A Tiling Approach for Fast Implementation of the Traveling Tournament Problem

Amotz Bar-Noy and Douglas Moody

City University of New York Graduate Center
New York, NY
amotz@sci.brooklyn.cuny.edu, dmoody@citytech.cuny.edu

1 Introduction

Sports in society today are played by millions of individuals at the professional, scholastic and amateur levels. The success of the leagues and tournaments often lies in the ability to generate a "good" schedule. Each league or tournament has a variety of constraints and objective measures to be used in the determination of a good schedule. Availability of venues, the order of opponents, travel time and distance are but of a few of the myriad of issues considered by the sports league scheduler.

The Traveling Tournament Problem (TTP), documented by Easton et al. in [4], describes a typical sports scheduling challenge. Specific instances and records can be found in [14]. The TTP is a double round robin tournament to be played by n teams over (2n-2) periods or *weeks*, where each team plays every period (we do not consider the "mirrored" version of the problem). Three unique constraints of the TTP are:

1. Maximum "Road Trip" of three games: each team can play at most 3 consecutive games away from the team's home site before playing again at the home site. A *road trip* is defined as one or more consecutive games played away from the team's home site, before returning home again. It is assumed that a team starts and ends the season from its home site.
2. Maximum "Home Stand" of three games: each can play at most 3 consecutive games at its home site. A *homestand* is defined as one or more consecutive games played at the team's home site.
3. Repeater Rule: a team can not play an opponent away in time period k and then home in time period $k+1$, or vice versa.

The TTP seeks to minimize the distance traveled by each team. A distance matrix is used to calculate the distance from home to each opponent, and the distance between consecutive opponents. This calculation is done for each road trip, with the total being the *schedule distance*. The selection of the opponents and their order on the road trip is critical, while homestands have no bearing on the distance calculation. Effective development and placing of road trips will yield good solutions to the TTP.

The following sections describe related work and our general 3-phase approach, followed by a detailed description of each phase's techniques and steps. The final two sections present results to date, and future work.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 351–358. ISBN 80-210-3726-1.

2 Related Work

The initial approaches to the TTP problem, and its more general round-robin tournament problem, centered on constraint and integer programmer approaches. Variations of models using this approach can be found in [5], [6], [7], [9], and [15]. The approaches create models with variables for each opponent pairing, home or away venue and usually other variables that represent the constraint being studied. As the number of teams grows slowly, these variables grow exponentially, significantly reducing their effectiveness with teams > 12.

Heuristic approaches have also been used extensively. Regin in [11] looked at minimizing breaks (home game followed by an away game and vice versa) and patterns of home and away games. Pattern analysis and generation are also key components to the heuristic approach by Ribeiro in [12]. The pattern generation was then followed with local neighborhood swap techniques to improve the solution. Shen and Zhang in [13] generated patterns of team match-ups using a greedy approach.

As Henz described in [7], the TTP problem can also be viewed as a neighborhood search problem. Simulating annealing approaches in [1] and [10] take advantage of this perspective. Cardemil in [2] used tabu search logic.

Several of the above approaches relied on making several runs to find good starting neighborhoods. This step was followed by an extensive and costly local search effort searching for an optimal solution. We seek a fast implementation method that can come within 5-7% of the good solutions without using intensive resources through tiling. A tiling approach that constructed tiles and then timetabled was also used for course scheduling by Kingston in [8]. Courses are combined into tiles for forms (high school student years) for similar purposes, as our roadtrip tiles, and then timetabled into the scheduling grid.

3 Approach

Our approach is to model the road trips as “tiles”. Each tile will contain “blocks”, which represent individual games. A road trip of 3 opponents is considered as one tile, with three blocks. A teams’ schedule can be thought of as a series of tiles, with home games as spacers between the tiles. Figure 1 shows the scheduling grid and tiles for Team 1 and Team 2.

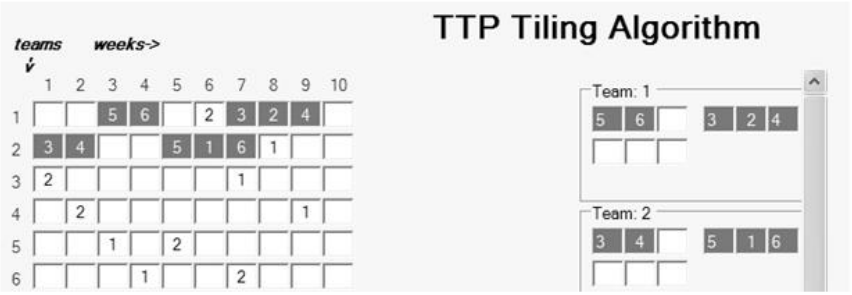


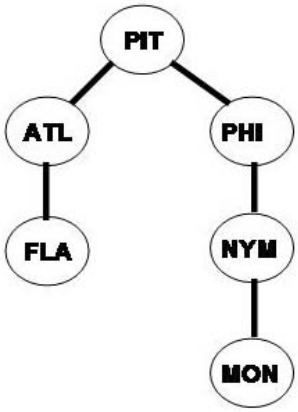
Fig. 1. Tile Placement for Teams 1 and 2 (shading cells indicate an away game)

For each team a set of tiles is created that seeks to minimize the distance traveled for a particular team without concern of any constraints involving other teams. These tiles are placed in a scheduling grid of n rows representing teams and (2n-2) columns representing weeks.

