The Traveling Tournament Problem Description and Benchmarks

Kelly Easton[1], George Nemhauser[1], and Michael Trick[2]

[1] School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, Georgia USA
30332
{kelly.easton,george.nemhauser}@isye.gatech.edu
[2] Graduate School of Industrial Administration
Carnegie Mellon
Pittsburgh, PA USA
15213
trick@cmu.edu

Abstract. The *Traveling Tournament Problem* is a sports timetabling problem that abstracts the important issues in creating timetables where team travel is an important issue. Instances of this problem seem to be very difficult to solve even for very small cases. Given the practical importance of solving instances similar to these, this makes this problem an interesting challenge for combinatorial optimization techniques. We introduce this problem, give some interesting classes of instances and give some base computational results.

1 Introduction

Professional sports leagues are big businesses around the world. In the United States, television networks pay more than $400 million per year for nationally televised baseball games alone, with that much again for local presentations. Even college basketball leagues such as the Atlantic Coast Conference take in more than $30 million per year in radio and television fees. The UK football team Manchester United, a publicly traded company, has a market capitalization of more than £400 million; the Premiership receives more than £100 million for overseas television rights alone. The Italian Serie A are Spanish Premira Liga are not far behind in total income.

One key to such income levels is the schedule the teams play. No rights-holder wants to pay large sums only to get unattractive teams playing on a prime date. Teams do not want to see their large investments in players and infrastructure undermined by poor scheduling. Fans, who ultimately provide the income for the leagues, are also greatly affected by schedules.

In addition to these large, successful leagues, there are a countless number of smaller leagues that need schedules. These leagues range from lower divisional professional sports down to weekend recreational leagues, and often have needs that make finding suitable schedules difficult.

Given these demands, it is not surprising that creating sports schedules is an active research area. Examples of recent papers include schedules for minor league baseball (Russell and Leung [8]), college basketball (Ball and Webster [1]; Nemhauser and Trick [7]; Walser [13]; Henz [5]), Australian basketball (de Werra, Jacot-Descombes, and Masson [16]), and Dutch professional football (Schreuder [11]).

Despite this flurry of research, there are important issues that have not been well studied. This research was inspired by work done for Major League Baseball (MLB) in the United States. MLB, together, with the National Football League, the National Basketball Association, and the National Hockey League, is one of the "Big Four" sports in the United States, dominating the sports airwaves during the summer months. From a scheduling point of view, creating a reasonable schedule is a daunting task, since thirty teams play 162 games each over a 180 day season that stretches from early April to the end of September. In terms of total number of games, no published work has addressed problems anywhere near this size.

The MLB schedule is further complicated by the distances involved. Teams in MLB stretch from the American east coast to the west coast and from Canada to the tip of Florida. In order to reduce the amount of travel, teams go on "road trips", where they will visit a number of opposing teams before returning home (this contrasts to leagues with just one or two games in a week where teams return home after every game). Total travel becomes an important issue for MLB teams.

In addition this wish to keep travel down, teams are also concerned with more traditional issues with regards to their home and away patterns. No team likes to be away more than two weeks or so (corresponding to visiting 3 or 4 teams since teams play multiple games before moving on), nor do teams want to be home for longer than that period. Issues with the home/away pattern lead to questions of the ideal flow of the schedule, which can vary from league to league.

For any particular year, the MLB schedule instance is a very complicated set of requirements and requests. For the 1999-2000 schedule, team requests alone amounted to more than one hundred pages of questionnaire responses. Added to this were a large number of television, league, and union requests and requirements. Simply defining the problem is a difficult task.

They key to the MLB schedule, however, is the conflict between travel distances and the home/away pattern. If this can be solved in a reasonable amount of time, additional constraints and objectives can be added to approach the real instance. Without an approach for the core problem, however, it is unlikely that suitable schedules are creatable (it should be noted that MLB has used a husband-and-wife team named Henry and Holly Stephenson to create their schedules for two decades: their ability to create playable schedules without apparently using advanced combinatorial optimization software is quite amazing).

This conflict between travel and flow is not unique to MLB. Any time teams travel from one opponent to another leads to issues of distance and flow. In college basketball, some leagues work on a Friday-Sunday schedule where teams travel from their Friday game to their Sunday game directly.

We propose a problem class called the *Traveling Tournament Problem* which abstracts the key issues in creating a schedule that combines travel and home/away pattern issues. While it seems that either current scheduling software (which concentrates on home/away patterns) or insights from the Traveling Salesman Problem (which the distance issues seem to mimic) would make this problem reasonably easy to solve, it seems that the combination make the problem very difficult. Even instances with as few as eight teams are intractable relative to the state-of-the-art. This makes the problem attractive as a benchmark: it is easy to state and the data requirements are minimal. The fact that multiple communities, including the integer programming and constraint programming communities, have studied similar problems without addressing this particular problem contributes to its interest, for this problem seems a good problem for contrasting approaches and for exploring combining methods.

