Constructing Optimal Schedules for certain Tournaments using Tabu Search

Luis B. Morales [1] *Felipe Maldonado* [2]

[1]IIMAS, Universidad Nacional Autónoma de México
Apdo. Postal 70-221, México, DF
04510, Mexico

[2]ESIT, Instituto Politécnico Nacional

Abstract

In this paper we describe a scheduling problem for certain types of tournaments. It is formulated as a discrete minimization problem. Using a tabu search method we solve various problem instances of different size and complexity. The influence of the tabu list size on the performance of the algorithm is studied. Tabu search is also compared with previous optimization techniques, which are available in related literature.

Keywords: Scheduling problem, discrete optimization, tabu search.

1 Introduction

A number of exact and heuristic methods of constructing schedules for different types of tournaments have been described in the Operation Research literature. However, the scheduling problem considered in this work has been poorly analyzed. The problem can be stated as follows: A sports club organizes a tournament which will be played during r meetings among t teams. At each meeting teams are divided into g groups of k teams ($t = kg$). Each of the k teams then plays against the remaining $k-1$ teams in its group. The club has requested a tournament schedule with the following property: The number of times any team competes against any opposing team should be the same for all teams. If no such schedule exists, then the club requests at least one where the number of times any team plays against any other team is as close as possible to the same for all teams.

There are $\binom{k}{2}$ games in each group and $g\binom{k}{2}$ games per meeting. Hence, the number of matches in the tournament will be $rg\binom{k}{2}$. Since each team plays r times against $k-1$ opposing teams, there are $r(k-1)$ potential opposing teams. Each team has

$t-1$ different possible opponents, because no team plays itself. So the club requires a schedule which ensures that the number of times any team plays against any other team is very close to $\lambda = r(k-1)/(t-1)$ (the "ideal" number of times that two teams should meet).

The instance problem $t = 12$, $r = 8$, $g = 3$ and $k = 4$ was presented in [3]. The authors studied this instance as an optimization problem and tried to solve it. They used the following discrete optimization techniques: a greedy algorithm, branch and bound, steepest descent, and simulated annealing. None of these methods produced a schedule required by the sports club. They also dealt with this instance as the construction of an experimental incomplete block design, but the required design is not available in the catalogues. In general, as we will see in the next section, there is a correspondence between the scheduling problem and the construction of a regular resolvable partially balanced incomplete block design with two distinct concurrences (see [6] for the definition of these designs). In [5] this instance was constructed using a hill-climbing technique. The others instance problems studied in [5] were constructed using techniques other than optimization.

The aim of this paper is the construction of required schedules using a tabu search technique (TS). In Section 2, we formulate the scheduling problem as a discrete optimization problem. For this problem, it is easy to find the value of an optimal solution but difficult to find an optimal solution. Section 3 describes the implementation of a TS method for this problem. TS was tested on 14 problem instances of different size and complexity. The numerical experiences obtained from TS will be given in Section 4. The influence of the tabu list length on the performance of the algorithm is also analyzed. The results were compared with those from the literature about optimization methods. Finally, conclusions are given in the last section.

2 Two Approaches to Solving the Scheduling Problem

In order to enumerate all possible schedules, we labelled the g groups at each meeting as $A, B, C, \ldots$, where each group contains k teams. The number of all assignments of the $t(= gk)$ teams to these g groups is given by the multinomial coefficient $\binom{t}{k_1 \cdots k_g}$ where $k_i = k$ for $i = 1, \ldots, g$. But since the $g!$ labels on groups are arbitrary, the number of possibilities for each meeting is the multinomial coefficient divided by $g!$, this is, $p = t!/gk!g!$. Since there are r meetings in the tournament and the same schedule can be chosen at different meetings, the number of all possible schedules is p^r. Therefore, it is not computationally feasible to use the exhaustive method for a problem of significant size.

The scheduling problem can be seen as the arrangement of t teams in $b = rg$ groups (blocks) in such way that

(1) each block contains k teams

(2) each team occurs in r blocks

(3) every pair of teams occurs together in λ blocks, and

(4) the blocks are divided into r sets such that each team occurs exactly once in the blocks of any set. Each set contains g blocks.

