Interfaces
Vol. 42, No. 3, May–June 2012, pp. 232–244
ISSN 0092-2102 (print) | ISSN 1526-551X (online)

http://dx.doi.org/10.1287/inte.1100.0514
© 2012 INFORMS

# Scheduling Major League Baseball Umpires and the Traveling Umpire Problem

Michael A. Trick
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, trick@cmu.edu

Hakan Yildiz
Eli Broad College of Business, Michigan State University, East Lansing, Michigan 48824, yildiz@msu.edu

Tallys Yunes
School of Business Administration, University of Miami, Coral Gables, Florida 33124, tallys@miami.edu

The scheduling needs of umpires and referees differ from the needs of sports teams. In some sports leagues, such as Major League Baseball in the United States, umpires travel throughout the league's territory; they do not have a "home base." For such leagues, balancing the need to minimize umpire travel and the objective that an umpire should not handle the games of a particular team too frequently is important. We have used our approach, which is based on network optimization and simulated annealing, to successfully schedule Major League Baseball umpires. To develop this approach, we created the traveling umpire problem, which includes the major umpire scheduling issues and also provides a test bed for alternative techniques.

*Key words*: sports; baseball; umpire scheduling; integer programming; constraint programming; heuristics; greedy matching; simulated annealing.
*History*: This paper was refereed. Published online in *Articles in Advance* June 1, 2011.

Major League Baseball (MLB) comprises 30 teams that play 2,430 games in 780 series during a six-month season each year; a series is defined as a sequence of two to four games played consecutively between two opponents. An umpire (referred to as a referee in other sports) officiates at each game and is responsible for the smooth running of the game, including any necessary rule interpretations that arise. Each umpire is part of an umpire crew, which consists of a group of four umpires; an umpire crew stays together as a team throughout the season. During the season, the umpire's job is full time, and the typical umpire handles approximately 142 games (a player plays 162 games during this period). Unlike players, who have a home city in which they play half of their games, umpires travel from city to city throughout the season, because one of MLB's objectives is to ensure that an umpire does not handle the games of any team too frequently in one season; assigning an umpire to a home base would conflict with this goal. A secondary reason is that baseball games are typically held in a particular city only half the time. MLB would have to almost double its number of umpire crews if it were to assign one crew to each city in which it plays (the 30 teams play in 27 different cities).

Because umpires travel throughout the season, minimizing their travel is important. However, the requirement that a crew not handle the games of any team too often forces each crew to travel after each series of games. Crews must travel long distances (i.e., up to 35,000 miles) during a season because of (1) the requirement that each crew must visit each MLB home city and (2) restrictions on a crew handling consecutive series for any team. In addition, the associated team schedules are not designed with umpire travel as a consideration. In Figure 1, we illustrate a "typical" umpire's extensive travel during one season. It consists of three paths, because the umpire's travel must be broken up by vacation weeks (we do not show three shorter trips around the all-star break and at the end of the season). In the top-left path in this example, the umpire starts in Oakland, California and travels to the following locations: Los Angeles, California; Phoenix, Arizona;

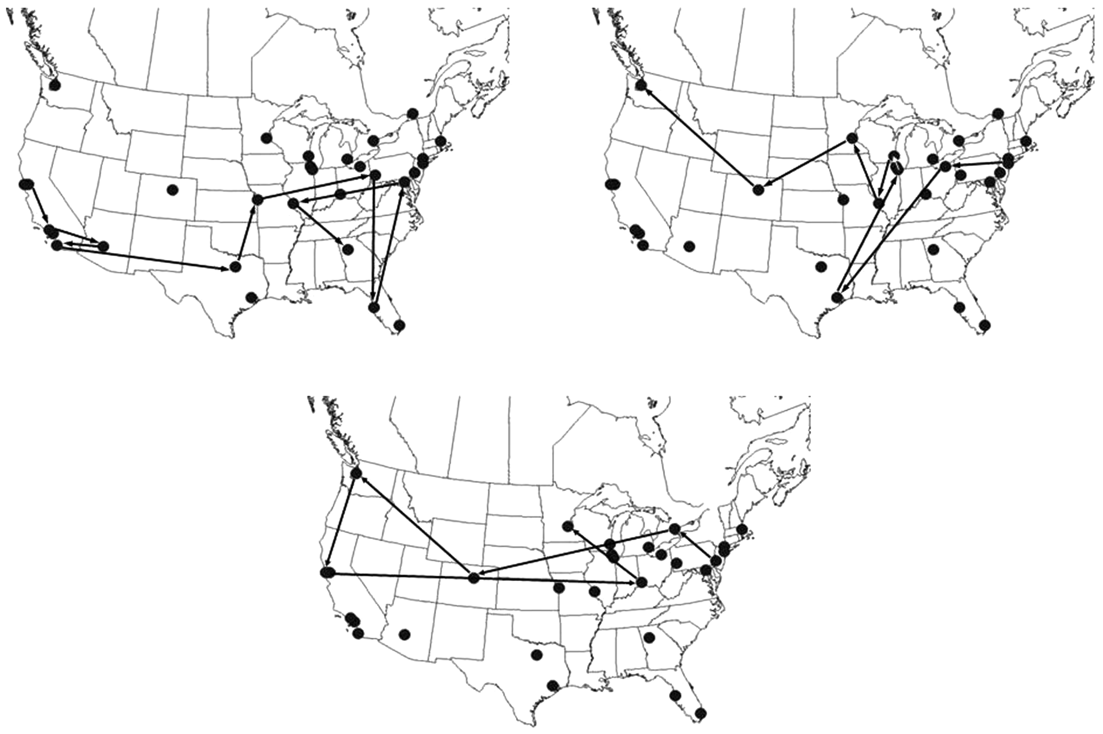

**Figure 1: The graphic shows three sample paths traveled by one crew in one year (MLB teams in the United States and Canada).**

San Diego, California; Dallas, Texas; Kansas City, Kansas; Pittsburgh, Pennsylvania; Tampa, Florida; Baltimore, Maryland; St. Louis, Missouri; and finally to Atlanta, Georgia. This 5,868-mile trip spans four time zones; the umpire would travel it over 38 days and would handle 35 games during this period.

The umpire scheduling problem is complex and difficult to solve because it consists of dozens of pages of constraints, including idiosyncratic constraints such as an umpire's preferred vacation dates. The schedule, which must satisfy league-imposed travel rules and restrictions, aims to optimize many conflicting goals. Prior to using our scheduling solution, one specialist, a former umpire, manually built MLB's umpire schedules using Microsoft Excel; this daunting task took weeks of planning. As an alternative to this approach, we worked with MLB to develop a heuristic; MLB accepted the schedule generated using this method in its 2006 season and used modifications of the approach during its 2008, 2009, and 2010 seasons.

