An efficient and robust approach to generate high quality solutions for the Traveling Tournament Problem

Douglas Moody, Graham Kendall and Amotz Bar-Noy

City University of New York Graduate Center and School of Computer Science (Moody,Bar-Noy), University of Nottingham, UK (Kendall)

dmoody@citytech.cuny.edu, gxk@cs.nott.ac.uk, amotz@sci.brooklyn.cuny.edu

The Traveling Tournament Problem (TTP) describes a typical sports scheduling challenge. The TTP, which is based on the U.S. Major League Baseball (MLB), has specific instances with results available on the web. Several approaches have been proposed since the problem's creation. The "best" of these solutions use extensive resources in local search activities to find high quality solutions. We propose a tiling method that can produce a good quality solution, using a fraction of the resources documented in other approaches. Our solution can also be expanded to handle the additional real-world scheduling requirements, including unbalanced schedules within the MLB.

1 Introduction

The Traveling Tournament Problem (TTP) is a double round robin tournament to be played by *n* teams over (2*n*-2) periods or weeks, where each team plays in every period (we do not consider the "mirrored" version of the problem). The three constraints of the TTP are:

1) Maximum "**Road Trip**" of three games: each team can play at most three consecutive games away from the team's home site before playing again at the home site. For teams beginning the season with an away game, it is assumed travel for that game would begin at the home site. Likewise, teams ending the season with an away game, would require to travel from that opponent's location to the team's home site to complete the season.

2) Maximum "**Home stand**" of three games: each team can play at most three consecutive games at its home site.

3) Repeater Rule: a team cannot 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 available to define the distance between each team. This calculation is used for each road trip, with the total being the accumulated distance for all teams. The selection of opponents and their order within the road trip is critical, while home stands have no bearing on the distance calculation. Efficient road trips across all teams are more likely to yield good quality solutions to the TTP.

The TTP has served as a benchmark problem for sports scheduling over the past decade. Easton et al. [6] defines the problem, and the latest results are available at [14]. However, the TTP definition notes its simplification of the actual Major League Baseball. This simplification eliminates some key complexities of the MLB, notably:

Existence of unbalanced schedules – Teams in MLB play other teams within their division more frequently than teams in other divisions.

Unequal home and away games – Due to Inter-league play, teams do not play an equal number of home and away games against all teams. It also introduces constraints of teams from the same city playing on the same day. Kendall [7] notes this constraint in scheduling the English football league.

These two simplifications enable many solution approaches to take advantage of round-robin tournament scheduling for the TTP. The solutions that depend on a mirrored approach would not be able to produce an unbalanced schedule. Other approaches use simple tournament generators that also would not handle an unbalanced schedule. We propose an approach that can compete with existing TTP tournament based solutions sets, and also be extendable to handle the real MLB problem.

2 Related Work

The TTP has spawned a variety of methodologies to find good quality solutions. Kendall et al. [8] surveys this broad array of approaches. The work to date can be viewed in two phases. The first phase consists of traditional approaches including simulated annealing by Anagnostopoulos [2], Lims' [11] work on integer programming, the constraint programming approach by Leong [10] , and tabu search proposed by Di Gaspero and Schaerf [5]. Also special heuristics have been developed as used by Shen and Zhang [13]. The second phase focuses on the use of parallel processing and server farms. Van Hentenryck and Vergados [16], and Araujo [1] use a parallel implementation of existing algorithms to improve on published results.

The majority of the above methods have focused almost exclusively on local search operations to produce the best solutions. The initial neighborhood for the local search is usually generated at random, and is not necessarily feasible. Hence the starting "neighborhood" may be extremely far from the optimal solution, in terms of a distance measured by the number of "swaps", which are defined as moves between neighborhoods.

Our approach seeks to create an initial neighborhood that is a feasible solution closer to the optimal solution than random initial solutions. We use a tiling method, proposed by Kingston [9] for course scheduling. Our tiling method creates an initial solution that has minimal hard constraint violations, and contains a core set of game assignments for a high quality solution. The second phase of local search is used to remove constraint violations, and tune the solution.