In the next section, we define the traveling tournament problem and show its computational complexity. Section 3 gives a variety of approaches for solving this problem; the following section develops particular instance classes and gives our arguments for the particular choices we have made. This section also gives computational results for these instances We conclude with some speculations on approaches that may work well.

2 The Traveling Tournament Problem

Given n teams, n even, a *round-robin tournament* is a tournament among the teams so that every team plays every other team. Such a tournament has $n-1$ slots during which $n/2$ games are played. For each game, one team is denoted the *home team* and its opponent is the *away team*. As suggested by the name, the game is held at the venue of the home team (this differs from other situations where all teams travel to a single venue). A *double round-robin tournament* has $2(n-1)$ slots and has every pair of teams plays twice, once at home and once away for each team.

Distances between team sites are given by an n by n distance matrix D. When a team plays an away game, it is assumed to travel from its home site to the away venue. When playing consecutive away games, teams travel from one away venue to the next directly. Each team begins the tournament at its home site, and must return to home at the end of the tournament.

Consecutive away games for a team constitute a *road trip*; consecutive home games are a *home stand*. The *length* of a road trip or home stand is the number of opponents played (not the travel distance).

The *Traveling Tournament Problem (TTP)* is as follows:

Input: n, the number of teams; D an n by n integer distance matrix; L, U integer parameters.

Output: A double round robin tournament on the n teams such that

- The length of every home stand and road trip is between L and U inclusive, and
- The total distance traveled by the teams is minimized.

In addition to the basic constraints, there may be additional requirements on the solution. These include:

- Mirrored. The double round robin tournament must have a round robin tournament in the first $n-1$ slots and then have the same tournament with venues reversed in the second $n-1$ slots.
- No Repeaters. There are no teams i, j such that i plays at j and then j plays at i in the next slot.

We will refer to the two variants (note that mirrored tournaments have no repeaters) as the Traveling Tournament Problem/Mirrored (TTP/Mirror) and the Traveling Tournament Problem/No Repeaters (TTP/No Repeat).

The parameters L and U define the tradeoff between distance traveled and the length of the home stands and road trips. For $L = 1$, $U = n$ allows for team schedules as short as the length of the traveling salesman tour of the cities; for U small, the team has to return often to the home site, so the schedules resemble that of vehicle routing solutions.

For $U = 1$, it is easy to see that there is no solution for $n > 2$. In this case, the only feasible home/away sequences are alternating home and away. There are only two such sequences (one beginning at home and one beginning away), so if there are more than 2 teams, two teams will have the same sequence. Such teams, however, cannot play each other so no round robin tournament is possible.

For $U = 2$, all away trips consist of either a single team or pairs of teams. This structure corresponds to a (slight generalization of) *matching*. This suggests this case may be polynomially solvable, though we have not found an algorithm to date.

2.1 Solution Methods

The Traveling Tournament Problem, is an intriguing problem not just for its modeling of issues of interest to real sports leagues. First, the problem combines issues of feasibility (with regards to issues of home/away patterns) and optimality (with regards to the distance traveled). Roughly, constraint programming excels at the former (see, for instance, Henz [5]) while integer programming does better at the latter. This combination seems to be difficult for both methods, making the TTP a good problem for exploring combinations of methods. Second, even small instances seem to be difficult. While $n = 4$ leads to easy instances, $n = 6$ is a challenging problem, and $n = 8$ is still unsolved for our sample distance matrices.

Fundamental for our approaches to solving this problem is the generation of tight lower bounds. The most useful bound is given by determining the minimal amount of travel for each team independent of any other team constraint.

Given a single team i, what is the minimal amount it must travel? This problem can be solved in a number of ways. For relatively small n and U, the easiest way to solve this is to generate all subsets of the teams of size between L and U. For each subset, there is an optimal tour for i visiting the subset. This value can be found by enumeration (if the set is small enough) or through combinatorial optimization techniques for the TSP (see, for instance Caseau and Laburthe [3] for constraint programming methods).

We are then faced with the problem of finding a collection of the subsets of minimum cost that contains all of the cities. If the set of subsets for city i is denoted S, with members s_j each with a cost of c_j, a formulation of the problem to be solved is

Minimize $\sum_j c_j x_j$
Subject to
$\sum_{\{j:k \in s_j\}} x_j = 1$ for all $k \in \{1..n\}, k \neq i$
$x_j \in \{0,1\}$

This subproblem can be solved with standard techniques (we use either integer programming or constraint programming within OPL [6]).