To develop this heuristic, we needed a test bed for our experimentation. Because only one schedule is generated per season, we have only one instance to work with each year. This was insufficient to allow us to determine appropriate, robust methods. For example, we might design a wonderful approach to the 2006 instance only to have it do poorly using the 2007 data. We therefore developed the traveling umpire problem (TUP). The TUP is similar to the traveling tournament problem (TTP) for league scheduling, which Easton et al. (2001) introduced, and is based on the most important features of MLB umpire scheduling. The TTP (http://mat.tepper.cmu.edu/TOURN) has encouraged research on team scheduling approaches; however, it does not get mired in idiosyncratic league details. Using the TUP allowed us to test alternative approaches to umpire scheduling.

The literature contains many papers that address the scheduling of sports leagues; examples include Easton et al. (2004), Rasmussen and Trick (2008), and

Briskorn (2008). However, few papers address the scheduling of sports umpires. Duarte et al. (2007) consider a general referee assignment problem. Farmer et al. (2007) consider scheduling umpires in the US Open; two papers (Wright 1991, 2004) deal with scheduling umpires for cricket leagues. However, the issues these papers discuss are somewhat different than those that we address. In most of these papers, the primary issue involves finding qualified referees to handle games; they put much less emphasis on intergame travel and balancing travel to different stadiums.

Some studies specifically relate to MLB umpire scheduling. Evans (1988), Evans et al. (1984), and Ordonez (1997) discuss scheduling issues and approaches during the 1980s and 1990s; their work provides the basis for much of the material that we discuss in this paper. However, MLB has made many changes since that period. The most critical change occurred in 2000, when two MLB leagues combined their umpires, creating a common umpire pool. This change roughly doubled the problem size and made requirements on seeing every team much more difficult to meet.

## Problem Description

In this section, we formally define the MLB umpire scheduling problem (MLB-USP) and the TUP, which extracts the most critical aspects of the MLB-USP, but not its extraneous features.

### The MLB Umpire Scheduling Problem

In MLB, each umpire crew comprises four trained and licensed major league umpires. These crews are assembled at the beginning of each season and work as a crew all season long, with very limited exceptions. Umpire union rules require that each umpire receives four week-long vacations during the baseball season, three as a crew and one individually. Thus, MLB requires 17 umpire crews; each week, 2 crews are on vacation and 15 crews are available for scheduling to series. The umpires are assigned to a previously developed schedule of games to be played by the teams. The umpire scheduler, whose main goal is to minimize the miles that each crew travels, must adhere to many rules; some are required because of union rules or physical limitations; others attempt to accommodate umpire preferences.

In principle, the objective function of an optimization model for the MLB-USP is to minimize the total number of miles that the 17 umpire crews travel. However, when all the constraints are combined, the problem becomes infeasible. Hence, we break the constraint set into two types, *hard constraints* and *soft constraints*; we summarize each below.

*Hard Constraints.* These constraints must always be satisfied for a schedule to be considered acceptable. Crews *must not*

- travel from the West Coast to the East Coast without an intermediate day off;
- umpire consecutive series more than 1,700 miles apart without an intermediate day off;
- travel more than 300 miles preceding a series whose first game is a day game (i.e., before 4 PM).

Crews *must*

- take three one-week prescheduled vacations (a crew cannot take its second vacation before all other crews have had their first vacation); the fourth week of vacation is taken individually and not as part of the scheduling process.

*Soft Constraints.* These constraints may be violated but at a penalty. Crews *should not*

- work more than 21 days without a day off (the "21-day rule");
- umpire more than one series played by any team within any 18-day period (the "18-day rule");
- umpire more than four series played by any team during the entire season (the "four-series rule").

Crews *should*

- travel to all cities at least once;
- see each team at home and on the road;
- have balanced schedules; they should travel a similar number of miles, umpire approximately the same number of games, and have the same number of days off.

In practice, we handle the soft constraints by penalizing their violation in the objective function. For example, for each umpire crew, we penalize the following measures:

- the number of times the umpire crew works more than 21 days without a day off; we penalize similarly for violations of the 18-day rule and the four-series rule;

• the difference between the total distance that an umpire crew travels and the average distance that all umpire crews travel; we penalize similarly for the number of games umpired and number of days off;

• the number of teams that an umpire crew does not see at home, away, or at all.

The MLB's current scheduling method incorporates "split" series (i.e., a series in which two crews cover a single series). For example, one crew works the first three games of a four-game series and then leaves for a new assignment; a new crew comes to finish the last game of the series. As another example, a crew may stay on for the first game of the next series in the stadium in which it is working and then move on to another assignment. In our approach, we avoid the use of split series whenever possible.

The concept of a *slot* is straightforward in the context of the TUP (see *The Traveling Umpire Problem* subsection); however, its use in the MLB-USP requires further explanation. Each series comprises two, three, or four games between the same teams in the same venue. Series are either weekday series played between Monday and Thursday or weekend series played between Friday and Sunday. Therefore, we divide each week into two slots: one for the weekdays and the other for the weekend. One Monday–Thursday slot is then subdivided into two slots because it consists of two sets of two-game series (Monday–Tuesday and Wednesday–Thursday), with an additional slot for the all-star break. Finally, we divide the season into 53 slots of 15 simultaneous series each. Some series cross over the weekday/weekend slot divisions. For example, a series might start on Monday and end on Thursday or start on Friday and end on Monday. We flag these series as crossovers of type one or type two, respectively; we then assign them to the slot that holds the majority of the games, thus simplifying the coding of our optimization algorithm. For each slot, we give each crew one assignment; we also assign 15 crews to one of the 15 series in the slot, and two crews to vacation.

This description does not exhaust the set of rules, requirements, and requests that comprises the MLB-USP. However, it is clearly sufficient to create an extremely difficult scheduling problem.

## The Traveling Umpire Problem

To explore computational approaches to the MLB-USP, we need a source of instances that mimic the problem without getting mired in the details. In this section, we introduce the TUP.

In contrast to the MLB-USP, the TUP limits the constraints to the key issues: an umpire crew should not be assigned to a team too often in short succession, and each umpire crew should be assigned to each team at some time during a season. Given these constraints, the objective is to minimize the travel of the umpire crews.

Given a double round robin tournament, in which each team plays against each other team twice, on $2n$ teams ($4n-2$ slots), we want to assign one of $n$ umpire crews to each game. Note that we do not have any extra crews, as we have in the MLB-USP.

The following constraints must be satisfied.

(1) Each game has an umpire crew;

(2) each umpire crew works exactly one game per slot;

(3) each umpire crew sees each team at least once at the team's home;

(4) no umpire crew is in a home site more than once in any $n-d_1$ consecutive slots;

(5) no umpire crew sees a team more than once in any $\lfloor n/2 \rfloor - d_2$ consecutive slots.