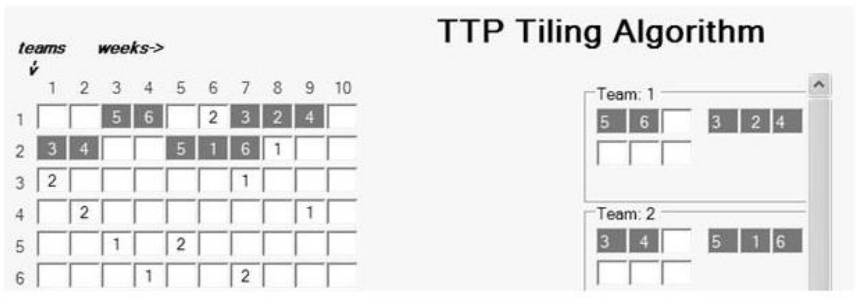
The concept of modeling the problem using away trips was also considered by Trick [15]. A new variable, representing a road trip is introduced into the model. The road trip variable is constructed as game assignments are made. This variable affects the value of the objective function, and hence the road trips are modified to reduce the objective function. Hence many road trips are considered for a given team during the running of the model. This approach differs from our approach, as we begin the scheduling process with existing road trip (tile) formations. During our scheduling phase, we do not re-formulate a tile, based upon other assignments. We may break a tile into its component parts, but these parts are not reassembled into another tile. Our approach is based on a fixed set of tiles.

3 Proposed Approach

Our approach is a two phase approach involving tiling, followed by a local search phase. The tiling phase creates an initial solution, which may not be feasible. The second phase (a local search) removes the hard constraint violations and improves the quality of the solution.

3.1 Tiling

We model the road trips of the TTP as "tiles". Each tile contains "blocks", which represent individual games. A road trip of three 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.



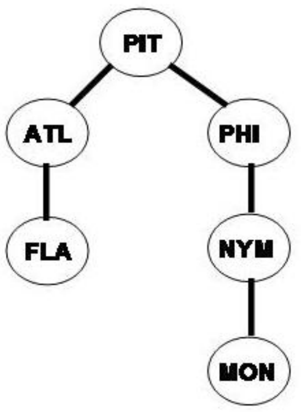
	1	2	3	4	5	6	7	8	9	10
1			5	6		2	3	2	4	
2	3	4			5	1	6	1		
3	2						1			
4		2							1	
5			1		2					
6				1			2			

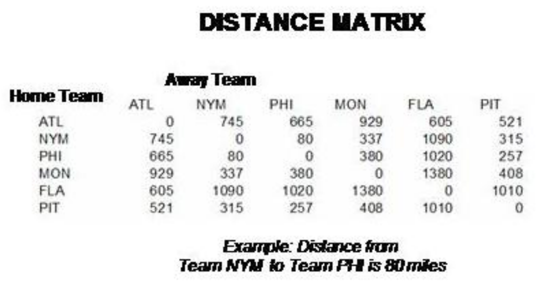
Fig. 1. Tile Placement for Teams 1 and 2 (shaded cells indicate an away game)

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

As tiles are placed, other cells of the grid are filled in to maintain schedule consistency with the tile placement. When no more tiles can be placed, the tiles are broken into their component blocks, and placed subject to TTP constraints. If all blocks cannot be placed, the consecutive home and away constraint is relaxed, allowing all blocks to be placed albeit in an infeasible solution.

The creation of the tiles is carried out on a team by team basis. For each team a minimum spanning tree (MST) is created by using the Prim [4] algorithm. The algorithm begins with the selection of a root node, in our approach this is a team. All distance edges in the distance matrix, defined in the problem, are searched for the smallest edge. Each edge 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 that are added to an opponent will suggest that the two vertices or teams will be on the same road trip or tile. Two edges having the same 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 documented in [14].





DISTANCE MATRIX

Home 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

Away Team

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

Fig. 2. Minimum Spanning Tree (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 the team completes these two road trips, the team may have travelled 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 three teams, a tile is created. When the parent must decide among its children, a greedy method is used, to pick the best set of three teams from the parent and children. Figure 3 provides a sample collapsing process.