Summing over all teams, we get a bound on the overall TTP. We call the resulting lower bound the *Independent Bound (IB)*.

To find feasible solutions and to prove the optimality of those solutions, we have developed two approaches: a constraint programming approach and an integer programming approach.

2.2 Constraint Programming

In Nemhauser and Trick [7], we solved a practical sports scheduling approach with a three phase approach: in the first phase, a collection of home/away patterns were chosen; in the second, game assignments were made consistent with those patterns; in the final phase, a team was assigned to each pattern, completing the schedule. This approach was improved by Henz [5], who suggested using constraint programming for each of the phases.

A straightforward implementation of this three-phase method does not work for the TTP: the number of feasible pattern sets is enormous even for small n. We can, however, modify this approach to solve problems up to size 6 and provide approximate solutions for larger n.

To get a good upper bound, we would like to find a good feasible solution. Since the objective is to minimize travel, it is generally the case that home/away patterns with fewer trips are better. Rather that generate all pattern sets, we can generate pattern sets in order of the number of trips they contain. This is a straightforward problem that can solved with most constraint programming packages.

Once we have a pattern set with a small number of trips, we can assign teams to the pattern set and minimize the distance they travel. This again is a straightforward problem (see Henz [5] for one closely related formulation).

By starting with a small number of trips, we generally quickly find a very good feasible solution. For small instances, we can extend this to prove optimality by strengthening the IB lower bound, as follows:

Given a feasible solution with K trips, we resolve the lower bound with an additional constraint that the number of trips (over all teams) is at least K. If this bound is no better than our feasible solution, we can terminate; otherwise we enumerate all feasible pattern sets with K trips, and resolve the bound with $K+1$.

The resulting method generally finds very good solutions quickly, and can prove optimality for small instances. For larger instances, the enumeration required before IB becomes binding can be very time consuming.

2.3 Integer Programming Approach

An alternative formulation is a generalization of the IB formulation that fully models the problem. This approach requires the generation of all possible trips; for example, Team 1 plays at (Team 4, Team 2, Team 3) starting in slot 3 is one trip. These trips are the foundation for an integer programming model. Every trip has a binary variable x_j that is 1 if and only if trip j is selected. The formulation is then fairly straightforward. The following are the constraint sets:

- Each team must play each opponent exactly once.
- Each team must play exactly one game per slot.
- Trips cannot be scheduled back to back (so no away trip followed by another away trip)
- Every U slots must have at least one trip.

We can also use the results for IB to add constraints that force every team to travel at least its lower bound. This constraint substantially improves run time.

The objective is the sum of distances associated with the selected trips.

3 Instance Classes and Computational Results

We propose two problem classes for algorithmic experiments of the TTP. The first is an artificial set of instances designed to determine the effect of the TSP aspects of the TTP. The second is a series of instances from Major League Baseball which provided the original inspiration for this work.

3.1 Circle instances

Arguments for the complexity of TTP revolve around the embedded traveling salesman problem. It is not clear, however, that the TTP is easy even if the TSP is trivial. We explore this with this instance class where the TSP is easily solved (for which the solution is unique) but the TTP still seems to be challenging.

The n node circle instance (denoted CIRCn) has distances generated by the n node circle graph with unit distances. In this graph, the nodes are labeled $0, 1, \dots n-1$; there is an edge from i to $i+1$ and from $n-1$ to node 0, each with length 1. The distance from i to j (with $i > j$) is the length of the shortest path in this graph, and equals the minimum of $i-j$ and $j-i+n$.

In this graph, $0, 1, \dots, n-1$ gives the optimal TSP tour. Does this make the TTP easy?

We suggest two types of instances for each size: one with $U = n$ and one with $U = 3$ ($L = 1$ in each type).

3.2 National League Instances

As given in the introduction, the primary impetus for this work was an effort to find schedules for Major League Baseball. Unfortunately, MLB has far too many teams for the current state-of-the-art for finding optimal solutions. MLB is divided into two leagues: the National League and the American League. Almost all of the games each team plays are against teams in its own league, so it is reasonable to limit analysis to an individual league. The National League has 16 teams; the American League has 14 teams. Leagues are not defined geographically: each league has teams on both the US east coast and US west coast and scattered in between. Each league also has a single Canadian team (Montreal in the NL, Toronto in the AL).