We next describe the properties of the parameters in the constraints; note that we do not arbitrarily select the parameters for Constraints (4) and (5).

Let $P$ represent the TUP and let $P_{(R)}$ be a relaxation of $P$ with constraint set $R$, where $R \subset \{1, 2, 3, 4, 5\}$.

THEOREM 1. *For $P_{(1,2,4)}$, for any tournament and any game, there exists a feasible schedule that covers that game for a single crew. Moreover, Constraint* (4) *has the following properties*: (1) *when $n$ is even, $n$ is an upper bound on $n-d_1$ for the existence of this feasibility, and* (2) *when $n$ is odd, $n+1$ is an upper bound on $n-d_1$ for the existence of this feasibility.*

THEOREM 2. *For $P_{(1,2,5)}$ and $d_2 = 0$ for any tournament and any game, there exists a feasible schedule that covers that game for a single crew.*

The proofs of both theorems are available in Yildiz (2008). Theorem 1 implies that although we do not want a crew to see the same team at home frequently,

| Slots | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| UmpireCrew1 | (1, 3) | (3, 4) | (1, 4) | (3, 1) | (4, 3) | (2, 3) |
| UmpireCrew2 | (2, 4) | (1, 2) | (3, 2) | (4, 2) | (2, 1) | (4, 1) |

**Table 1: This table illustrates a round robin tournament for four teams and a corresponding feasible schedule for two crews.**

a limit exists on the enforcement of this constraint—even for the relaxation $P_{(1,2,4)}$ of the TUP. This limit is at most $n$ consecutive games, in which case $d_1 = 0$. Theorem 2, on the other hand, does not imply any restrictions on the number of consecutive slots during which an umpire crew can see the same team more than once. It only shows that the relaxation $P_{(1,2,5)}$ of the TUP is always feasible when $d_2 = 0$. In this case, $d_1$ and $d_2$ are parameters that represent the level of constraint required. Setting each to 0 leads to the most constrained system; setting each to $n$ and $\lfloor n/2 \rfloor$, respectively, "turns off" the corresponding constraint.

We present an example of a round robin tournament for four teams and a feasible umpire schedule for $d_1 = d_2 = 0$ in Table 1. We represent a game as a pair $(i, j)$ where $i$ is the home team and $j$ is the away team. Rows correspond to crew schedules; columns correspond to games that are played in the corresponding time slots.

Using the TUP definition and the schedules from the TTP, we can generate instances with far fewer teams than MLB, allowing us to experiment with different approaches.

## Exact Solution Approaches

The MLB-USP and TUP have many characteristics in common with the vehicle routing problem with time windows (VRPTW), which also emphasizes minimizing the total travel cost of multiple routes. If we ignore TUP Constraints (3), (4), and (5), it becomes a special case of the VRPTW. VRP and almost all its variants, including VRPTW, are NP-hard (Lenstra and Kan 1981), and exact solution approaches are ineffective in solving large instances. Because both the MLB-USP and TUP have side constraints in addition to the routing constraints, solving them is more challenging than solving the VRP and its variants.

We tried various integer programming (IP) formulations of the MLB-USP (see Appendix A); however,

| Crews | Slots | Variables | Constraints | Nodes | Optimality gap (%) |
|---|---|---|---|---|---|
| 17 | 3 | 5,878 | 10,028 | 143,216 | 0.23 |
| 17 | 4 | 9,607 | 14,646 | 46,495 | 0.29 |
| 17 | 5 | 12,412 | 22,230 | 23,454 | 1.11 |
| 17 | 6 | 15,513 | 26,294 | 4,919 | ∞ |
| 17 | 7 | 18,575 | 34,945 | 2,228 | 3.83 |
| 17 | 8 | 21,956 | 39,187 | 1,301 | 10.89 |
| 17 | 9 | 24,525 | 46,734 | 350 | 23.34 |
| 17 | 10 | 27,353 | 50,466 | 131 | ∞ |

**Table 2: This table shows IP results for the MLB-USP. Instances ran for 20 CPU hours each.**

we found all these formulations to be impractical for even a few slots, let alone the MLB's full 53 slots. In Table 2, we report the number of variables and constraints after CPLEX's preprocessing step, the number of search nodes explored in the branch-and-bound tree, and the final optimality gap. An optimality gap equal to "∞" means that no feasible solution was found within 20 hours.

Our experience with the MLB-USP carries over to the TUP, suggesting that the TUP includes constraints that cause the difficulty in solving the MLB-USP. We formulated the TUP as an IP (see Appendix B) and as a constraint program (CP) (see Appendix C). Table 3 shows the computational results for the IP and CP models for seven instances with $d_1 = d_2 = 0$, which means Constraints (4) and (5) are the most restrictive; the *Heuristic Approach* section gives more detail on how we generated these instances. We allowed a maximum of 24 CPU hours for each approach. Although both approaches were able to solve the 4-, 6-, and 8-team instances to optimality very quickly, only the IP approach could solve the 10-team instance

| No. of teams | OPT distance | BEST distance |  | Time to prove OPT or find BEST |  |
|---|---|---|---|---|---|
|  |  | IP | CP | IP | CP |
| 4 | 5,176 | 5,176 | 5,176 | 0 | 0 |
| 6 | 14,077 | 14,077 | 14,077 | 0 | 0 |
| 8 | 34,311 | 34,311 | 34,311 | 2 secs | 0 |
| 10 | 48,942 | 48,942 | 49,400 | 72 secs | 24 hrs |
| 12 | Infeasible | — | Infeasible | 24 hrs | 2 mins |
| 14 | Unknown | 187,374 | 176,903 | 24 hrs | 24 hrs |
| 16 | Unknown | — | — | 24 hrs | 24 hrs |

**Table 3: This table shows IP and CP results for TUP with $d_1 = d_2 = 0$.**

to optimality within 72 seconds; the CP approach was unable to solve that instance to optimality within 24 hours. In only 25 seconds, the CP model showed the 12-team instance to be infeasible, whereas the IP model was unable to prove infeasibility in the allowed 24 hours. For the 14-team instance, no approach was able to prove optimality. However, the CP model found a much better solution for this instance much faster. Finally, no model was able to find a feasible solution for the 16-team instance.

Based on these results, finding optimal solutions to either the MLB-USP or the TUP instances with 30 teams is clearly not practical; this forces us to rely on a heuristic approach.

## Heuristic Approach

Given the difficulties we face when trying to find optimal USP solutions, we explore heuristic approaches to find good solutions in a reasonable amount of time. We begin with a simple greedy heuristic to generate an initial solution, which we then improve by using local search in a simulated annealing framework.

### Greedy Matching Heuristic

The greedy matching heuristic is a constructive heuristic that allows us to build the umpire schedules starting from the first slot and ending at the last slot. This approach is similar to that of Evans (1988) and Evans et al. (1984), who scheduled the American League umpires for several years. (The American League includes approximately half the MLB teams.)

