# Round Robin Tournament Analysis - First Fit Musings

Richard A. DeVenezia
richard@devenezia.com
Updated: 8/15/2003

This document organizes my thoughts on some of the mathematics that need understanding to implement a **first fit** algorithm to schedule a round robin tournament. There are other, much simpler, algorithms that work on the concept of cycling. The cycling algorithm is not discussed here.

Definition:

A tournament schedule, $S_T$ , is an arrangement of an even number of $n$ items whose unique pairwise combinations are organized into $n-1$ groups of $n/2$ pairs, such that each group contains unique items.

number of items: $n$
number of groups: $n-1$

number of pairs in a group: $m=\frac{n}{2}$

number of pairs in tournament: $n_p = {}_nC_2 = \binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2} = (n-1)\times m$

Let each item be uniquely identified by a number between 1 and $n$.
Let $p_{ij}$ indicate a pairing between items $i$ and $j$, $0<i<j$

Let the value of $p_{ij}$ be an integer that uniquely identifies the pair in a sequence, $1 \le p_{ij} \le n_p$ .

Consider $P$ that organizes these identifications.

$$P = \begin{bmatrix} \cdot & p_{12} & \cdots & p_{1n} \\ \cdot & \cdot & \ddots & \vdots \\ \vdots & \ddots & \cdot & p_{(n-1)n} \\ \cdot & \cdots & \cdot & \cdot \end{bmatrix}_{n\times n}$$

Let $p_{ij} = E(i,j)$ be the enumerating mapping function.

Note: There are $n_p!$ possible mappings since $E$ is an arbitrary unique mapping of $n_p$ pairs.

Consider one such function: $E_U$ , the left to right, top to bottom, monotonic pair identifying enumerator.

$$E_U = \begin{bmatrix} \cdot & 1 & 2 & \ldots & n-1 \\ \cdot & \cdot & n & \ddots & n-1+n-2 \\ \cdot & \cdot & \cdot & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & n_p \\ \cdot & \cdots & \cdot & \cdot & \cdot \end{bmatrix}_{n\times n}$$

$$E_U(0,n) = 0$$
$$E_U(i,j) = E_U(i-1,n) + j - i; \qquad i = 1..n-1,\ j = i+1..n$$

Define $S_x$ , a schedule matrix containing $n_p$ entries.

$$S_x = \begin{bmatrix} s_{11} & \cdots & s_{1j} & \cdots & s_{1m} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ s_{i1} & \ddots & s_{ij} & \ddots & s_{im} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ s_{(n-1)1} & \cdots & s_{(n-1)j} & \cdots & s_{(n-1)m} \end{bmatrix}_{(n-1)\times m} \quad \text{where } s_{ij} \in E,\ s_{ij} = E(i', j')$$

$S_x$ represents any possible arrangement of pair items, not necessarily a tournament schedule. $s_{ij} = E(i', j')$ means that the values in $S_x$ are found in $E$ by look up. Consider an arbitrary $S_x$ when $n = 6$ : $s_{3,5} = 10 = E_U(4,5)$ ; $i = 3; j = 5; i' = 4; j' = 5$ .

The purpose of the first fit algorithm is to examine arrangements of $s_{ij}$ selected from enumerations of $i', j'$ pairs until the conditions of a tournament schedule are met.

Question: What kind of space is the first fit algorithm going to work in ?

Let $S^*$ = the set of all $S_x$, there are $n_p!$ matrices in $S^*$. Each $S_x \in S^*$ corresponds to one of the $n_p!$ permutations of the sequence of numbers generated by $E$ .

Let group $g_i = \{s_{i1},...,s_{im}\}$

Define equivalence $S_x \equiv S_y$ meaning $g_i^x$ is permutable to $g_j^y$ .

Define a partition of $S^*$ as all $S_x \equiv S_y$

In other words, two matrices in $S^*$ are in the same partition and are equivalent if they can be made identical by rearranging $s_{ij}$ within a group and rearranging groups $g_i$ within $S_x$ . (i.e. group-wise rearrangement after within group rearrangement.)

Question: How many partitions are contained in $S^*$ , and how many $S_x$ are contained in a partition?

