Sports Scheduling: Algorithms and Applications

1. *Introduction*

We consider the practical task of creating a schedule for a sports tournament or league, a task in which differing levels of optimality can result in differences of millions of dollars for sports clubs, television networks, and other interested parties. Essentially all the sports scheduling papers that we studied discussed actual schedules created for some specific league. The algorithms used by schedulers depend on the complexity of the schedule and constraints for each problem. The definition of an "optimal" schedule varied, as did the set of constraints, but the theme of searching for an optimal or near-optimal schedule that meets constraints remained consistent. Some applications forced the authors to rebuild the schedule many times as constraints were added or clarified [Trick & Nemhauser]. As a result, optimization of their scheduling algorithm was critical to being able to produce new schedules for the league commissioner to review.

The simplest league scheduling examples were in the form of small round-robin tournaments where each team plays the other team one time at a single location. Additional constraints can turn a simple scheduling problem into an NP-complete one, in which case the size of the schedule input plays a big role in determining the right algorithmic approach. Sophisticated approaches for coping with NP-complete algorithms are needed when given a large number of teams, games, and constraints—such as the task of creating the schedule for major league baseball, where 30 teams each play 162 games over the course of several months. Constraints on the intra- and inter-division games, the number of home and away games, the lengths of road trips, and reasonable travelling requirements further complicate the Major League Baseball schedule [Yang, Huang, & Horne].

We discuss the sports league scheduling problem and a variety of real-life constraints, and the merits of various algorithmic approaches to create such schedules. We conclude with a brief analysis of a paper describing the algorithm used to schedule the Atlantic Coast Conference (ACC) college basketball league, and conclude with directions for future work and open problems in sports scheduling.

2. *The Problem: constraints and solutions*

The definition of the sports scheduling problem we are interested in goes roughly as follows: find an optimal schedule for a set of teams over a period of time that satisfies a set of particular constraints. In some scenarios, there is no feasible schedule, for others, there can be many, and depending on constraints, one may be optimal. One frustrating aspect of sports scheduling is that league requirements differ enough that techniques used for one league's schedule do not work for a different league. In response, there are some academic scheduling problems as such as the "Travelling Tournament Problem" (TTP), which asks for an optimal round robin schedule where every team plays each other team once at home and once away, while minimizing the distance travelled by any team and meeting some other constraints [Leong]. The existence of such problems allows various techniques to be

compared, which has some use, though it does not mean the techniques that work best for the TTP will work best for a real sports league.

The amount and type of scheduling constraints often determines the difficulty of the problem. Constraints can be applied globally to the entire schedule or locally to some portion (only in June, weekends, etc.). Consider the impact of sports league scheduling constraints as given below:

- Venue constraints: every game is either "home" or "away". Fairness of venue is often applied to the entire schedule and portions of the schedule, for example, correct proportion of home games on weekends, as well as in the first half of the season.
- Break constraints: a break is defined as consecutive home or away games (as opposed to "no breaks"—an alternating sequence of home and away games). Finding min-break schedules has been an active research topic, and yet a minimum break schedule is not appropriate for many scheduling applications, such as Major League Baseball.
- Time and Distance constraints: fairness of game times (avoid Friday night at 9PM followed by Saturday at 12 noon) and accounting for time zone differences, such as fairness to each team with respect to the time and cost of travelling between venues.
- Variety and carry-over: a good schedule has "variety". As an example, suppose each team plays every other team N times: intuitively, you would like to spread the N times over the course of the schedule while minimizing "carry-over". Carry-over is easiest explained with an example: suppose that the times when A plays team C, A has always just played team B the game before. Then if B is a very strong or weak team, it could have a "carry-over" affect on A's performance against team C.
- Hierarchal constraints: an example might be the division and conference structure of a league, where a team should play a certain portion of games in its own conference, and another portion within its own division, etc. Another example is application of time constraints in terms of per-month and per-week.
- Desired, fixed, or prohibited games: often relating to certain times of the season, for example, "Duke should play UNC in the last week of the season", or "don't match up the two conference football champions in the first two weeks of the season".
- Preferential constraints: constraints relating to such things as maximizing predicted television ratings, attendance, and other special preferences.

3. Algorithmic Alternatives

The variety of sports scheduling applications and constraints in turn leads to a wide range of sports scheduling algorithms. The algorithmic alternatives that we studied and will summarize include greedy algorithms, integer programming, constraint programming, and other approaches for hard problems.