As tiles are placed, other cells of the grid are filled in to keep the schedule consistent with the tile placement. When there are no tiles remaining that can be placed, the tiles are broken into their component blocks, and placed as allowable by TTP constraints. If not all blocks can be placed, the block placement is backtracked in an attempt to find a solution. The backtracking may reach back to the tile placement step, causing reassignments of tiles. If all blocks can be placed, a solution is generated, its distance calculated. Backtracking is again employed to find additional solutions.

4 Phase I – Tile Creation

The creation of the tiles is done on a team by team basis. For each team a minimum spanning tree (MST) is created by upon the Prim algorithm described in [3]. The algorithm begins with the selection of a root node, or in our approach a team. All distance edges in the distance matrix, defined in the problem, are searched for the smallest edge, which has a vertex of a team in the tree, and a vertex of a team not in the tree. This edge is then added between the two teams. For the first branch, we are finding the nearest opponent of the root team. The second edge is the nearest team to the root team, or its first opponent. Edges are that are added to opponent will suggest the two vertices or teams will be on the same road trip or tile. Two edges both with having the root node as a vertex, suggest that the two opponents of the root team will be on different road trips. Figure 2 presents an example MST for the team Pittsburgh (PIT) in the nl6 problem in [4].



DISTANCE MATRIX

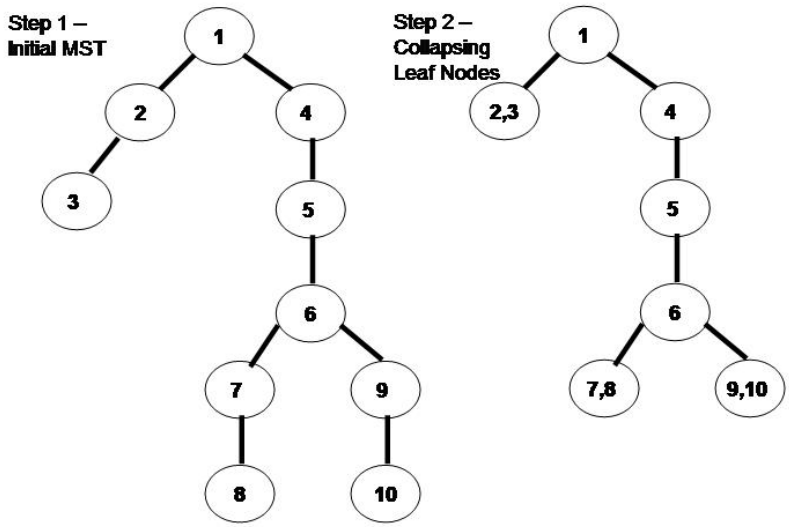
Home Team \ Away Team	ATL	NYM	PHI	MON	FLA	PIT
ATL	0	745	665	929	605	521
NYM	745	0	80	337	1090	315
PHI	665	80	0	380	1020	257
MON	929	337	380	0	1380	408
FLA	605	1090	1020	1380	0	1010
PIT	521	315	257	408	1010	0

Example: Distance from Team NYM to Team PHI is 80 miles

Fig. 2. MST for PIT in nl6 and the accompanying distance matrix.

Figure 2 suggests that the team PIT should have 2 road trips or tiles – one with ATL and FLA, and the other with PHI, followed by NYM and MON. If these two road trips are traveled by the team, the team will have the optimal minimum distance. This does not imply the league overall will have optimal minimal distance, but rather just this team.

We use a tree collapsing algorithm to create tiles from the tree structure. Separate trees are created with each tree having the root node of a team. The collapsing approach merges child nodes into their parent node. When the parent has 3 teams, a tile is made. When the parent must decide among its children, a greedy method is used, to pick the best set of 3 teams from the parent and children. Figure 3 provides a sample collapsing process.