For every slot $t$, the heuristic assigns umpire crews to series. The best possible assignment minimizes both the total umpire travel in slot $t$ and the constraint violations. To do that, the heuristic solves a perfect matching problem on a bipartite graph in each slot $t$. This bipartite graph shows the *umpire crews* on one side of the partition and the *series* of slot $t$ on the other side. Let $E(t)$ represent the edges between the two sides. Moreover, let $u$ be an umpire crew and $(i, j)$ indicate a series played by teams $i$ (at home) and $j$ (away). The cost of an edge $(u, (i, j)) = \text{distance}(k, i) - \text{incentive}(u, i) + \text{penalty} * \text{violations}(u, i, t)$. In this cost function, $k$ is the location of crew $u$ in slot $t-1$; distance$(k, i)$ is the distance between city $k$ and team $i$'s home city; incentive$(u, i)$ takes a positive value if crew $u$ has never visited team $i$'s home in the previous slots; violations$(u, i, t)$ is the number of constraint violations caused by assigning $u$ to $i$ at $t$, and *penalty* is a large cost associated with a single constraint violation. This cost structure guides the greedy matching heuristic toward assigning umpire crews to cities that they have not visited yet, while violating the fewest number of constraints. In slot $t$, if $\sum_{(u,(i,j))\in M(t)} \text{violations}(u, i, t) > 0$, where $M(t)$ is the set of edges in the best matching solution at $t$, no feasible matching is available at $t$. When this happens, the greedy matching heuristic backtracks to the previous slot $t-1$, picks the second-best matching, and tries again. Backtracking is made at most once at each slot. Thus, by using the greedy matching heuristic, we might end up with an infeasible solution. When that happens, we correct the infeasibility by using our improvement heuristic, which we describe in the *Local Search* and *Simulated Annealing* subsections.

### Local Search

A *neighborhood* of a solution $S$ is a set of solutions that are close to $S$ (i.e., they can be easily computed from $S$ or they share a significant amount of structure with $S$). An algorithm that starts at some initial solution and iteratively moves to solutions in the neighborhood of the current solution is called a *neighborhood search algorithm* or a *local search algorithm*.

The local search for the umpire scheduling problems discussed in this paper tries to improve the solution quality at each iteration as follows. Given an umpire schedule $S$, we use a *two-exchange move* to swap the umpire crews assigned to two series played in the same slot. The neighborhood of $S$ according to this move is the set of all schedules that can be obtained from $S$ by performing a single two-exchange move.

### Simulated Annealing

The major disadvantage of a pure local search algorithm is that it terminates in the first local optimum it reaches, which may be far from any global optimum, because the algorithm only executes moves that generate a decrease in cost. Simulated annealing, a powerful stochastic local search method, alternatively attempts to avoid becoming trapped in a local optimum by sometimes (with a nonzero probability that gradually decreases as the algorithm continues its execution) executing a move that generates an increase in cost; this can enable it to "climb out of" the local minimums.

Simulated annealing has its origins in the fields of materials science and physics (Pinedo and Chao 1999). Kirkpatrick et al. (1983) established an analogy between minimizing the cost function of a combinatorial optimization problem and the slow-cooling process of a solid by using an optimization process. This algorithm has proven to be a good technique for many applications (Vidal 1993).

Algorithm 1 shows the pseudocode for the simulated annealing algorithm that we used.

For the TUP, our cost function is the total distance that the crews travel. However, our MLB-USP cost function is more complicated. It consists of the total mileage traveled by the 17 umpire crews plus penalties related to the violation of the soft constraints described in the *MLB Umpire Scheduling Problem* section. All violations have their specific penalty weights (coefficients in the cost function), which we adjust empirically. For example, each traveled mile might have a weight of 1, and each violation of the 21-day rule might have a weight of 1,000; this would (approximately) mean that we would be willing to increase the total mileage by 1,000 in exchange for one fewer violation of the 21-day rule.

We empirically tested the parameters for the simulated annealing algorithm; in our tests, we used the values shown in Table 4.

**Algorithm 1** (Simulated annealing).

```
1: while time limit and iteration limit not exceeded do
2:    $S$ = initial solution with prob. $p$ or incumbent solution with prob. $(1-p)$
3:    $t = t_0$
4:    while $t >$ TEMP_LIMIT do
5:       for all ITER iterations do
6:          Pick one feasible exchange $E$ at random
7:          $d$ = impact of $E$ in objective function
8:          if $d < 0$ then
9:             Execute $E$
10:            if new solution better than incumbent then
11:               Update incumbent
12:            end if
13:         else
14:            $x$ = random number in $[0, 1]$
15:            if $x < \exp(-d/t)$ then
16:               Execute $E$
17:            end if
18:         end if
19:      end for
20:      $t = t * \text{ALPHA}$
21:   end while
22: end while
```

| Parameter | Value |
|---|---|
| $t_0$ | 2,000 |
| TEMP_LIMIT | 500 |
| ITER | 2, 500 |
| ALPHA | 0.95 |
| $p$ (for MLB-USP) | 0.1 |
| $p$ (for TUP) | 0.2 |

**Table 4: This table contains the parameters for the simulated annealing algorithm.**

## Computational Results

In this section, we report the computational results of testing our solution approach on the 2006 MLB schedule and on a set of TUP instances.

### TUP Instance Description

An instance of the TUP has two matrices: the distance matrix, which stores the pairwise distances between cities, and the opponents matrix, which stores the tournament information. We used instances with a number of teams ranging from 4 to 16. The instances with 14 teams or fewer use the TTP Tournaments as Trick (2009) discusses. The 16-team instance uses the distance matrix for the National Football League (Trick 2009); the game schedule is generated using a constraint program (Trick 2003) that creates a round robin tournament. The instances we used in this study (and additional instances) are available at http://mat.tepper.cmu.edu/TUP.

Depending on the choice of $d_1$ and $d_2$, the difficulty of the problem changes. Assigning a positive value to either parameter creates a relaxation of the original problem; therefore, as we increase the values of these two parameters, the problem becomes easier to solve. For example, choosing $d_1 = n-1$, which makes $n - d_1 = 1$, or $d_2 = \lfloor n/2 \rfloor - 1$, which makes $\lfloor n/2 \rfloor$ - $d_2 = 1$,

| | 2005 | 2006a | 2006b | 2008 | 2009 | 2010 |
|---|---|---|---|---|---|---|
| Total mileage | 430,795 | 465,175 | 463,452 | 445,932 | 458,258 | 455,642 |
| Max – min mileage | 9,078 | 4,540 | 4,426 | 3,842 | 6,868 | 4,257 |
| 21-day rule violations | 2 (23, 27) | 0 | 0 | 3 | 2 | 0 |
| 18-day rule violations | 16 | 1 | 0 | 0 | 0 | 0 |
| Max – min no. of games | 6 | 3 | 4 | 3 | 1 | 1 |
| Missed at home | 26 | 18 | 11 | 28 | 9 | 14 |
| Missed on the road | 79 | 76 | 59 | 69 | 44 | 70 |
| Missed completely | 0 | 0 | 3 (1) | 0 | 1 | 0 |

**Table 5: This table contains the results for the MLB-USP obtained with simulated annealing.**

simply means that Constraint (4) or Constraint (5), respectively, is not in effect.