Notice that λ is the number of times that any one team plays against any other, and b is the total number of groups formed during the r meetings.

An arrangement of t elements in b blocks that satisfies conditions (1) and (2) is called an *incomplete block design*. Designs which satisfy (3) and (4) are called *balanced* and *resolvable* design, respectively. The terms t, b, r, k, λ are known as the *parameters* of a balanced incomplete block design. It is not hard to see that

$$bk = tr, \qquad r(k-1) = l(t-1).$$

In order to see the scheduling problem as a discrete optimization problem, we define the *competition matrix* of a schedule as a $t \times t$ matrix C where $C_{ij} = C_{ji}$ is the number of times that team i plays against team j in this schedule and $C_{ij} = 0$ for $i = j$. The scheduling problem can therefore be formulated as the problem of finding a schedule whose competition matrix C minimizes the following *cost function*:

$$f = \sum_{i<j}^{t} (C_{ij} - \lambda)^2. \tag{1}$$

Since matrix C is symmetric, only $t(t-1)/2$ entries below (or above) the main diagonal need to be examined. Note that $f \geq 0$ and $f = 0$ if and only if $C_{ij} = \lambda$ for all $i \neq j$. So, the number of times any team competes against any other team is the same for all teams.

In particular, for the instance $t = 12$, $r = 8$, $g = 3$ and $k = 4$, each team plays $r(k-1) = 24$, but there are only $t - 1 = 11$ different opponents. So, the value of λ $(= 2.18\ldots)$ is not an integer. Hence a required schedule would have each team playing with 9 other teams twice and with 2 other teams three times. Thus, each row of its competition matrix C should consist of nine 2's and two 3's (with 0's on the diagonal). Taking $\lambda = 2$ in (1), the only non-zero summands in the cost function f –for this schedule– are those for which $C_{ij} = 3$. In consequence, the cost of any requested schedule is $12 = 12 \times 2/2$.

The purpose of this paper is to study the construction of schedules where $\lambda = r(k-1)/(t-1)$ is not an integer, but there are integers l_1, l_2, n_1 and n_2 such that

$$r(k-1) = \lambda_1 n_1 + \lambda_2 n_2.$$

More specifically, λ_1 is the integer part of $r(k-1)/(t-1)$, $\lambda_2 = \lambda_1 + 1$ and $t - 1 = n_1 + n_2$. Note that $n_2 = r(k-1) - \lambda_1(t-1)$. For these values, each pair of teams plays λ_1 or $\lambda_1 + 1$ times. Hence, each row of the competition matrix C of a requested schedule should have n_1 λ_1's, n_2 λ_2's and 0 on the diagonal. It is easy to see that the cost of these schedules is $tn_2/2$ when we take $l = l_1$ in (1).

The corresponding incomplete block design for the scheduling problem with the above property is not a balanced design, because every pair of teams occurs together in either λ_1 or λ_2 blocks. If $l_2 = l_1 + 1$ the design is said to be *regular*. Thus, in the language of blocks design, the scheduling problem is equivalent to constructing a regular resolvable partially balanced incomplete block design with two concurrences. Although there are procedures for constructing partially balanced designs, these procedures are usually restricted to very specific classes of problems, see for example [2].

Let $f_0 = b\binom{k}{2} - l_1\binom{t}{2}$. It is not hard to check that for regular designs, $f_0 = tn_2/2$. In [1] it was proved that for any design, $f \geq f_0$ and $f = f_0$ if and only if the design is a regular partially balanced incomplete block design (taking $l = l_1$ in (1). This result allows one to formulate the scheduling problem as an minimization problem with the cost function:

$$f = \sum_{i<j}^{t} (C_{ij} - \lambda_1)^2.$$

It follows that $tn_2/2$ is the global minimum of f whenever an appropriate schedule does exist.

3 Implementation of Tabu Search

TS is an iterative heuristic procedure for optimization. It has been designed to overcome local optimality. It is distinguished from other methods because it incorporates a tabu list of transitions (moves) that forbids the reinstatement of certain attributes of previously visited solutions. These forbidden moves are called *tabu*. For a more detailed presentation of TS see [4].