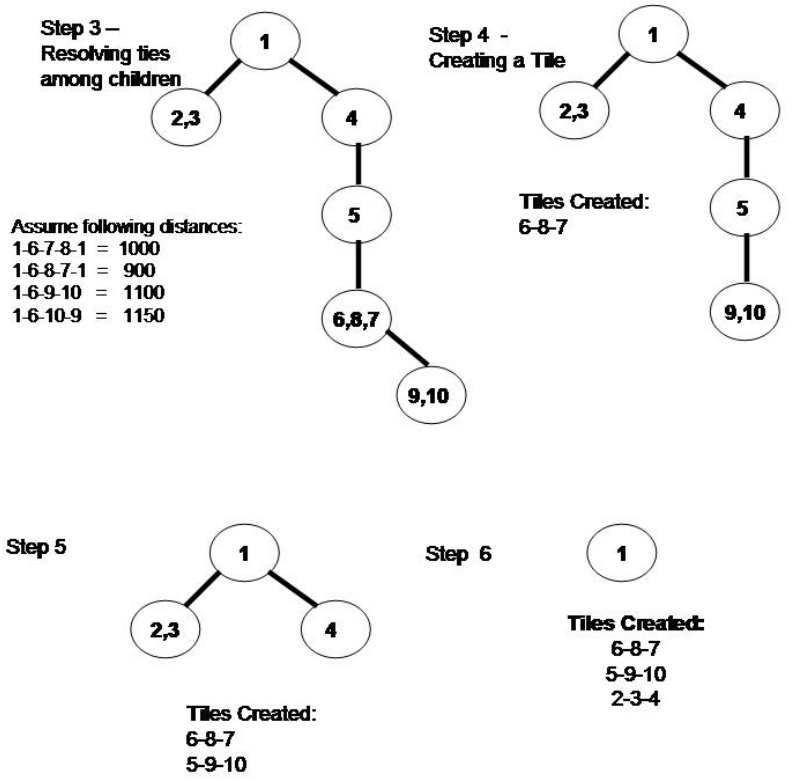


Fig. 3. Creation of Tiles through collapsing of the tree

The tile creation process creates ceiling((n-1)/3)*n tiles. Note that all tiles have 3 blocks, with the possible exception of the last tiles created for the team. At the root node, if both children have a weight of two nodes, two tiles of 2 blocks are created for each child.

5 Phase II – Tile Placement

The tile placement process places tiles, in team order by highest distance schedule, on the scheduling grid. All the tiles for a team are considered together. If necessary a tile is "rotated", or its opponents are re-ordered. Switching opponent 1 and 3 within the tile does not affect the distance of the tile, however other switches do impact the distance. Only switches resulting in a 10% or less impact are acceptable.

The tile is moved through the schedule week by week. After all weeks have been tried, the blocks within the tile (games) are rotated in order to find an acceptable placement. After each placement, consistency checks are made to ensure the league schedule meet all the TTP constraints.

When no more tiles can be placed, Phase III is executed, which breaks the unplaced tiles in individual blocks. Backtracking is then performed to rearrange the blocks and look for additional solutions. During the backtracking, some tiles are placed in a “tabu” like status, so they are ignored temporarily giving less costlier tiles the chance to be placed earlier.

One noteworthy aspect of our approach is that tiles are never broken and then formed into new tiles, referred to as *reconfiguration*. This process involves breaking 2 tiles in their component blocks, and regrouping the blocks into two new tiles.

6 Phase III – Block Placement

Phase III is similar to other solutions focused on constraint programming solutions ([5],[7],[9]). The phase breaks all remaining unplaced tiles into individual blocks. These blocks, or games, are now placed in the scheduling grid one by one. Backtracking is used when a set of placed blocks can no longer lead to a solution. Multiple solutions can be found for the set of placed tiles through this phase. Other works have used a constraint programming package, such as ILOG, to accomplish this task.

7 Results

The first set of results examines the success of MST collapsing algorithm to generate valid tiles, based upon solutions in [4]. For comparison purposes, we looked at the solution records for leagues with teams of 6,8 and 16. When multiple away games appear in the solution schedule, a tile can be created (deduced) to represent those games. We compared the blocks of these tiles with those that are generated form the MST collapsing algorithm. The results are shown in Figure 5.

These results show a high correlation between the roadtrips embedded in the solutions with the roadtrip tiles generated by the MST. For a low number of teams the percentage match is about 80% of the tiles consisting of over 85% of the games. In the higher number of teams, the number of tiles and associated games involved is about 60%. In this case, the tiles that are different only add 5-7% miles to the total distance.

The second set of results considers the generation of the solution set itself. For the small instances where n = 6, the tiling approach produced a distance of 24,102. When the number of allowable free blocks (the maximum number of blocks or games needed to be scheduled individually) for entering Phase III (versus immediate backtrack) was increased beyond 3n, our approach achieved the current goal of 23,916. This parameter ensures that sufficient tiles have been used before attempting individual block scheduling. This parameter could be relaxed, since the number of teams was small enough not to materially affect the length of phase III processing. Processing times for both solutions was less than 10 minutes on a 1.63 GHz computer with 1 Gig of memory, executing a Visual Basic custom application.