Consider a specific $S_x$ composed of $g_i; i = 1..n-1$

There are $m!^{(n-1)}$ within group arrangements.
Each group can be permuted $m!$ ways. There are $n-1$ groups. If the group order is *locked* then $S_x$ was selected from a set of $m!^{(n-1)}$ different forms. I.e. group 1 is 1 of $m!$ choices, group 2 is 1 of $m!$, … group $n-1$ is 1 of $m!$ choices, $m!$ x $m!$ x…x $m!$ = $m!^{(n-1)}$

When the group order is *unlocked* there are $(n-1)!$ group arrangements.

I speculate that any $S_x$ is in a partition containing $m!^{(n-1)}(n-1)!$ equivalent matrices.

I further speculate the number of possible partitions to be

$$\frac{n_p!}{m!^{(n-1)}(n-1)!} = \frac{(m(n-1))!}{m!^{(n-1)}(n-1)!}$$ possible partitions

As for factoring the above, only vague ideas about decomposition and prime number distributions come to mind.

If a tournament schedule matrix exists it must be within a partition of $m!^{(n-1)}(n-1)!$ variations. There is always at least one solution partition (the undiscussed cyclic algorithm guarantees that). I do not know if there is only one solution partition or what conditions might be required for multiple solution partitions.

$S_T$ is a tournament schedule if and only if the $(i', j')$'s of $E$ of $s_{i1} \cdots s_{iM}$ are unique for each $i = 1..n-1$.

Question: Roughly, what does the algorithm do?

Consider a binary value $n$ bits longs. Each bit corresponds to an item. The value is 2 raised to the power of one less than the item number.

| *Item Number* | *Binary Value* |
|---|---|
| 1 | 000001 |
| 2 | 000010 |
| 3 | 000100 |
| 4 | 001000 |
| 5 | 010000 |
| 6 | 100000 |

Let GROUP be an $n$ bit value that tracks items planned for a group. When all the bits of GROUP are on then all the items have been scheduled for a group.

Consider an enumeration vector of item pairs. Each element of the vector has two fields *One* and *Two*. The fields contain the items that are paired. The enumeration vector is the pool from which pairs are taken from and tested for suitability in the current group.

| *Pair* | *One* | *Two* | *Items* | *One OR Two* |
|---|---|---|---|---|
| 1 | 000001 | 000010 | 1,2 | 000011 |
| 2 | 000001 | 000100 | 1,3 | 000101 |
| 3 | 000001 | 001000 | 1,4 | 001001 |
| 4 | 000001 | 010000 | 1,5 | 010001 |
| 5 | 000001 | 100000 | 1,6 | 100001 |
| 6 | 000010 | 000100 | 2,3 | 000110 |
| 7 | 000010 | 001000 | 2,4 | 001010 |
| 8 | 000010 | 010000 | 2,5 | 010010 |
| 9 | 000010 | 100000 | 2,6 | 100010 |
| 10 | 000100 | 001000 | 3,4 | 001100 |
| 11 | 000100 | 010000 | 3,5 | 010100 |
| 12 | 000100 | 100000 | 3,6 | 100100 |
| 13 | 001000 | 010000 | 4,5 | 011000 |
| 14 | 001000 | 100000 | 4,6 | 101000 |
| 15 | 010000 | 100000 | 5,6 | 110000 |

GROUP is built up incrementally by selecting pairs from the enumeration vector and testing them. If they ‘fit’ they are added to a plan vector. If they don’t fit, the next pair in the enumeration vector is tested. If the enumeration vector is exhausted without a pair being added to the plan vector, then the last pair in the plan vector is ‘unplanned’ and the hunt continues using the next pair of the enumeration vector.

When a new pair is under consideration the ONE and TWO values of the pair are AND'd with GROUP. A non-zero result means one of the items in the pair is already planned and thus the pair is not suitable. A zero result means neither item has been planned and the pair can be added to the plan vector.
When a pair is planned (ONE or TWO) is OR'ed to GROUP.
When a pair has to be unplanned NOT (ONE OR TWO) is AND'ed to GROUP.