Let us now describe how we used tabu search to find an optimal schedule. In our approach, a *feasible solution* would be any schedule, that is, the r assignments of t teams into g groups of k teams each. A *move* is a transition from one schedule to another. An *attribute* of a move is a triple (m, i, j) which exchanges two teams i and j from different groups in the same meeting m. The *value of a move* is the difference between the cost function value before and after the move. At each iteration the best move is chosen, even if it does not improve the cost function. The search is carried out on all the moves (m, i, j) such that teams i and j do not belong in the same group. The number of possible moves (m, i, j) in each iteration is $r\binom{g}{2}kk$, because there are r different meetings, and in each one there are $\binom{g}{2}$ ways to choose the two groups

involved in the change, and k ways to choose a team from each of the two groups.

The only entries of matrix C that change, after the move (m, i, j), are those belonging to the ith or jth row, or column. This allows us to calculate the cost of a move in $O(t)$ operations. By contrast, the entire function f can be calculated in $O(t^2)$, because the order of C is t. Note that complexity has been reduced by a factor of $1/t$.

To prevent cycling, a *tabu list* T of length $|T|$ is constructed and updated circularly during the process. At each iteration we introduce the best found move (m, i, j) into the tabu list. This means that for $|T|$ iterations of TS, the teams i and j cannot be exchanged in the meeting m. The tabu status of a move may be dropped whenever it gives a cost function value that is strictly better than the best obtained so far. This is the *aspiration level criterion*.

The search process will be stopped if an optimal solution (or schedule) is reached, or if the number of iterations without improving the best solution is greater than a $nimax$ limit.

4 Computational Experience

The tabu search method was tested on 14 problem instances with parameters $t = 12, 16, 20, 24$; $7 \leq r \leq 24$ and $k = 4, 5, 6$. From our experiments, we see that there are instances which are very easy to solve (always less than 150 iterations), whilst others are difficult (average of under 200 iterations) and some others are very hard to solve (average of over 1000 iterations). Finally, from the 14 instances tested, there were only two that the TS procedure could not solve optimally. All TS runs were carried out with a random initial solution. The TS algorithm was implemented in C programming language, and all computations were executed on a 66MHz 488 Pentium PC.

In order to discover the influence of $|T|$ and $nimax$ on the performance of tabu search, a number of runs were performed using different values of $|T|$ and $nimax$ on each instance tested. The best results obtained by TS are given in Table 1. Each row in the table corresponds to an instance. Columns 2-8 show the parameters of the instances. Column 10 gives the best tabu size. Columns 11, 12 and 13 respectively show the average number of iterations where an optimal solution was found (ITO), the percentage of runs with an optimal solution (POS) and CPU time.

Table 1. Computational Results of Tabu Search

#	t	r	k	g	λ_1	λ_2	n_1	n_2	T	ITO	POS	CPU time (s)
1	12	8	4	3	2	3	9	2	27	192	100	1.19
2	12	9	4	3	2	3	6	5	6	19	100	0.13
3	12	10	4	3	2	3	3	8	18	141	100	3.55
4	16	7	4	4	1	2	9	6	6	22	100	0.27
5	16	8	4	4	1	2	6	9	6	31	100	0.43
6	16	9	4	4	1	2	3	12	39	2010	32	31.81
7	16	11	4	4	2	3	12	3	30	1317	56	25.44
8	16	12	4	4	2	3	9	6	6	38	100	0.80
9	20	7	4	5	1	2	17	2	33	1908	54	44.04
10	20	8	4	5	1	2	14	5	20	159	100	4.21
11	20	10	5	4	2	3	17	2	-	-	0	-
12	20	12	5	4	2	3	9	10	27	1907	42	78.99
13	24	7	6	4	1	2	11	12	-	-	0	-
14	24	19	4	6	2	3	12	11	6	71	100	7.35

For each tabu length: $1 \leq |T| \leq 10$, TS was run 50 times using $nimax = 100$ on the easy problem instances. TS always reached the global minimum $tn_2/2$ for any tabu list size greater than 6. Rows 2, 4, 5, 8 and 14 of the Table 1 give the best results for the easy instances.