Solution Name	Total Solution Tiles	Total Games Solution Tiles	MST Complete Matching Tiles	MST 2 Partial Matching Tiles	% of MST Matching Tiles incl. Partial	% of MST Matching Games in Tiles
Easton – 6 teams	11	28	6	3	82%	86%
Easton – 8 teams	18	48	13	2	72%	90%
Cardil – 16 teams	83	211	25	27	63%	61%
Zhang – 16 teams (8/6/02)	83	219	17	40	69%	60%

Fig. 4. Comparison of tiles deduced from existing solutions versus MST collapsed tree generation

For n = 8 teams, a distance of 41,957 was achieved in under 30 minutes. This compares favorably to the existing best solution of 39,721, a 5.6% difference. During the tiling approach execution, 4,360 solutions were found. For 10 teams, the tiling approach achieved a distance of 62,916 compared to the existing best solution of 59,436, a 5.8% difference. The result took nearly an hour, however a solution of 68,204 was found within minutes. For 16 teams, Phase III processing needs further efficiencies to complete the single block processing inherent in that step.

8 Conclusion and further work

The MST collapsing approach, coupled with tile processing looks promising for producing very good, but not optimal, solutions in quick fashion. Our work to date shows that with a custom application program, we can create tiles and generate solutions within a short time, that come within 5-7% of existing solution records. As our Phase III constraint processing logic comes closer to commercial packages, our processing times (currently concentrated in Phase III processing) will decrease.

Improvement to our process will focus on consistency checks at the team level. Stronger analysis of the home and away pattern of a team at various points, can highlight inconsistencies, enabling necessary tile placement backtracking to occur earlier in the process. Also, Phase III processing may be replaced by a call to a constraint programming language such as ILOG.

References

1. Anagnostopoulos, A., Michel, L., Van Hentenryck, P., Vergados, Y. (2003) *A simulated annealing approach to the traveling tournament problem*, Proceedings CPAIOR'03, Montreal.
2. A. Cardemil. *Optimizacion de fixtures deportivos: Estado del arte y un algoritmo tabu search para el traveling tournament problem*. Master's the-

sis, Universidad de Buenos Aires, Departamento de Computacion, Buenos Aires, 2002.
3. Cormen, Thomas H. et al. Introduction to Algorithms. pp. 570-573. Boston: McGraw-Hill, 2001.
4. Easton, K., Nemhauser, G., Trick, M. (2001) *The traveling tournament problem: description and benchmarks*, in: Proceedings CP'01, Lecture Notes in Computer Science 2239, Springer, 580-585.
5. Easton, K., Nemhauser, G., Trick, M. (2003) *Solving the traveling tournament problem: a combined integer programming and constraint programming approach*, in: E. Burke and P. De Causmaecker (eds.), PATAT 2002, Lecture Notes in Computer Science 2740, Springer, 100-109.
6. Henz, M. (1999) *Constraint-based round robin tournament planning*, in: D. De Schreye (ed.), Proceedings of the International Conference on Logic Programming, Las Cruces, New Mexico, MIT Press, 545-557.
7. Henz, M. (2004) *Playing with constraint programming and large neighborhood search for traveling tournaments*, Proceedings PATAT 2004, Pittsburgh, USA.
8. Kingston, J. (2004) *A tiling algorithm for high school timetabling*, Proceedings PATAT 2004, Pittsburgh, USA.
9. Leong, G. (2003). Constraint programming for the traveling tournament problem. www.comp.nus.edu.sg/henz/students/gan_tiaw_leong.pdf
10. Lim, A., Zhang, X. (2003) *Integer programming and simulated annealing for scheduling sports competition on multiple venues*, Proceedings MIC 2003.
11. Regin, J.-C. (2001) *Minimization of the number of breaks in sports scheduling problems using constraint programming*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 57, 115-130.
12. Ribeiro, C.C., Urrutia, S. (2004) *Heuristics for the mirrored traveling tournament problem*, Proceedings PATAT 2004, Pittsburgh, USA.
13. Shen, H., Zhang, H. (2004) *Greedy Big Steps as a Meta-Heuristic for Combinatorial Search*. The University of Iowa AR Reading Group, Spring 2004 Readings. goedel.cs.uiowa.edu/classes/AR-group/04spring.html
14. Trick, M.A. Challenge Traveling tournament instances. Online document at http://mat.gsia.cmu.edu/TOURN/. Last update as of writing: January 18, 2006
15. Trick, M.A. (2003) Integer and constraint programming approaches for round-robin tournament scheduling, in: E. Burke and P. De Causmaecker (eds.), PATAT 2002, Lecture Notes in Computer Science 2740, Springer, 63-77.