Greedy algorithms work best for simple schedules. Consider a weekend tournament where it is expected for each team to play each other once. By 1847, it was shown by Kirkman that a round robin schedule exists for any 2n teams and 2n-1 time slots [Trick]. Such a schedule is typically found using either "the circle method" or greedy algorithms:

- The circle method numbers the teams from 1..2n, and then considers the set of teams and the set of time slots. In slot i, it plays team i against team 2n (the last team), and for all A and B other than teams i and 2n, team A plays B if index(A)+ index(B) $\equiv$ 2i (mod 2n-1).
- The greedy method matches teams lexicographically, where team i plays team j is marked "(i,j)", and (i,j) is scheduled before (i,k) if j<k, and (i,j) is before (k,l) if i<k. Games are repeatedly assigned into current slot or next feasible slot.

Both the greedy and circle methods are linear in the number of games and teams. "Simple" scheduling problems are also sometimes solved with combinatorial approaches such as "Latin Squares" or "Room Squares" [Dinitz]. Interestingly enough, suppose we add prohibitions of the form (k,i,j) to a scheduling problem with a greedy solution, that is, in game slot k, team i cannot play at j's home. Now the problem becomes NP-complete! [Schaerf] Harder scheduling problems are often solved using integer or constraint programming. A common approach seems to include first splitting the problem into subproblems. Authors seemed to use different subproblem breakdowns, but in general, each scheduling problem is reduced to 2-4 subproblems where each takes as input the solution sets found by the subproblem before it, and gives all solutions that meet its constraints to the subproblem after it.

The integer programming approaches that we studied split the league scheduling problem into the following high-level steps:

1. Use integer programming to find feasible home-away-bye patterns that meet the constraints on number of games.
2. Use integer programming to transform the "Home-Home-Away"-type patterns from step 1 into a pattern that could be a team's schedule, such as "Team 1 plays Team 2 at home, Team 3 at home, and Team 4 away".
3. Use enumerative approaches or integer programming to assign actual teams to the team numbers (e.g. Team 1 is the New York Yankees, etc.)

Some common constraints for integer programming models included:

- no team can play itself
- if team A plays at team B at time t, then team B plays A at home at time t
- a team can only play one other team at a time
- a team plays every other team once before it plays some team for a second time

Scheduling via constraint programming is very similar to integer programming, at least from our perspective. Constraint programs start with variables assigned to domains, and then the domains are gradually reduced by the constraints. If the domains can be reduced to singletons then a schedule has been found. In a paper by Leong, the steps used for solving a Travelling Tournament Problem via a Constraint Satisfaction Problem (CSP) were almost identical to the IP approaches we had studied, the only key difference being that the third stage was solved using a branch-and-bound technique rather than solving by enumeration.

In a study done by Michael Trick, constraint programming was clearly better than integer programming in terms of finding schedules with a constraint on the number of consecutive home or away games. At

the same time, when the constraint was changed to “prohibit home-away singletons”, that is, AHA or HAH sequences, it was very clear how to write the integer programming constraint and as such IP was much better. The take-home point seems to be that cleverness in writing constraints can impact the effectiveness of a solution, and poor performance of either integer or constraint programming can be due to poorly developed constraints.

The authors of the paper on the ACC basketball schedule did not win in a contest to compute the Major League Baseball schedule, losing to a group of researchers who used a simulated annealing approach. In a different publication, Trick claims that simulated annealing is difficult in scheduling problems due to lack of a natural neighborhood structure and difficulty determining what types of swaps improve the solutions. For extremely hard scheduling problems, approaches common to coping with NP-complete problems are used like branch-and-bound, backtracking, and evolutionary algorithms.

4. *A league scheduling approach for ACC Basketball*

We now turn and look at the ACC Basketball scheduling problem more in detail, as described in the paper “Scheduling a Major College Basketball Conference”, by George L. Nemhauser and Michael A. Trick published in 1997. This paper took the moderately difficult scheduling task of scheduling the 9-team ACC basketball league, where each team plays the other team twice during the season (once home and once away) with many constraints (television network requirements, travel requirements, maximized length of time between the two meetings of teams, etc). The algorithm used by Nemhauser and Trick relied heavily on integer programming and can be described at a high level as follows:

1) Find feasible home-away-bye patterns

A feasible pattern is one that could potentially be included in the schedule. At a minimum, it would include the appropriate number of home games, away games, and byes following the generic scheduling constraints such as maximum consecutive home/away games, etc. Since the number of potential patterns determined in the first step leads directly to the increased number of solutions in each future step, it is very important to generate enough valid patterns such that a reasonable number of schedules can eventually be created. However, it must be balanced with not over-generating patterns such that we result in too many schedules and significantly increased processing time with minimal (if any) gain. Given a small set of 9 teams, the number of potential patterns is much less than with a larger league and thus using enumeration, a reasonable number of patterns can be generated rather quickly. As the number of teams increase, this step would need to be reexamined for efficiency.