In each difficult instance problem, TS was run 50 times using $nimax$ values of 500, 700 and 1000 and varying the tabu size from 9 to 60 with a step size of 3. The best results obtained for these instances are given in rows 1, 3 and 10 of Table 1.

Table 2 contains the influence of the tabu list size on the performance of TS for the difficult instance $(12, 8, 4, 3)$. Row 1 gives the values used for the tabu size. Row 2 shows the percentage of these runs in which the TS method reached an optimal solution for each value of $|T|$, using $nimax = 700$. Row 3 gives the average number of iterations where an optimal solution was found. The table shows that the best tabu length was about 27. For small tabu size ($t \leq 12$), there was a high probability that cycling would be detected. When we used $nimax = 1000$, TS always reached an optimal solution for any tabu list length greater than 24, while, for $nimax = 500$, the TS algorithm never achieved 100% of successful runs.

Table 2. Performance of TS vs length of tabu list for instance $(12, 8, 4, 3)$

T	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
POS	38	52	70	76	94	96	100	98	96	96	100	100	96	96	100	98	96	100
ITO	81	95	144	168	169	186	192	215	179	208	192	198	216	193	175	198	174	219

Although some of our instance problems tested were also presented in [5], we only compared our results with those obtained using an optimization method. The only scheduling problem previously solved through discrete optimization is instance $(12, 8, 4, 3)$. Table 3 provides a comparison of six heuristic methods used to seek an optimal schedule. Columns 2, 3 and 4 respectively show the best cost found, the number of cost function evaluations, and the percentage of successful runs where the global minimun 12 was reached. None of these methods, except the last two, produced an optimal solution. Nevertheless, TS appears to be superior to hill-climbing algorithms, with respect to successful runs as well as the number of evaluations of the cost function (see Columns 2, 3 and 4). An optimal schedule obtained by TS and its

competition matrix for instance $(12, 8, 4, 3)$ are given in Figure 1.

Table 3. Comparison of Methods for instance $(12, 8, 4, 3)$

Method	No. of Evaluations	Lowest Cost	% successful runs
Greedy Algorithm [3]	40397	22	0
Partial Branch and Bound [3]	100000	20	0
Steepest Descent [3]	75000	14	0
Simulated Annealing [3]	50000	14	0
Hill-climbing technique [5]	1000000	12	2
Tabu Search	73728	12	100

Schedule
(entries are group label assigned to team)

Team #	Meeting # 1	2	3	4	5	6	7	8
1	B	C	C	B	B	A	B	B
2	A	B	C	C	C	B	B	A
3	C	A	C	B	A	B	C	C
4	A	A	A	B	C	C	A	B
5	B	A	A	A	C	A	C	A
6	C	B	B	B	B	A	A	A
7	C	C	B	C	C	C	C	B
8	A	A	B	C	A	A	B	C
9	B	C	A	C	B	B	A	C
10	B	B	B	A	A	B	A	B
11	A	B	C	A	B	C	C	C
12	C	C	A	A	A	C	B	A

Competition Matrix
(entries are # of times team plays opponent)

Team #	Opponent # 1	2	3	4	5	6	7	8	9	10	11	12
1	0	2	2	2	2	3	2	2	3	2	2	2
2	2	0	2	2	2	2	2	3	2	2	3	2
3	2	2	0	2	2	2	2	3	2	2	3	2
4	2	2	2	0	3	2	3	2	2	2	2	2
5	2	2	2	3	0	2	2	2	2	2	2	3
6	3	2	2	2	2	0	2	2	2	3	2	2
7	2	2	2	3	2	2	0	2	2	2	2	3
8	2	3	3	2	2	2	2	0	2	2	2	2
9	3	2	2	2	2	2	2	2	0	3	2	2
10	2	2	2	2	2	3	2	2	3	0	2	2
11	2	3	3	2	2	2	2	2	2	2	0	2
12	2	2	2	2	3	2	3	2	2	2	2	0

Figure 1. An optimal schedule and its competition matrix for instance $(12, 8, 4, 3)$

Finally, for each very hard instance, we used *nimax* values of 1000 and 1500, and for each of these values the range of $|T|$ was from 9 to 60 with step size of 3. TS was run 50 times for each value of *nimax* and $|T|$. Table 4 shows the percentage of these runs in which TS reached an optimal solution for the very hard instance $(16, 9, 4, 4)$, and the average number of iterations where an optimal solution was found, using $nimax = 1500$. For this instance, the best tabu sizes seem to be integers somewhere between 24 and 51.

