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 {keaston,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 two issues in creating timetables: home/away pattern feasibility and team travel. Instances of this problem seem to be very difficult even for a very small number of teams, making it an interesting challenge for combinatorial optimization techniques such as integer programming and constraint programming. We introduce the problem, describe one way of modeling it, and give some interesting classes of instances with base computational results.

1 Introduction

This research was inspired by work done for Major League Baseball (MLB) in North America. Creating a reasonable MLB 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. While creating a playable schedule involves juggling hundreds of requests and requirements, the key issues for a schedule revolve around travel distance and "flow", the pattern of home and away games in the schedule.

While teams wish to limit the total amount they travel, teams are also concerned with more traditional issues with respect 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.

The 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. This has been explored by Campbell and Chen [3] where the goal was to minimize the distance traveled over such weekend pairs. Russell and Leung [6] had a similar travel objective in their work for scheduling minor league baseball. In both of these cases, the limit on the number of consecutive away games was set to two, leading to interesting bounds based on variants of the matching problem. Many other references to sports scheduling problems can be found in Nemhauser and Trick [5].

We propose a problem class called the *Traveling Tournament Problem (TTP)* which abstracts the key issues in creating a schedule that combines travel and home/away pattern issues. While it seems that either insights from sports scheduling problems that involve complex home/away pattern constraints or from the Traveling Salesman Problem (which the distance issues seem to mimic) would make this problem reasonably easy to solve, the combination makes this 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 neither the integer programming nor the constraint programming community has studied this type of problem contributes to its interest. The TTP seems a good medium for contrasting approaches and for exploring combinations of methods.

2 The Traveling Tournament Problem

Given n teams with n even, a double round robin tournament is a set of games in which every team plays every other team exactly once at home and once away. A game is specified by and ordered pair of opponents. Exactly $2(n-1)$ slots or time periods are required to play a double round robin tournament. Distances between team sites are given by an n by n distance matrix D. Each team begins at its home site and travels to play its games at the chosen venues. Each team then returns (if necessary) to its home base at the end of the schedule.

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 TTP is defined 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.

The parameters L and U define the tradeoff between distance and pattern considerations. For $L = 1$ and $U = n - 1$, a team may take a trip equivalent to a traveling salesman tour. For small U, teams must return home often, so the distance traveled will increase.

3 Modeling

The TTP is an intriguing problem not just for its modeling of issues of interest to real sports leagues. First, the problem combines issues of feasibility (the home/away patterns) and optimality (the distance traveled). Roughly, constraint programming excels at the former (see, for instance, Henz [4]) 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. 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.

The generation of tight lower bounds is fundamental to proving optimality. A simple lower bound is obtained by determining the minimal amount of travel for each team independent of any other team constraint. This problem, while formally difficult (it can easily be seen to be equivalent to a capacitated vehicle routing problem), can be solved easily for the problem sizes of interest. The sum of the team bounds gives a lower bound (the Independent Lower Bound or ILB) on the TTP. We can then use this lower bound to attack the TTP.

A straightforward constraint programming formulation of this problem, even armed with the ILB, cannot solve instances larger than $n = 4$. Instances with $n = 6$ require some interesting search techniques. We first find a good upper bound, then we work to increase the lower bound from ILB.

The key to our search is to order solutions by the number of trips taken by the the teams. In general, fewer trips means less distance traveled because a team does not have to return home too often. Let a pattern be a vector of home and away designations, one for each slot. Let a pattern set be a collection of patterns, one for each team. It is easy generate pattern sets in increasing order of the number of trips. For a given pattern set, forcing a solution to match that set is a much easier problem, and is the basis of a large part of the sports scheduling literature (see [5] for references). We can therefore generate pattern sets by increasing number of trips and find the minimum length distance for each pattern set. Once we have a feasible solution, we can add a constraint that we only want better solutions, which will further speed the computation.

We do not want, however, to work with all the pattern sets: there are far too many even for $n = 6$. Instead, we can modify ILB to include a minimum total number of trips constraint. Once the ILB with this constraint is above our feasible solution, we know that we do not need to consider any pattern with more trips.

This method generally finds very good solutions quickly and can prove optimality for small instances. For larger instances, we have worked on a combination of integer and constraint programming methods involving column generation approaches [1]. In these models, the variables correspond to higher level structures, including road trips, homestands, and even complete team schedules. Constraint programming methods are used to generate variables that are then combined using integer programming techniques. Success depends heavily on the initial set of variables and on the branching rules used. For more detail on this, see the longer version of this paper, available from the web page http://mat.gsia.cmu.edu/TTP.

4 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.

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 (and 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 $0, 1, \ldots n-1$; 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, $0, 1, \ldots, n-1$ gives the optimal TSP tour. Does this make the TTP easy?

National League Instances. As stated 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.

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. All of these instances are on the challenge page associated with this work: `http://mat.gsia.cmu.edu/TOURN`.

4.1 Computational Results

We have attempted to solve the benchmark instances using a wide variety of techniques, including those given in Section 3. 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 some 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 (the results in the table are the best bounds from all of our efforts).

Name	U	IB	LB	UB	Optimal?
NL4	3		8276	8276	Y
NL6	3	22969	23916	23916	Y
NL8	3	38670	38870	41113	
NL16	3	248,852	248,852	312,623	
CIRC4	3	16	20	20	Y
CIRC6	3	60	64	64	Y
CIRC8	128	128	148		

Table 1. Some Benchmark Results for Challenge Instances

5 Conclusions and Future Directions

We propose the TTP as a benchmark problem for two primary reasons:

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 proposed instances seem to be unusually difficult for either constraint programming or integer programming alone. One interesting study of some of these instances has been given by Benoist, Laburthe, and Rottembourgh [2] who propose an algorithm combining lagrangean relaxation and constraint programming. While their results to date have not been competitive with the techniques in this work, their paper does exactly what we hoped would happen with these instances: spurring research in combining different methods to solve hard combinatorial problems.

References

1. Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H.Vance. 1998. "Branch-and-Price: Column Generation for Huge Integer Programs", Operations Research **46:** 3, 316- 329.
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. Campbell, R.T., and D.S. Chen, 1976. "A Minimum Distances Basketball Scheduling Problem", in *Optimal Strategies in Sports*, S.P. Ladany and R.E Machol (eds.), North-Holland, Amsterdam, 32-41.
4. Henz, M. 2001. "Scheduling a Major College Basketball Conference: Revisted", Operations Research, 49:1,.
5. Nemhauser, G.L. and M.A. Trick. 1998. "Scheduling a Major College Basketball Conference", Operations Research, 46, 1-8.
6. Russell, R.A. and J.M Leung. 1994. "Devising a cost effective schedule for a baseball league", Operations Research 42, 614-625.