## Summary of Results

We used the methodology described in this paper to provide MLB schedules during the 2006 and 2008–2010 seasons. In this section, we describe the results for the 2006 schedule.

We coded the greedy matching heuristic using Visual Basic within Microsoft Excel, and the simulated annealing heuristic using C. Both algorithms are run on a Linux PC with Pentium 4 3.7 GHz processor. We obtained the best solutions using a sequence of runs with different penalty weights; the longest single run took approximately four days (approximately two billion iterations). Table 5 summarizes the results.

Column "2005" shows the characteristics of the 2005 MLB umpire schedule, which we constructed manually. Columns "2006a" and "2006b" show two of the best schedules we were able to obtain for the 2006 season. In summary, the schedules we obtained improve almost every measure of quality in exchange for higher total mileage (approximately 2,000 additional miles per crew over the entire season). In keeping with MLB's standard reporting, the mileage shown is for travel through the end of August (approximately 85 percent of the schedule), while the other statistics represent the full season. Schedules 2006a and 2006b show no violations of the 21-day rule, and schedule 2006a has only one violation of the 18-day rule. In 2005, two crews worked for 23 and 27 days without a day off, and, in 16 instances, a crew saw the same team more than once within an 18-day period. Our schedules are also more balanced, both in terms of the number of miles traveled and number of games umpired by the crews. Finally, we also reduced the number of times that the umpire crews fail to see the different teams at home and on the road. The entry in the last row of schedule 2006b means that there are three umpire crews that fail to see one of the 30 teams.

MLB used our scheduling method for its 2006 season; it used another method for its 2007 season, and returned to using our scheduling method in 2008. We also provided the schedules MLB used for its 2009 and 2010 seasons. Our methods consistently provide high-quality schedules, even as the underlying team schedule changes and the scheduling requirements vary. Note that, over time, the MLB trade-offs have changed; for example, during the 2009 and 2010 seasons, it strongly emphasized equal game counts among the crews, and ensuring that each crew sees as many teams at home as possible. Our methods generate schedules that meet each year's unique objectives and requirements, thus providing MLB with flexibility.

In summary, MLB benefited from this study; its umpire schedules are more balanced and have fewer rule violations, and its umpires miss fewer at-home and on-the-road teams. In addition, generating the schedules requires less time and manual effort.

To solve the TUP instances, we implemented the greedy matching heuristic and the simulated annealing algorithm using the script language in ILOG OPL Studio 3.7. We ran the algorithms on a Linux server with an Intel(R) Xeon(TM) 3.2 GHz processor. However, as the problem size increased, even finding a feasible solution to the TUP became difficult. Table 6 summarizes the results using the heuristic approach on the smallest seven instances with $d_1 = d_2 = 0$, which means that Constraints (4) and (5) are the most restricting. Although the heuristic was able to solve

| No. of teams | OPT distance | Distance | Time (secs) |
|---|---|---|---|
| 4 | 5,176 | 5,176 | 0 |
| 6 | 14,077 | 14,077 | 0 |
| 8 | 34,311 | 34,311 | 60 |
| 10 | 48,942 | 50,196 | 228 |
| 12 | Infeasible | — | — |
| 14 | Unknown | — | — |
| 16 | Unknown | — | — |

**Table 6: This table shows the simulated annealing results for the TUP with $d_1 = d_2 = 0$.**

the 4-, 6-, and 8-team instances to optimality fast, it was unable to solve the 10-team instance to optimality. However, it was able to find a quality solution fast. As we stated above, the 12-team instance proved to be infeasible, and we were unable to find a feasible solution for the 14- and 16-team instances using the heuristic approach.

Based on the results we obtained using the three different techniques, it is obvious that the 14- and 16-team instances are very difficult instances to solve when $d_1 = d_2 = 0$. To further investigate the heuristic approach's performance on the TUP, we solved the relaxations of these two instances by increasing the values of $d_1$ and $d_2$ (see Table 7). We also solved these instances using the IP formulation. We ran the IPs for 24 hours, whereas we ran the heuristic for three hours. For the 14-team instances, we see that the simulated annealing approach obtained as good or better results than the IP approach in a shorter time. For the 16-team instance, neither method was able to find a feasible solution, except for the relaxation with $n - d_1 = 7$ and $\lfloor n/2 \rfloor - d_2 = 2$. For that instance, we were able to find a solution using the simulated annealing method.

| No. of teams | $n - d_1$ | $\lfloor n/2 \rfloor - d_2$ | Integer program: Distance | Integer program: Time (hrs) | Simulated annealing: Distance | Simulated annealing: Time (hrs) |
|---|---|---|---|---|---|---|
| 14 | 6 | 3 | 182,531 | 24 | 180,697 | 3 |
| 14 | 5 | 3 | 169,012 | 24 | 169,173 | 3 |
| 16 | 8 | 2 | — | 24 | — | 3 |
| 16 | 7 | 3 | — | 24 | — | 3 |
| 16 | 7 | 2 | — | 24 | 176,527 | 3 |

**Table 7: This table shows IP and simulated annealing results for TUP on the relaxations of 14- and 16-team instances, with $d_1 + d_2 > 0$.**

| No. of teams | $n - d_1$ | $\lfloor n/2 \rfloor - d_2$ | IP: Dist. | IP: Time (hrs) | CP: Dist. | CP: Time | SA: Dist. | SA: Time |
|---|---|---|---|---|---|---|---|---|
| 30 | 5 | 5 | — | out of memory | — | 24 hrs | 581,363 | 5 hrs |

**Table 8: This table shows IP, CP, and simulated annealing (SA) results for TUP on the MLB's 2006 game schedule with 30 teams.**

As we stated when we defined it, the TUP is an abstraction of the MLB-USP, which is defined on a 30-team league and game schedule. To reconcile the TUP and MLB-USP, we created an instance of the TUP on this set of teams and the 2006 game schedule. To mimic the "18-day rule," we set $\lfloor n/2 \rfloor - d_2 = 5$ and tried to solve this instance using the IP, CP, and heuristic approaches (see Table 8). We see that the heuristic approach outperformed both the IP and CP approaches on this instance.