Table 4. Performance of TS vs length of tabu list for instance $(16, 9, 4, 4)$

$\|T\|$	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60
POS	4	8	12	18	16	20	4	12	14	10	10	12	12	14	20	16	10	10
ITO	243	1156	1179	1227	1124	987	276	895	924	686	1214	888	1054	1032	1075	1110	1167	973

In each very hard instance, we observed irregular behavior, in the percentage of successful runs of TS, in the best range of $|T|$ (see row 2 of Table 4), while, this behavior was regular in the difficult instances (see row 2 of Table 2). Additionally experimentation for each very hard instance using $nimax = 3000$ in the best range of tabu size showed that the percentage of successful runs can be increased. Nevertheless, the behavior remains irregular. Rows 6, 7 and 9 of Table 1 give the best result for very hard instances, using $nimax = 3000$.

Even starting from different initial solutions and modifying the values of $|T|$ and *nimax*, TS did not produce the theoretical optimal solution for problem instances

$(20, 10, 5, 4)$ and $(24, 7, 6, 4)$.

Comparing the results obtained from instances 6 with 7; 12 with 13; and 13 with 14, we can see that instances where r is similar to k seem to be harder than those instances where r is relatively greater than k. We also have from computational results that the difficulty of a instance seems to increase when the absolute difference between n_1 and n_2 grows. A problem may be solved in a few iterations when r is significantly greater than k and $|n_1 - n_2|$ is small, see instances 2, 4, 5, 8 and 14. On the contrary, if at least one of theses conditions fails, the instance seems harder to solve, and in some others TS does not produce the theoretical optimal solution, see for example instances 6, 7, 9, 11 and 13.

In our experiments we have also seen that, for a fixed tabu length, the number of necessary iterations to find a good solution in all instances, strongly depends on the initial solution. For example, an optimal solution for instance $(12, 8, 4, 3)$ was found in 9 iterations with an initial solution, while about 400 iterations were required to reach an optimal schedule using a different initial schedule.

5 Conclusions

In this paper we have defined the problem of constructing appropriate schedules for a specific type of tournament. The problem was formulated as a discrete minimization problem. We have presented an adaptation of tabu search to seek a good solution. The influence of the length of the tabu list on the behavior of the method was also analyzed. Of the 14 instances tested, TS was able to solve 12, many of them with a frequency of 100% of successful runs. For instance $(12, 8, 4, 3)$, in particular TS gave better results in efficiency and the number of evaluations of the cost function than previously reported in earlier papers. From the experimental results, we have seen that an instances is relatively easy to solve when r is significantly greater than k and $|n_1 - n_2|$ is small,

Likewise, we have seen that the construction of an optimal schedule is equivalent to proving the existence of a regular resolvable partially balanced incomplete block design with two concurrences. Some of the designs constructed were not available in the catalogues of incomplete block designs.

References

[1] Brown, R.B., "Nonexistence of a regular graph design with $r = 17$ and $k = 6$", *Discrete Math.* **68** (1988) 54-64.

[2] Clatworthy, W.H., *Table of Two-Associate-Clas Partially Balanced Design*, Nat. Bureau of Stand., Appl. Math. Series 63 (1973).

[3] Elenbogen, B.S. and B.R. Maxim, "Scheduling a bridge club (a case study in discrete optimization)", *Math. Mag.* 65 (1992), 18-26.

[4] GLOVER, F., "Tabu Search" Part I, *ORSA Journal on Computing* 1 (1989), 190-206.

[5] KREHER, D.L, G.F. ROYLE AND W.D. WALLIS, "A family of resolvable regular graph designs", *Discrete Math.* 156 (1996) 269-272.

[6] JARRET, R.G., "Definitions and properties for m-concurrence designs", *J. R. Statis. Soc.* B. 45 (1983), 1-10 .