We have generated the National League distance matrices by using "air distance" from the city centers. To generate smaller instances, we simply take subsets of the teams. In doing so, we create instances NL4, NL6, NL8, NL10, NL12, NL14, and NL16, where the number gives the number of teams in the instance.

3.3 Computational Results

We have attempted to solve the benchmark instances using a wide variety of techniques, including those given in previous sections. In general, size 4 instances are trivial, size 6 instances are difficult, and size 8 and larger instances are unsolved. In Table 1, we give bound values for each of the instances. Computation time seems less interesting for these instances at this stage due to their difficulty. In short, size 4 problems take at most a couple of seconds, size 6 solutions are found in between 1 and 4 hours, and we have spent days of computation time on the size 8 instances without proving optimality. All of these results are on the challenge page associated with this work: `http://mat.gsia.cmu.edu/TOURN`.

4 Conclusions and Future Directions

We propose the Traveling Tournament Problem as a benchmark problem for two primary reasons:

Name	U	IB	LB	UB	Optimal?
CIRC4	3	16	20	20	Y
CIRC6	3	60	64	64	Y
CIRC8	3	128			
CIRC10	3	220			
CIRC12	3	348			
CIRC14	3	588			
CIRC16	3	832			
CIRC18	3	1188			
CIRC20	3	1400			
CIRC4	3	16	20	20	Y
CIRC6	5	36			
CIRC8	7	64			
CIRC10	9	100			
CIRC12	11	144			
CIRC14	13	196			
CIRC16	15	256			
CIRC18	17	324			
CIRC20	19	400			
NL4	3		8276	8276	Y
NL6	3	22969	23916	23916	Y
NL8	3	38670	38870	41113	
NL10	3				
NL12	3				
NL14	3				
NL16	3	248,852	248,852	312,623	

Table 1. Benchmark Results for Challenge Instances

1. The problem has practical importance in modeling important issues from real sport schedules
2. The mix of feasibility and optimality, together with no long history in either field, make the problem interesting to both the operations research and constraint programming communities.

The resulting instances seem to be unusually difficult for either constraint programming or integer programming alone. One interesting look at these instances has been given by Benoist, Laburthe, and Rottembourgh [2]. They combine lagrangean relaxation and constraint programming to attack this problem. While their results to date have not been competitive with the techniques in this work, their paper does exactly what we hoped with these instances: spurring research in combining different methods to solve hard combinatorial problems.

References

1. Ball, B.C. and D.B. Webster. 1977. "Optimal scheduling for even-numbered team athletic conferences", AIIE Transactions 9, 161-169.
2. Benoist, T., F. Laburthe, and B. Rottembourg, 2001. "Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems", CPAI-OR, Wye College, UK, 15-26.
3. Caseau, Y. and F. Laburthe. 1997. "Solving Small TSPs with Constraints", Proceedings of the 14th International Conference on Logic Programming, L. Naish, Ed., The MIT Press.
4. Henz, M. 1999. "Constraint-based Round Robin Tournament Planning", Proceedings of the 1999 International Conference on Logic Programming, Las Cruces, NM.
5. Henz, M. 2000. "Scheduling a Major College Basketball Conference: Revisted", Operations Research, to appear.
6. ILOG. 2000. "ILOG OPL Studio", User's Manual and Program Guide.
7. Nemhauser, G.L. and M.A. Trick. 1998. "Scheduling a Major College Basketball Conference", Operations Research, 46, 1-8.
8. Russell, R.A. and J.M Leung. 1994. "Devising a cost effective schedule for a baseball league", Operations Research 42, 614-625.
9. Schaerf, A. 1999. "Scheduling Sport Tournaments using Constraint Logic Programming", Constraints 4, 43-65.
10. Schreuder, J.A.M. 1980. "Constructing timetables for sport competitions", Mathematical Programming Study, 13, 58-67.
11. Schreuder, J.A.M. 1992. "Combinatorial aspects of construction of competition Dutch Professional Football Leagues", Discrete Applied Mathematics 35, 301-312.
12. Wallis, W.D. 1983. "A tournament problem", Journal of the Australian Mathematics Society Series B, 24, 289-291.
13. Walser, J.P. 1999. Integer Optimization by Local Search: A Domain-Independent Approach, Springer Lecture Notes in Artificial Intelligence 1637, Springer, Berlin.
14. de Werra, D. 1980. "Geography, games, and graphs", Discrete Applied Mathematics 2, 327-337.
15. de Werra, D. 1988. "Some models of graphs for scheduling sports competitions", Discrete Applied Mathematics 21, 47-65.
16. de Werra, D., L. Jacot-Descombes, and P. Masson. 1990. "A constrained sports scheduling problem", Discrete Applied Mathematics 26, 41-49.