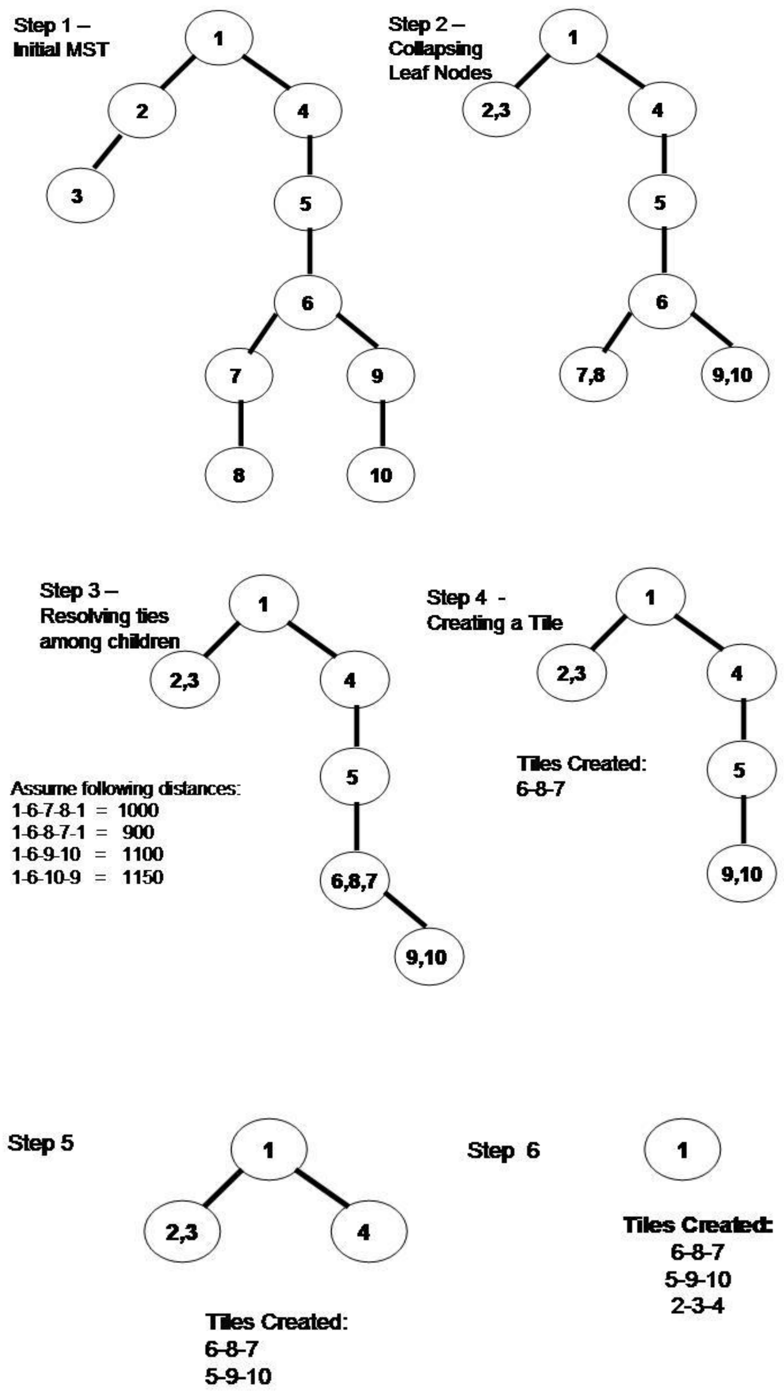


Fig. 3. Creation of Tiles through collapsing of the tree

The three team tile creation process creates *ceiling((n-1)/3)*n* tiles. Note that all tiles have three 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 two blocks are created for each child.

After tile creation, each tile is given a cost. This cost is calculated to capture some measurement of the impact on the objective function of breaking up the tile. The cost is the sum of a round-trip of the home team to each opponent within the tile, minus the actual distance to be traveled between teams within the tile. This cost is then used to prioritize the order that tiles will be placed in the schedule.

We select the tile with the highest cost. We place this tile on the first available space on the schedule without violating any constraints. We start with week one, and move forward checking all constraints for the three games represented in the tile. If the tile cannot be placed in week 1, we rotate the tile, by changing the order of the first and last teams in the road trip. This rotation does not alter the distance associated with the tile. The tile is moved through the schedule week by week until a placement can be found.

The tile placement process is continued for each tile in cost priority order. When no more tiles can be placed we break all tiles into individual games. The games are then placed in the schedule, starting from the first week of the schedule. We relax the maximum away games and home stand rules as well as the repeater rule constraints at this point to allow for placement of all games. When a game cannot be placed, given these relaxations, previous assignments are backtracked. If the backtracking of the individual games does not produce a feasible schedule, then the last place tile is backtracked.

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 two tiles in their component blocks, and regrouping the blocks into two new tiles.

3.2 Local Search

The local search phase of our approach is designed to remove any hard constraint violations and improve the quality of the schedule. When it is possible to place all tiles, a near optimal solution based on the quality of our tile set can be generated. The breaking of the tiles is where our initial solution degrades. Hence, the games placed singularly are the root of all hard constraint violations, and higher than optimal travel distances. Our local search seeks to reach the best local minimum of our initial solution, after performing a variety of "swaps". Each swap moves a set of games within the schedule while maintaining, or improving, feasibility and the objective function of the solution.

