs first by rounds, then by games within rounds, any $n-1$ consecutive games do not repeat a team. For our 8 team example, the ordering gives

(8,1) (2,7) (3,6) (4,5) (8,2) (3,1) (4,7) (5,6) (8,3) (4,2) (5,1) (6,7) (8,4) (5,3) (6,2) (7,1) (8,5) (6,4) (7,3) (1,2) (8,6) (7,5) (1,4) (2,3) (8,7) (1,6) (2,5) (3,4)

and no 3 consectutive pairs repeats a team.

Call a slot with $b_i = n$ a "full slot"; any other slot is a "partial slot". If we begin by filling in the full slots, then the partial slots can be filled in with the ordering given above without fear of having a team play twice in the same slot. So our algorithm for creating TA($2n$,b) based on the standard round robin schedule is

Step 1. Beginning with the first slot of the standard round robin schedule, schedule all full slots by assigning the games in the first slots of the round robin schedule. If there are k full slots, the first k slots of the round robin schedule are then used.

Step 2. Order the remaining pairs of the round robin schedule first by slots, then by games. Assign pairs to the partial slots in that order.

For example, to create TA(8,[3,3,4,3,3,4,3,2,3]), we do the following:

Step 1. Schedule the full slots:

1	
2	
3	(8,1) (2,7) (3,6) (4,5)
4	
5	
6	(8,2) (3,1) (4,7) (5,6)
7	
8	
9	

Step 2. Schedule the partial slots. The first partial slot becomes:

1	(8,3) (4,2) (5,1)
2	
3	(8,1) (2,7) (3,6) (4,5)
4	
5	
6	(8,2) (3,1) (4,7) (5,6)
7	
8	
9	

The rest then becomes

1	(8,3)	(4,2)	(5,1)	
2	(6,7)	(8,4)	(5,3)	
3	(8,1)	(2,7)	(3,6)	(4,5)
4	(6,2)	(7,1)	(8,5)	
5	(6,4)	(7,3)	(1,2)	
6	(8,2)	(3,1)	(4,7)	(5,6)
7	(8,6)	(7,5)	(1,4)	
8	(2,3)	(8,7)		
9	(1,6)	(2,5)	(3,4)	

To prove correctness of this algorithm, we need only show that team j does not appear twice in any slot.

Matching Based Tournament Array An alternative approach begins with any round robin schedule (not just the standard one). This approach relies on the following theorem:

Given $2n$ teams, matching M_1 on those teams with k_1 edges, and perfect matching M_2 on those teams (hence with n edges), for any k_2, $k_1 \leq k_2 \leq n$, there is a matching $M \subseteq M_1 \cup M_2$ such that $|M| = k_2$ and $(M_1 \cup M_2)/M$ is also a matching.

With this theorem, we begin with the games in the first slot of the round robin tournament and assign them to the first slot of the tournament array. We then move to the second slot of the tournament array. If we have enough games remaining from the current slot of the round robin tournament to schedule the tournament array, we do so. If not, then we set M_1 to be the remaining games from the round robin slot and let M_2 be the matching in the next slot of the round robin. We then invoke the theorem to extract a matching from the union of M_1 and M_2 of size equal to the b_i of the tournament array, leaving a matching to continue the algorithm. At every step we have a partial matching, so the algorithm works by induction.

3 Gaps

In most real schedules, it is important that every team plays regularly: long periods without games makes for a poor schedule. We can measure the number of games played between consecutive games for a given team and define the gap of a tournament array to be the maximum such value for all teams.

For the Kirkman Tournament Array, the gap is less than or equal to n. For the Matching-based Tournament Array, the gap is less than or equal to $3n/2$.

4 Conclusion and Future Directions

We have shown that it is possible to find tournament arrays for arbitrary TA($2n$,b) and have given two construction algorithms. The gaps for these tournament arrays are small.

It would be interesting to look at optimization versions for this problem, where there is a cost of $c(i, j, k)$ of having teams i and j play in slot k. Trick [6] looked at such a problem for round-robin scheduling, and compared integer and constraint programming approaches for the problem.

Another interesting direction would be to balance the courts or other venues used by the teams. Suppose each game position had a venue associated with it. Can games be assigned so as to balance team use of the venues? Mendelsohn and Rodney [5] showed that is the case for TA$(2n,c)$ for a constant c; it would be interesting to explore this generalization.

Acknowledgement

I thank Larry Waldman for bringing this problem to my attention.

References

1. Anderson, I. 1991. "Kirkman and GK$_2n$", Bulletin of the I.C.A., **3:** 111–112.
2. Anderson, I. 1997. *Combinatorial Designs and Tournaments*, Oxford University Press.
3. Horton, J.D. 1989/90. "Orthogoanl starters in finite abelian groups", Discrete Mathematics, **79:**, 265–278.
4. Kirkman, T.P. 1847. "On a problem in combinations", Cambridge and Dublin Mathematics Journal, **5:** 255–262.
5. Mendelsohn, E. and P. Rodney 1994. "The existence of court balanced tournament designs", Discrete Mathematics, **133:** 207–216.
6. Trick, M.A., "Integer and constraint programming approaches for round robin tournament scheduling",to appear, Springer Series on Computer Science.