## Conclusion

In this paper, we present the method we have used to schedule the MLB umpires in 2006 and 2008–2010. We formally define the MLB-USP, introduce a new approach to developing umpire schedules, and define the TUP.

This project started in the spring of 2005 as an elective course for MBA students in the operations research track of Tepper School of Business. The research team held periodic meetings with a former MLB umpire who had been responsible for constructing the umpire schedules, which he manually built in a few weeks. Based on what they learned in these meetings, the research team members were able to understand what an acceptable schedule should look like. At the end of the spring semester, the team produced a few schedules; however, they were unsuitable for actual implementation because they underemphasized the prohibition on repeating visits in too short of a time. Early in 2006, we developed the approach described in this paper; we generated the actual schedules for the 2006 season in February 2006.

We show that umpire scheduling, although simpler than game scheduling, is a challenging problem. We also show that conventional optimization methods are ineffective in solving the MLB-USP and large instances of the TUP, and even find it difficult to find feasible solutions. This experience led us to believe that metaheuristics provide better solutions than exact solution methods.

We demonstrate that the heuristic approach based on simulated annealing is simple to implement and provides good results. However, our method needs fine-tuning. One improvement to our MLB software could be the use of IP large neighborhood search. For example, we could unassign a piece of the schedule, reassign it in an "optimal" way, and repeat the process for other pieces.

The algorithm we describe in this paper is specific to MLB umpire scheduling, particularly because of the objective function and the slot structure. The high-level technique (greedy matching heuristic followed by simulated annealing) could also be reused to solve other umpire scheduling problems.

## Appendix A. IP Formulation for the MLB-USP

### A.1. Problem Data

*Constants That Appear in the Constraints*

- $S$, $T$, $C$ = sets of all series, teams, and crews, respectively;
- $H_k$, $R_k$ = sets of all series in which team $k$ plays at home or on the road, respectively;
- $V$ = set of pairs of series $(i, j)$ that form a valid transition (i.e., they are in consecutive slots and do not violate any *hard* travel restrictions);
- $M$ = set of pairs of series $(i, j)$ that, if assigned to the same crew, will cause a violation of the 18-day rule;
- $U$ = set of pairs of series $(i, j)$ that can never be assigned to the same crew for a reason (e.g., they constitute an unacceptable transition; they violate the 18-day rule by too much, etc.);
- $g_i$ = number of games in series $i$.

*Objective Function Coefficients*

- $d_{ij}$ = travel miles for all $(i, j) \in V$;
- $m_{ij}$ = penalty for assigning series $i$ and $j$ to the same crew for all $(i, j) \in M$ (the smaller the separation, the larger the $m$ value);
- $p_h$, $p_r$, $p_a$ = penalties for not seeing a team at home, on the road, and at all, respectively;
- $p_f$ = penalty for seeing a team more than four times;
- $p_g$, $p_m$ = penalties for deviating too much in number of games and mileage, respectively;
- $p_o$ = penalty for not having a day off in a 22-day period.

### A.2. Variables

- $x_{ic} = 1$ if series $i \in S$ is assigned to crew $c \in C$ (binary);
- $x_{ijc} = 1$ if crew $c$ umpires series $j$ immediately after series $i$, for all $(i, j) \in V$ (binary);
- $y_{ij} = 1$ if both series $i$ and $j$ are assigned to the same crew, for all $(i, j) \in M$ (binary);
- $h_{ck} = 1$ if crew $c$ does not see team $k \in T$ at home (continuous, $\geq 0$);
- $r_{ck} = 1$ if crew $c$ does not see team $k$ on the road (continuous, $\geq 0$);
- $a_{ck} = 1$ if crew $c$ does not see team $k$ at all (continuous, $\geq 0$);
- $f_{ck}$ = number of times crew $c$ sees team $k$ in excess of MEET_TOL times (continuous, $\geq 0$);
- $\delta_g$, $\delta_m$ = difference between maximum and minimum number of games and mileage, respectively, over all crews (continuous, $\geq 0$). These values are positive only if they are larger than GAMEDEV_TOL and MILEAGEDEV_TOL, respectively;
- $o_{wc} = 1$ if crew $c$ does not have a day off during the 22-day window $w$ (continuous, $\geq 0$).

### A.3. The IP Model

$$\begin{aligned}\text{Minimize}\quad & \sum_{(i,j)\in V}\sum_{c\in C} d_{ij}x_{ijc} + \sum_{(i,j)\in M} m_{ij}y_{ij} + p_h\sum_{k\in T}\sum_{c\in C} h_{ck}\\ &+ p_r\sum_{k\in T}\sum_{c\in C} r_{ck} + p_a\sum_{k\in T}\sum_{c\in C} a_{ck} + p_f\sum_{k\in T}\sum_{c\in C} f_{ck}\\ &+ p_g\delta_g + p_m\delta_m + p_o \sum_{\substack{\text{all 22-day}\\ \text{windows } w}}\sum_{c\in C} o_{wc}\end{aligned}$$

subject to

$$\sum_{c\in C} x_{ic} = 1, \quad \forall\, i \in S \tag{A1}$$

$$\sum_{i\in l} x_{ic} \leq 1, \quad \forall \text{slot } l, c \in C \tag{A2}$$

$$x_{ic} + x_{jc} - x_{ijc} \leq 1, \quad x_{ijc} \leq x_{ic}, \quad x_{ijc} \leq x_{jc}, \quad \forall\,(i, j) \in V,\ c \in C \tag{A3}$$

$$x_{ic} + x_{jc} - y_{ij} \leq 1, \quad \forall\,(i, j) \in M,\ c \in C \tag{A4}$$

$$x_{ic} + x_{jc} \leq 1, \quad \forall\,(i, j) \in U,\ c \in C \tag{A5}$$

$$1 - \sum_{i\in H_k} x_{ic} \leq h_{ck}, \quad \forall\, k \in T,\ c \in C \tag{A6}$$

$$1 - \sum_{i\in R_k} x_{ic} \leq r_{ck}, \quad \forall\, k \in T,\ c \in C \tag{A7}$$

$$h_{ck} + r_{ck} - a_{ck} \leq 1, \quad \forall\, k \in T,\ c \in C \tag{A8}$$

$$\sum_{i\in H_k\cup R_k} x_{ic} - \text{MEET\_TOL} \leq f_{ck}, \quad \forall\, k \in T,\ c \in C \tag{A9}$$

$$\sum_{i\text{'s off day} \in w} x_{ic} + \sum_{\substack{(i,j)\in V \\ (i,j)\text{'s off day} \in w}} x_{ijc} + o_{wc} \geq 1, \quad \forall \text{22-day window } w,\ c \in C \tag{A10}$$

$$\sum_{i\text{'s off day} \in w} x_{ic} + \sum_{\substack{(i,j)\in V \\ (i,j)\text{'s off day} \in w}} x_{ijc} \geq 1, \quad \forall (\text{MAX\_WORK}+1)\text{-day window } w,\ c \in C \tag{A11}$$