We employ the following sets of swaps during our local search phase:

Home / Away Swap – We swap two games, involving the same two teams, where each team is home in one game and away in the other game. The swap is done between two weeks.

Round Swap – All games for two weeks are moved between the two weeks.

Partial Round Swap - The partial round swap described in [2], moves a set of connected games between two weeks. We begin this operation by selecting two games in different weeks. The teams in these games create our swap set. We then add teams playing these two teams, in either week, to the swap-set. The process continues until all teams, in both weeks, are in the swap set or its complement. We then move all games involving teams in the swap set between weeks. We can also choose to move all teams in the complement of the swap set between the weeks.

The local search phase is an iterative process to optimize the impact of the above swap actions. We look at each swap in the order above, and analyze the schedule improvement, if any, of each possible swap. The swap with the largest improvement is performed to reach a new schedule. If no improvement can be found, we move to the next type of swap in the list above. The same process is used until no improvement can be found. After performing all possible swaps of the last type (partial round), we have reached a local minimum and stop the local search.

4 Results

Our two phase approach of tiling and local search enables us to produce a high quality solution with minimal resources. We only use a small fraction of the resources consumed by the best known approaches. Our platform is a 2.13 Ghz Dell laptop, using a .NET software application.

We use three approaches for comparison, presented in Table A. The first approach, described by Van Hentenryck and Vergados [16] is a parallel processing approach, carried out on clusters of dual-processor blade servers. They use a simulated annealing based Traveling Tournament approach first proposed by Anagnostopoulos [2]. The second approach by Di Gaspero and Schaerf [5] uses tabu search. The final comparison is with Araujo et al. [1], using parallel processing. This approach is also a two phase approach, with a random initial neighborhood construction, followed by a greedy search phase to reach a local minimum to produce a mirrored solution. The local search phase is iterated after a perbutation of the initial neighborhood.

Instance	Tiling Results	Van Hentenryck and Vergados Results [16]	Di Gaspero and Schaerf Results [5]	Araujo et al. (GRASP) Results [1]	Average % Difference from Tiling
NL 16	317,764	**267,194**	279,465	285,614	14.54%
NFL 16	266,231	**235,930**	238,581	N/A	12.21%
NFL 18	339,822	**296,638**	N/A	299,134	14.08%
NFL 20	406,463	**332041**	352947	359,748	16.71%
NFL 22	482,374	**412,812**	439,626	418,086	13.90%
NFL 24	544,354	**463,657**	499017	465,491	14.34%
CON16	354	N/A	**328**	342	5.67%
CON18	466	N/A	**418**	432	9.64%
CON20	568	520	**521**	522	9.02%

Table A: Results Comparison (best solutions in **bold**)

Instance	Tiling Time	Van Hentenryck and Vergados Time – Best [15]	Di Gaspero and Schaerf Time – [5]
NL 16	38	1,815	51,022.4
NFL 16	35	2,220	N/A
NFL 18	105	3,120	N/A
NFL 20	135	6,750	N/A
NFL 22	150	8,100	N/A
NFL 24	320	5,490	N/A
CON16	18	N/A	19,665
CON18	22	N/A	33,979
CON20	23	N/A	46,579

Table B: Time Comparison in seconds

We choose instances with a higher number of teams and the constrained constant distance instances of the problem, for comparison. The instances with fewer teams have been addressed by a variety of algorithms, whereas instances with 16 teams or more have been addressed successfully by only the above approaches.

Table A compares the results and resources of our tiling approach with the best-known approaches. Our tiling approach comes withinn 10-16% of the objective function for the TTP for all approaches. The tiling approach produces slightly better results where the number of games to be played by each team is divisible by three. This situation allows all games for a team to be placed in 3-team tiles. Hence our 16 and 22 team instances are slightly better in comparison with the other approaches.

For the constrained instances, our approach is even closer than the higher NFL instances. The construction and costing of the tiles is not important, since all distances are constant. Hence each tile for a team has the same cost, when the tile is broken. Our 16 team instance, dictating 5 tiles of 3 teams per tile, provides our best result. One factor in this result is that the games for each team can be constructed into a set of tiles with 3 games each. This enables more tiles to be placed in the initial neighborhood, increasing the quality of the initial solution.

We use only a few seconds compared to the substantive time used in the other approaches. Table B compares our times with the published times of the approaches with comparable metrics.

The key difference between our tiling approach and the other approaches is the initial neighborhood. Our initial neighborhood is built based upon tiles, which are high quality partial solutions. Hence our approach saves the time used by others in the local search to develop a high quality initial solution. We follow our tiling phase with an efficient greedy approach to quickly develop a solution relatively close to the best known solutions.