Sample of the sequence number generator for the values of $p_{ij}$

| n=6 | | j | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| i | 0 | | 0 | | | | | |
| | 1 | | . | 1 | 2 | 3 | 4 | 5 |
| | 2 | | . | . | 5 | 6 | 7 | 9 |
| | 3 | | . | . | . | 8 | 9 | 12 |
| | 4 | | . | . | . | . | 10 | 14 |
| | 5 | | . | . | . | . | . | 15 |
| | 6 | | . | . | . | . | . | . |

Problem space

| n | n-1 | m | np | np! / (m!^(n-1) (n-1)!) | # of partitions ? | # in partition ? |
|---|---|---|---|---|---|---|
| 2 | 1 | 1 | 1 | 1<br>( 1 ) ( 1 ) | 1 | 1 |
| 4 | 3 | 2 | 6 | 6 5 4 3 2<br>( 2 2 2 ) ( 3 2 ) | 15 | 48 |
| 6 | 5 | 3 | 15 | 15 14 13 12 11 10 9 8 7 6 5 4 3 2<br>( 6 6 6 6 6 ) ( 5 4 3 2 ) | 1,401,400 | 933,120 |
| 8 | 7 | 4 | 28 | 28 27 26 25 24 23 22 21 20 19 18 17<br>16 15 14 13 12 11 10 9 8 7 6 5 4 3 2<br>( 24 24 24 24 24 24 24 ) ( 7 6 5 4 3 2 ) | 13,189,599,057,009,400 | 23,115,815,976,960 |
| ... | | | | | | |

Integer factorization
Prime power matrix

| n\p | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | . | | | | | | | | |
| 2 | 1 | | | | | | | | |
| 3 | | 1 | | | | | | | |
| 4 | 2 | | | | | | | | |
| 5 | | | 1 | | | | | | |
| 6 | 1 | 1 | | | | | | | |
| 7 | | | | 1 | | | | | |
| 8 | 3 | | | | | | | | |
| 9 | | 2 | | | | | | | |
| 10 | 1 | | 1 | | | | | | |
| 11 | | | | | 1 | | | | |
| 12 | 2 | 1 | | | | | | | |
| 13 | | | | | | 1 | | | |
| 14 | 1 | | | 1 | | | | | |
| 15 | | 1 | 1 | | | | | | |
| 16 | 4 | | | | | | | | |
| 17 | | | | | | | 1 | | |
| 18 | 1 | 2 | | | | | | | |
| 19 | | | | | | | | 1 | |
| 20 | 2 | | 1 | | | | | | |
| 21 | | 1 | | 1 | | | | | |
| 22 | 1 | | | | 1 | | | | |
| 23 | | | | | | | | | 1 |
| 24 | 3 | 1 | | | | | | | |
| 25 | 1 | | 1 | | | | | | |

Factorial factorization
Prime power matrix

| n\p | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | |
| 2 | 1 | | | | | | | | |
| 3 | 1 | 1 | | | | | | | |
| 4 | 3 | 1 | | | | | | | |
| 5 | 3 | 1 | 1 | | | | | | |
| 6 | 4 | 2 | 1 | | | | | | |
| 7 | 4 | 2 | 1 | 1 | | | | | |
| 8 | 7 | 2 | 1 | 1 | | | | | |
| 9 | 7 | 4 | 1 | 1 | | | | | |
| 10 | 8 | 4 | 2 | 1 | | | | | |
| 11 | 8 | 4 | 2 | 1 | 1 | | | | |
| 12 | 10 | 5 | 2 | 1 | 1 | | | | |
| 13 | 10 | 5 | 2 | 1 | 1 | 1 | | | |
| 14 | 11 | 5 | 2 | 2 | 1 | 1 | | | |
| 15 | 11 | 6 | 3 | 2 | 1 | 1 | | | |
| 16 | 15 | 6 | 3 | 2 | 1 | 1 | | | |
| 17 | 15 | 6 | 3 | 2 | 1 | 1 | 1 | | |
| 18 | 16 | 8 | 3 | 2 | 1 | 1 | 1 | | |
| 19 | 16 | 8 | 3 | 2 | 1 | 1 | 1 | 1 | |
| 20 | 18 | 8 | 4 | 2 | 1 | 1 | 1 | 1 | |
| 21 | 18 | 9 | 4 | 3 | 1 | 1 | 1 | 1 | |
| 22 | 19 | 9 | 4 | 3 | 2 | 1 | 1 | 1 | |
| 23 | 19 | 9 | 4 | 3 | 2 | 1 | 1 | 1 | 1 |
| 24 | 22 | 10 | 4 | 3 | 2 | 1 | 1 | 1 | 1 |
| 25 | 23 | 10 | 5 | 3 | 2 | 1 | 1 | 1 | 1 |