2) Find feasible pattern sets

Given the list of all feasible patterns, select the sets of 9 patterns such that all home, away, and bye games line up for a possible schedule. These sets will be determined using integer programming based on our list of constraints. The integer program in this case consists of 38 variables and 18 constraints and is solved by optimizing with respect to the objective listed in the paper and then applying a constraint, and re-optimizing until a solution is found. The result is 17 feasible pattern sets.

3) Find timetables

This step assigns numbers to the home-away-bye patterns using integer programming. For example, if we were writing a schedule for teams A,B,C, and D, then we might have the following transformation done in this step, where "1,2,3,4" are not yet assigned to teams A,B,C,D:

$1: HHA \rightarrow$ 1: 2, 3, @4 $2: AHA \rightarrow$ 2: @1, 4, @3)

$3: HAH \rightarrow$ 3: 4, @1, 2 $4: AAH \rightarrow$ 4: @3, @2, 1

Again, the global constraints (those that apply to all teams) must be enforced in this step so that when we assign teams to these patterns, the schedule is still valid. The integer program in this step is significantly more complicated, including 234 constraints and over 300 variables. The 17 feasible pattern sets in step 2 result in 826 feasible timetables in step 3.

4) Assigning teams to patterns

Once the schedule is created, the final computational step is to assign the teams to the given patterns. In this step, individualized constraints must be taken into account. These include stadium scheduling conflicts, school finals schedules, rivalry week requirements, etc.

5) Manually inspecting the overall schedule

Inevitably there are constraints that are very hard to quantify and thus require vast knowledge of the league and teams involved in order to determine if a potential schedule will work as the league intends. In the case of this paper, they supplied 3 potential schedules that met all of the constraints, yet were declined due to these intricacies.

5. Open Problems & Possible Extensions

Given the exponential growth of the number of possible schedules as the number of teams, games, and constraints increase, the computational time to enumerate these permutations also grows exponentially. Enhancements of any of the subproblems listed in Nemhauser & Trick's paper would likely improve the efficiency with which a number of league schedules could be generated. However, as soon as yesterday's infeasible problem becomes feasible, consideration of additional scheduling constraints as requested by referees, television, etc, can again complicate the algorithms. As an algorithm's efficiency increases, it allows for "weighted" constraints as well as decision based-constraints (e.g., if team A plays team B on Wed., then their next match must be on a Saturday, otherwise they can play on Friday or Sunday). As we've seen with many of the papers listed, the minor details of the constraints have a major impact in the final result of the optimal schedule. Finding a way to determine which constraints can be slightly modified for algorithmic purposes could potentially lead to much more effective solutions.

Lastly, the field of sports scheduling is an interesting mix of theory and practicality, a field where non-trivial problems are NP-complete yet some solution needs to be found, optimal or not. Since the optimality of the schedule has a lot to do with the amount of money made by the various organizations associated with the leagues, a theoretical advancement with practical effect could be worth millions of dollars.

Bibliography

1. "Scheduling a Major College Basketball Conference", a paper by George L. Nemhauser and Michael A. Trick published in 1997, available online at http://mat.gsia.cmu.edu/trick/acc.pdf

2. "Constraint Programming for the Travelling Tournament Problem", a paper by Gan Tiaw Leong from the National University of Singapore in 2002/03, found online at http://www.comp.nus.edu.sg/~henz/students/gan_tiaw_leong.pdf

3. "Devising a Cost Effective Baseball Scheduling by Evolutionary Algorithms", a paper by Jih Yang, Hsien-Da Huang, and Jorng-Tzong Horng at National Central University in Taiwan, published in 2002 and available online at: http://ieeexplore.ieee.org/iel5/7875/21687/01004491.pdf?tp=&isnumber=&arnumber=1004491

4. "Designing Schedules for Leagues and Tournaments" a paper by Jeffrey H. Dinitz of the Math department of the University of Vermont, published in 2001 (approx), available online at: http://www.emba.uvm.edu/~dinitz/preprints/design_tourney_talk.pdf

5. Power point Sports scheduling tutorial by Michael Trick, prepared for Banff CORS/INFORMS conference of May 2004 but never presented, available online at: http://mat.tepper.cmu.edu/trick/banff.ppt