$$\sum_{i\in S} g_i x_{ic_1} - \sum_{i\in S} g_i x_{ic_2} - \text{GAMEDEV\_TOL} \leq \delta_g, \quad \forall \text{ordered pairs of crews } c_1, c_2 \tag{A12}$$

$$\sum_{(i,j)\in V} m_{ij} x_{ijc_1} - \sum_{(i,j)\in V} m_{ij} x_{ijc_2} - \text{MILEAGEDEV\_TOL} \leq \delta_m, \quad \forall \text{ordered pairs of crews } c_1, c_2 \tag{A13}$$

$$x_{ic} = 0, \text{ if } i \neq 0 \text{ is in a slot where } c \text{ is off} \tag{A14}$$

$$x_{ic} = 0, \text{ if } c \text{ is off in slot } l \text{ and } i \text{ is in slot } l-1 \text{ or } l+1 \text{ and crosses over.} \tag{A15}$$

Brief descriptions of the constraints follow. Constraint (A1): Each series must be assigned to a single crew; Constraint (A2): for each slot and crew, at most one series is played; Constraint (A3): if $x_{ic} = x_{jc} = 1$ for a given crew, then $x_{ijc} = 1$, and if either $x_{ic} = 0$ or $x_{jc} = 0$, then $x_{ijc} = 0$; Constraint (A4): if $x_{ic} = x_{jc} = 1$ for a given crew, then $y_{ij} = 1$; Constraint (A5): do not allow unacceptable constraint violations; Constraints (A6)–(A8): define variables that indicate whether or not a crew sees a given team at home, on the road, and at all, respectively; Constraint (A9): make $f_{ck}$ equal to the number of times above MEET_TOL (which was chosen to be equal to 4) that a crew sees a team during the season; Constraint (A10): define variables that indicate whether a crew has at least one day off inside a given 22-day window; Constraint (A11): prohibit a crew from working more than MAX_WORK+1 days without a day off (this is the hard limit on the 21-day rule); Constraints (A12)–(A13): define variables that measure the maximum difference in miles traveled and number of games played, respectively, across all crews. These differences only become meaningful once they are above the values of MILEAGEDEV_TOL and GAMEDEV_TOL, respectively; Constraint (A14): assign prescheduled vacations; Constraint (A15): prohibit crossover series around vacation slots.

We also studied the effect of adding cutting planes to the above formulation and decided to extend the model using the following cuts.

$$\sum_{c\in C} \sum_{j \mid (i,j)\in V} x_{ijc} = 1, \quad \forall i \in S,\ i \text{ is neither in the last slot nor possibly followed by a vacation;}$$

$$\sum_{c\in C} \sum_{i \mid (i,j)\in V} x_{ijc} = 1, \quad \forall j \in S,\ j \text{ is neither in first slot nor possibly preceded by a vacation.} \tag{A16}$$

## Appendix B. IP Formulation for the TUP

### B.1. Problem Data

- $S, T, C$ = sets of all slots, teams, and crews (umpires), respectively;
- $\text{OPP}[t,i] = \begin{cases} j, & \text{if team } i \text{ plays against team } j \text{ at venue } i \text{ in slot } t; \\ -j, & \text{if team } i \text{ plays against team } j \text{ at venue } j \text{ in slot } t; \end{cases}$
- $d_{ij}$ = travel miles between venues $i$ and $j$.

The following constants are defined to have a more readable model.

- $n_1 = n - d_1 - 1$;
- $n_2 = \lfloor n/2 \rfloor - d_2 - 1$;
- $N_1 = \{0, \ldots, n_1\}$;
- $N_2 = \{0, \ldots, n_2\}$.

### B.2. Variables

- $x_{isc} = 1$ if series, which is played at venue $i \in T$ in slot $s \in S$, is assigned to crew $c \in C$ (binary);
- $z_{ijsc} = 1$ if crew $c$ umpires series, which is played at venue $i$ in slot $t$, then umpires series played at venue $j$ in slot $t+1$ (binary).

### B.3. The IP Model

$$\text{Minimize} \quad \sum_{i\in T} \sum_{j\in T} \sum_{c\in C} \sum_{s\in S:\, s<|S|} d_{ij} z_{ijsc}$$

subject to

$$\sum_{c\in C} x_{isc} = 1, \quad \forall i \in T, s \in S\colon OPP[s,i] > 0 \tag{B1}$$

$$\sum_{i\in T:\, OPP[s,i]>0} x_{isc} = 1, \quad \forall s \in S,\ c \in C \tag{B2}$$

$$\sum_{s\in S:\, OPP[s,i]>0} x_{isc} \geq 1, \quad \forall i \in T,\ c \in C \tag{B3}$$

$$\sum_{s_1\in N_1} x_{i(s+s_1)c} \leq 1, \quad \forall i \in T, c \in C, s \in S\colon s \leq |S| - n_1 \tag{B4}$$

$$\sum_{s_2\in N_2} \left( x_{i(s+s_2)c} + \sum_{k\in T:\, OPP[s+s_2,k]=i} x_{k(s+s_2)c} \right) \leq 1, \quad \forall i \in T,\ c \in C,\ s \in S\colon s \leq |S| - n_2 \tag{B5}$$

$$x_{isc} + x_{j(s+1)c} - z_{ijsc} \leq 1, \quad \forall i,j \in T,\ c \in C,\ s \in S\colon s \leq |S|. \tag{B6}$$

We strengthened the formulation with the following additional valid inequalities.

$$x_{isc} = 0, \quad \forall i \in T,\ c \in C,\ s \in S\colon OPP[s,i] < 0 \tag{B7}$$

$$z_{ijsc} - x_{isc} \leq 0, \quad \forall i,j \in T,\ c \in C,\ s \in S\colon s < |S| \tag{B8}$$

$$z_{ijsc} - x_{j(s+1)c} \leq 0, \quad \forall i,j \in T, c \in C, s \in S\colon s < |S| \tag{B9}$$

$$\sum_{i\in T} z_{ijsc} - \sum_{i\in T} z_{ji(s+1)c} = 0, \quad \forall j \in T,\ c\in C,\ s\in S\colon s < |S|-1 \qquad \text{(B10)}$$

$$\sum_{i\in T}\sum_{j\in T} z_{ijsc} = 1, \quad \forall c\in C,\ s\in S\colon s<|S|. \qquad \text{(B11)}$$

Brief descriptions of the constraints follow. Constraint (B1): Each series must be assigned to a single crew; Constraint (B2): each crew is assigned to exactly one series per slot; Constraint (B3): each crew sees each team at least once at the team's home; Constraint (B4): no crew should visit a venue more than once in any $n-d_1$ consecutive slots; Constraint (B5): no crew should see a team twice in any $\lfloor n/2\rfloor - d_2$ consecutive slots; Constraint (B6): if crew $c$ is assigned to a series at venue $i$ in slot $s$ and to a series at venue $j$ in slot $s+1$, then the crew should move from $i$ to $j$ in slot $s$; Constraint (B7): if team $i$ plays away in slot $t$, no crew can be assigned to the series at venue $i$; Constraint (B8): if crew $c$ moves from venue $i$ to venue $j$ in slot $t$, it must be assigned to a series at venue $i$ in $t$; Constraint (B9): if crew $c$ moves from venue $i$ to venue $j$ in slot $t$, it must be assigned to a series at venue $i$ in $t+1$; Constraint (B10): number of crews moving to venue $j$ at slot $t$ should be equal to the number of crews moving from venue $j$ at $t+1$; Constraint (B11): each crew must move in each slot.

