Applying Heuristic Methods to Schedule Sports Competitions on Multiple Venues Extended Abstract

E.K. Burke[1], D. de Werra[2], J.D. Landa Silva[1], and C. Raess[2]

[1] Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science and IT, University of Nottingham, UK
{ekb, jds}@cs.nott.ac.uk
[2] Chaire de Recherche de Operationnelle
EPFL, Lausanne, Switzerland
{dewerra, raess}@dma.epfl.ch

1 Overview

Scheduling sports competitions is often a difficult task because it is usually very difficult to construct schedules that are considered to be fair by all competitors. In addition, the schedules normally need to satisfy a considerably large number of additional requirements and constraints. Most of the sports scheduling problems that have been tackled in the literature refer to competitions in which a venue is associated with each competitor (home ground). In these cases, selecting the venue in which each tie will be played is not an issue because this is defined by the status, home or away, of the competitors. Scheduling problems in most sports leagues (football, baseball, rugby, cricket, etc.) fall into this category because each team has its own venue. However, many other sports competitions take place on a set of venues that are neutral to all competitors. This is the case in some international competitions (such as the football world cup, and the Wimbledon tennis tournament) and in some recreational leagues. In these cases, choosing the venue to play each tie is part of the scheduling process and this often makes the problem more difficult to solve. Here, we propose the application of heuristic methods for constructing schedules for this type of sports competition and also the use of metaheuristics for improving the quality of a given schedule. Initial experiments demonstrate the promise of these approaches.

2 Problem Description

In the last three decades or so, the automated construction of sports competition schedules has received considerable attention [1]. Among the approaches that have been proposed there are exact algorithms (including methods based on combinatorial design theory) (e.g. [2, 3]) and heuristics (e.g. [5, 6, 9]). We are interested in the problem in which a set of N teams must compete on a set of S neutral venues over T timeslots [8]. A feasible competition schedule should be constructed so that the assignment of venues to the matches is as balanced and

as fair as possible. A feasible schedule must satisfy the hard constraints: only one match can take place on a given venue at a given timeslot, and each team can compete in exactly one venue in a given timeslot. A particular case is when the problem is balanced. By this, we mean that the number of competitors denoted by N equals the number of timeslots denoted by T and it is twice the number of venues denoted by S, i.e. $N = T = 2S$. Then, each team competes N times and therefore, each team competes against one of the other teams exactly twice (this can be called the repeated match). In each of the T schedule timeslots, all teams in the league must be competing simultaneously. There are three soft constraints that should be satisfied. First, each team must play against each of the other teams at least once. Second, each team must compete in each venue exactly twice. Finally, any pair of teams should not compete against each other more than one time on the same venue. Then, the problem is to find whether a feasible schedule that satisfies all the soft constraints exists.

3 Related Work

Urban and Russell tackled the above scheduling problem using integer goal programming [8]. They reported optimal results (i.e. perfectly balanced schedules) for the cases in which N equals 4, 6, 8 and 10. They also reported best-known results for problems with N equal to 12, 14 and 16. They observed that the branch-and-bound process could easily find solutions for some cases (e.g. $N = 8$ or $N = 10$) while it was unable to find a solution satisfying all goals in other cases (e.g. $N = 6$) after a considerably large amount of computation time. They also noted that the branch-and-bound approach was too time consuming for larger problems (i.e. $N > 10$). For those cases, they proposed to run the branch-and-bound process for a fixed number of iterations (alternatively a fixed amount of computation time) followed by a post-processing phase that seeks to improve the quality of the schedule (see [8] for details).

Recently, de Werra et al. designed constructive algorithms to solve some specific cases of this problem [4]. Specifically, they described constructive approaches to generate perfectly balanced schedules for five cases:

1. $N = 12$
2. $N = 14$
3. $N = 2S = 2^p$ where $p \geq 2$, i.e. $N = 4, 8, 16 \ldots$
4. $N = 2S = 2 \bmod 8$, i.e. $N = 10, 18, 26 \ldots$
5. $N = 2S = 4 \bmod 16$, i.e. $N = 20, 36, 52 \ldots$

As noted by Urban and Russell in [8], the combinatorial nature of the problem limits the applicability of the branch-and-bound approach to small problems ($N \leq 10$). The construction methods designed by de Werra et al. in [4] apply only to the five specific cases listed above. There are several cases of this problem for which no optimal solution is yet known (e.g. $N = 22$). Then, it would be interesting to find a more general approach to generate perfectly balanced schedules for a wider number of cases of this problem. Moreover, such an approach could

also be useful to tackle some variants of the problem that are also of practical interest. For example, the case in which there are fewer than $N/2$ venues and therefore, not all teams can compete simultaneously in each timeslot. Another important case is when a 'home' team is required in each match for administrative purposes. Then, the balance between 'home' and 'away' matches must be considered too. All of these observations motivated our interest to investigate heuristic techniques to tackle the problem of scheduling sports competitions on multiple venues. Our initial efforts are focused on the balanced case, i.e. when $N = T = 2S$.

4 Heuristic Methods and Preliminary Results

The metaheuristic approach implemented here is shown in fig. 1. This is a hybrid algorithm that was originally proposed to tackle the space allocation problem in academic institutions [7] and has been adapted for the problem tackled here.