The most illuminating result is the NL16 figure for Araujo et al. in Table A. The result of 285,614 was achieved by running the GRASP algorithm in sequential mode outlined in [1] for 5 days. This algorithm is similar to our local search, because of its greedy nature. Both algorithms explore all possible swaps of a given category and perform the swap with the greatest improvement. When no improvements can be made, the local minimum is reached. Since the local search phases are similar, the key difference between the approaches is the construction of the initial neighborhood. The results in the table show the results of using GRASP on neighborhoods created, over a 5 day time span. After each local minimum is reached, random swaps are done to create a new neighborhood. Hence large numbers of initial neighborhoods are created over the 5 day processing period. The best of these neighborhoods led to only an 8.5% improvement of creating one initial neighborhood solution through tiling.

5 Conclusion and further work

Our approach indicates substantial resource savings in finding good quality solutions for the TTP. Our initial neighborhood, along with a minimal local search phase can produce high quality solutions. In our future work we will compare solution results among the best approaches, to understand the distance of our initial neighborhood from the best known solution sets. This will enable us to identify the transformations needed to move from our initial neighborhood more directly to the best solutions.

We also plan to expand the TTP to handle a complete Major League Baseball schedule, which was the original motivation for the problem. Albeit the actual MLB schedule has a wide variety of real-world constraints, we look to expand the TTP by implementing the actual unbalanced team schedules and inter-league play of the MLB master schedule. These instances will greatly increase the complexity of the schedule due to its unbalanced nature.

References

1. Araújo, A., Boeres, M.C., Rebello, V.E., Ribeiro, C.C., Urrutia, S., Exploring Grid Implementations of Parallel Cooperative Metaheuristics, A Case Study for the Mirrored Traveling Tournament Problem. Chapter 16, Metaheuristics Volume 39, US:Springer, 2007.
2. Anagnostopoulos A., Michel L., Van Hentenryck P., Vergados Y. A simulated annealing approach to the traveling tournament problem. Journal of Scheduling 2006;9:pp. 177–93.
3. Bar-Noy, A. and Moody, D. "A Tiling Approach for Fast Implementation of the Traveling Tournament Problem," Practice and Theory of Automated Timetabling (PATAT06, Brno, August 2006), Conference Proceedings, pp. 351-358.
4. Cormen, Thomas H. , Leiserson, Charles E., Rivest Ronald L., Stein, Clifford, Introduction to Algorithms._pp. 570-573.Boston: McGraw-Hill, 2001.
5. Di Gaspero L, Schaerf A. A composite-neighborhood tabu search approach to the traveling tournament problem. Journal of Heuristics 2007;13:pp. 189–207.
6. Easton K., Nemhauser G.L., Trick M.A. The travelling tournament problem: description and benchmarks. In: Walsh T, editor. Principles and practice of constraint programming. Lecture notes in computer science, vol. 2239. Berlin: Springer; 2001. pp. 580–5.
7. Kendall G. Scheduling English football fixtures over holiday periods. *Journal of the Operational Research Society 2008*; 59:743–55.
8. Kendall G., Knust S., Ribeiro C. C. and Urrutia S. Scheduling in Sports: An Annotated Bibliography. Computers & Operations Research (2009), 37: pp.1-19
9. Kingston, J. A tiling algorithm for high school timetabling, Proceedings Practice and Theory of Automated Timetabling (PATAT04, Pittsburgh, August 2004, USA).,Conference Proceedings, pp. 208-225.
10. Leong, G. (2003). Constraint programming for the traveling tournament problem. www.comp.nus.edu.sg/henz/students/gan_tiaw_leong.pdf
11. Lim, A., Zhang, X. (2003) Integer programming and simulated annealing for scheduling sports competition on multiple venues, Proceedings of the Fifth Metaheuristics International Conference (MIC 2003).
12. Ribeiro CC, Urrutia S. Heuristics for the mirrored traveling tournament problem. European Journal of Operational Research 2007;179:pp. 775–87.
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
14. Trick, M.A., Challenge Traveling tournament instances. Online document at http://mat.gsia.cmu.edu/TOURN/.
15. Trick MA., Formulations and Reformulations in Integer Programming. Lecture Notes In Computer Science; Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Volume 3524/2005 pp: 366-379. Springer 2005.
16. Van Hentenryck, P. and Vergados Y., "Population-Based Simulated Annealing for Traveling Tournaments", AAAI - National Conference on Artificial Intelligence, 2007