## Appendix C. CP Formulation for TUP in OPL

### Model Parameters

$T=\{1,\ldots,2n\}$ is the set of teams;
$S=\{1,\ldots,4n-2\}$ is the set of slots;
$U=\{1,\ldots,n\}$ is the set of umpire crews;

$$\text{opponents}[t,i] = \begin{cases} j, & \text{if team } i \text{ plays against team } j \text{ at venue } i \text{ in slot } t;\\ -j, & \text{if team } i \text{ plays against team } j \text{ at venue } j \text{ in slot } t;\end{cases}$$

dist$[i,j]$ = distance between venues $i$ and $j$.

### Decision Variables

team_assigned$[u,t,0]$ = the home team that umpire crew $u$ sees in slot $t$.
team_assigned$[u,t,1]$ = the away team that umpire crew $u$ sees in slot $t$.

The formulation in the OPL language is as follows:

```
minimize with linear relaxation
   sum (u in U, t in S: t < 4*n-2)
     dist[team_assigned[u,t,0],
     team_assigned[u,t+1,0]]
subject to {
forall(u in U, t in S)
    team_assigned[u,t,1]
      = opponents[t,team_assigned[u,t,0]];

//Constraints (1)&(2)
forall(t in S) {
    distribute( all(i in G) 1,
       all(j in T: opponents[t,j] < 0)
       -1*opponents[t,j],
    all(u in U) team_assigned[u,t,0]); };

//Constraint (3)
forall(u in U)
    atleast(all(i in T) 1, all(i in T) i,
       all(t in S) team_assigned[u,t,0]);

//Constraint (4)
forall(u in U, t in S: t<= (4*n-2)-(n-d1-1) )
   alldifferent(all(r in [0..n-d1-1])
       team_assigned[u,t+r,0]);

//Constraint (5)
forall(u in U, t in S: t<= (4*n-2)
   -(floor(n/2)-d2-1) )
   alldifferent(all(r in [0..floor(n/2)-d2-1],
   y in [0..1]) team_assigned[u,t+r,y]);

search {
    forall(t in S)
       forall(u in U)
           tryall (i in T: opponents[t,i] > 0)
               team_assigned[u,t,0] = i; };
```

## Acknowledgments

The authors would like to thank Heather Barringer and Jonathan Levine, two former MBA students at the Tepper School of Business, for their contribution. Both were involved in the development of the 2006 MLB umpire schedule that started as part of an MBA-level project course in the spring semester of 2005.

## References

Briskorn, D. 2008. *Sports Leagues Scheduling: Models, Combinatorial Properties, and Optimization Algorithms*. Springer Verlag, Berlin.

Duarte, A. R., C. C. Ribeiro, S. Urrutia, E. H. Haeusler. 2007. Referee assignment in sports leagues. E. Burke, H. Rudová, eds. *Practice and Theory of Automated Timetabling VI, Lecture Notes in Computer Science*, Vol. 3867. Springer, Berlin, 158–173.

Easton, K., G. Nemhauser, M. Trick. 2001. The traveling tournament problem description and benchmarks. T. Walsh, ed. *Proc. Seventh Internat. Conf. Principles Practice Constraint Programming* (*CP'01*), *Lecture Notes in Computer Science*, Vol. 2239. Springer, Berlin, 580–584.

Easton, K., G. Nemhauser, M. Trick. 2004. Sports scheduling. J. Y.-T. Leung, ed. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Boca Raton, FL, 52-1–52-19.

Evans, J. R. 1988. A microcomputer-based decision support system for scheduling umpires in the American Baseball League. *Interfaces* **18**(6) 42–51.

Evans, J. R., J. E. Hebert, R. F. Deckro. 1984. Play ball!—The scheduling of sports officials. *Perspect. Comput.: Appl. Academic Sci. Community* **4**(1) 18–29.

Farmer, A., J. S. Smith, L. T. Miller. 2007. Scheduling umpire crews for professional tennis tournaments. *Interfaces* **37**(2) 187–196.

Kirkpatrick, S., C. D. Gelatt, M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* **220**(4598) 671–680.

Lenstra, J. K., A. H. G. Rinnooy Kan. 1981. Complexity of vehicle routing and scheduling problems. *Networks* **11**(2) 221–227.

Ordonez, R. I. L. E. 1997. Solving the American League umpire crew scheduling problem using constraint logic programming. Doctoral dissertation, Illinois Institute of Technology, Chicago.

Pinedo, M., X. Chao. 1999. *Operations Scheduling*. Irwin/McGraw-Hill, Boston.

Rasmussen, R. V., M. A. Trick. 2008. Round robin scheduling—A survey. *Eur. J. Oper. Res.* **188**(3) 617–636.

Trick, M. A. 2003. Integer and constraint programming approaches for round robin tournament scheduling. E. K. Burke, P. DeCausmaecker, eds. *Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science*, Vol. 2740. Springer, Berlin, 63–77.

Trick, M. A. 2009. Challenge traveling tournament instances. Retrieved February 6, 2010, http://mat.tepper.cmu.edu/TOURN/.

Vidal, R. V. V. 1993. *Applied Simulated Annealing*. Springer Verlag, Berlin.

Wright, M. B. 1991. Scheduling English cricket umpires. *J. Oper. Res. Soc.* **42**(6) 447–452.

Wright, M. B. 2004. A rich model for scheduling umpires for an amateur cricket league. Working paper, Lancaster University Management School, Lancaster, UK.

Yildiz, H. 2008. Methodologies and applications for scheduling, routing and related problems. Doctoral disseration, Carnegie Mellon University, Pittsburgh.

Thomas E. Lepperd, Director, Umpire Administration, writes: "This is to confirm that Major League Baseball used assignment schedules for our umpires for the 2006, 2008, and 2009 playing seasons that were created by a team led by Michael Trick. We have found their process to be significantly easier, more efficient, and able to produce much better results than our prior process."