Figure 1. The hybrid metaheuristic implemented

1.Generate the initial schedule x.
2. Set $\rho \approx 0.95$, $\alpha \approx 0.97$, $R_{iter} \approx 0$, $R_{step} \approx NS$, $D_{step} \approx 5$, *iterations*=0, $FA_{max} \approx 3NS$.
3. For R_{step} iterations do,
  3.1. Find a candidate solution x' by using the heuristic H_{LS}, then evaluate x' and if it is better than x then accept it as the new current solution.
4. Find a candidate solution x' using the heuristic H_{LS}.
  4.1.Calculate the fitness variation ΔF between x and x', if x' is better than x then accept it as the new current solution.
  4.2. If the x' is not better than x,
    4.2.1. If $\rho \leq 0.001$ then reject x, make $R_{iter} = R_{iter} + 1$ and if (R_{iter} mod R_{step}) equals zero then make $\rho \approx 0.95$ and $R_{iter} = 0$.
    4.2.2. If $\rho > 0.001$ and $\rho >$ random$[0, 1]$ then make $x = x'$, otherwise reject x'.
  4.3. *iterations*=*iterations*+1.
  4.4. If *iterations* mod D_{step} equals zero then make $\rho = \rho\alpha$.
5. If not better candidate solution x' was found,
  5.1. Increment *failed move attempts*.
  5.2. If *failed move attempts* > FA_{max} then apply the *heavy mutation operator* to x and make *failed move attempts*=0.
6. If the termination criterion is satisfied stop, otherwise go to 4.

We have implemented several constructive heuristics to generate initial schedules (Step 1 in fig. 1). These heuristics serve to assess the effect that the quality of the initial solution has on the performance of the metaheuristic approach. These constructive heuristics are briefly described below. We decided to represent a competition schedule using a matrix M of size $N \cdot S$ where each cell

$M(t, s)$ contains the match (i,j) to be played on timeslot t at venue s.

Random. First, we generate the list of $(N-1) \cdot S$ matches of the form (i, j) where $i \neq j$ for $i, j = 1, 2 \ldots N$. Then, we generate S repeated matches of the form $(i, i+1)$ for $i = 1, 3 \ldots N-1$. Finally, each of the $N \cdot S$ matches is scheduled on a cell $M(t, s)$ selected at random. This heuristic generates many infeasible solutions, i.e. schedules with the same team scheduled in more than one venue in the same timeslot.

Improved Random. This is very similar to the previous heuristic, but when scheduling each match, the first free location $M(t, s)$ that keeps the solution feasible is selected if such a value of $M(t, s)$ exists. Otherwise, any random location is used to schedule the match. This heuristic produces initial schedules of higher quality than the previous heuristic, but of course, is more time consuming.

Progressive Feasible. This is a general constructive heuristic that generates feasible initial schedules using combinatorial design. Two variants are implemented: when S is even and when S is odd. The approach splits the teams into two groups $C_1 = [1 \ldots N/2]$ and $C_2 = [(N/2) + 1 \ldots N]$. Then a partial schedule is constructed containing all the external matches, i.e. ties between a team in C_1 and a team in C_2. The schedule is completed by scheduling all the internal matches, i.e. ties between two teams in the same group.

The heuristic H_{LS} is used to explore the neighbourhood of the current solution (Steps 3 and 4 in fig. 1). This heuristic uses one type of neighbourhood structure, the *swap* between two matches selected at random. The H_{LS} heuristic selects the best of a number of $k \approx (N \cdot S)/3$ *swap* moves. A *heavy mutation operator* is implemented in order to disturb the schedule if no improvements have been achieved for a number of iterations (Step 5.2 in fig. 1). First, this operator identifies those ties that contribute the most to the cost of the current solution. Then, these ties are removed from the schedule (a maximum of $(N \cdot S)/5$ ties are permitted to be unscheduled in this way). Finally, in random order, each of the scheduled ties is re-scheduled to the best available timeslot until the schedule is complete.

We have carried out experiments applying the algorithm described above for problems up to $N = 20$. The algorithm is capable of producing perfectly balanced schedules for problems of size N equal to 8,10 and 12. It also finds the optimal solutions for the cases where N equals 4 and 6. For the rest of the cases, so far, the algorithm is able to generate feasible schedules with low cost (between 8 and 18).

References

1. Easton K., Nemhauser G., Trick M.: Sports scheduling. In: Joseph Y-T. Leung (ed.) Handbook of scheduling: algorithms, models, and performance analysis. CRC Press, 2004.

2. de Werra D.: Minimizing irregularities in sports schedules using graph theory. Discrete applied mathematics. Vol. 2, 217-226, (1982).
3. de Werra D.: Some models of graphs for scheduling sports competitions. Discrete Applied Mathematics. Vol. 21, 47-65, (1988).
4. de Werra D., Ekim T., Raess C.: Construction of sports schedules with multiple venues. Internal report. Chaire de recherche operationnelle, EPFL, Lausanne Switzerland, November, (2003).
5. Ferland J.A., Fleurent C.: Computer aided scheduling for a sports league. INFOR 29, 14-24, (1991).
6. Henz M.: Scheduling a major college basketball conference - revisited. Operations research, Vol. 49, No. 1, 163-168, (2001).
7. Landa Silva J.D.: Metaheuristic and multiobjective approaches for space allocation. PhD thesis. School of computer science and IT, University of Nottingham, UK (2003).
8. Urban T.L., Russell R.A.: Scheduling sports competitions on multiple venues. European Journal of operational research. Vol. 148, 302-311, (2003).
9. Wright M.: Timetabling county cricket fixtures using a form of tabu search. Journal of the operational research society. Vol. 45, No. 7, 758-770, (1994).