A Combined Local Search and Integer Programming Approach to the Traveling Tournament Problem

Marc Goerigk · Stephan Westphal

Received: date / Accepted: date

Abstract The *traveling tournament problem* is a well-known combinatorial optimization problem with application to sport leagues scheduling, that sparked intensive algorithmic research over the last decade. With the *Challenge Traveling Tournament Instances* as an established benchmark, the most successful approaches to the problem use meta-heuristics like tabu search or simulated annealing, partially heavily parallelized. Integer programming based methods on the other hand are hardly able to tackle larger benchmark instances.

In this work we present a hybrid approach that draws on the power of commercial integer programming solvers as well as the speed of local search heuristics. Our proposed method feeds the solution of one algorithm phase to the other one, until no further improvements can be made. The applicability of this method is demonstrated experimentally on the *galaxy* instance set, resulting in currently best known solutions for most of the considered instances.

Keywords traveling tournament problem · tabu search · integer programming · sports scheduling

1 Introduction

Before the start of each season, every sports league is faced with the problem of scheduling the games among their teams such that a variety of require-

Partially supported by grant SCHO 1140/3-2 within the DFG programme *Algorithm Engineering.*

M. Goerigk and S. Westphal
Institut für Numerische und Angewandte Mathematik
Universität Göttingen Lotzestr. 16-18
D-37083 Göttingen
Germany
Tel.: +49-551-3920035
Fax: ++49-551-393944
E-mail: s.westphal@math.uni-goettingen.de

ments is taken into account. The planners have to synchronize every feasible schedule to the availability restrictions of the sports sites, the most interesting games have to be matched to available TV slots and specific league-requested matchups have to be taken care of. Additionally, there is a wish to minimize the total distances driven by the teams over the season, which becomes even more important for bigger countries.

In this paper we will focus on the problem of minimizing the total distances driven by the teams in the way addressed by the well-known Traveling Tournament Problem (TTP). This sports scheduling problem which has been introduced by Easton et al. [ENT01] is inspired by Major League Baseball and is considered to be practically hard to solve.

1.1 Sports Scheduling and the Traveling Tournament Problem

Sports Scheduling in general deals with the design of tournaments. A *single round robin tournament* on n teams, where n is an even number, consists of $(n-1)$ rounds (also called *slots*). In each round $n/2$ games, which are themselves ordered pairs of teams, take place. Every team has to participate in one game per round and must meet every other team exactly once. It is standard to assume n to be even since in sports leagues with n being odd, usually a dummy team is introduced, and whoever plays it has a day off, which is called a *bye*. For scheduling single round robin tournaments a rather general and useful scheme called the *canonical schedule* has been known in sports scheduling literature for at least 30 years [dW81]. It is based on the polygon/circle method, which was first suggested by Kirkman in 1847 [Kir47]. One can think of Kirkman's method as a long table at which n players sit such that $n/2$ players on one side face the other players seated on the other side of the table. Every player plays a match against the person seated directly across the table. The next round of the schedule is obtained when everyone moves one chair to the right with the crucial exception that there exists one person at the end of the table who never moves and always maintains the seat from his or her first round. Note that this method only specifies who plays whom when and not where. The canonical schedule introduced by de Werra defines for each of the encounters specified by the method described above, at whose site they take place such that the number of successive home or away games is minimized [dW81].

A *double round robin tournament* on n teams consists of $2(n-1)$ rounds and every team must meet every other team twice: once at its own home venue (*home game*) and once at the other team's venue (*away game*). A popular policy in practice is to obtain a double round robin tournament from a single round robin tournament by *mirroring*, that is repeating the matches of round k for $k = 1, ..., n-1$ in round $k + n - 1$ with changed home field advantage. Consecutive home games are called a *home stand* and consecutive away games form a *road trip*. The *length* of a home stand or road trip is the number of opponents played (and not the distance traveled).

In this work we consider the *traveling tournament problem* (TTP) as described in [ENT01]:

Definition 1 (The Traveling Tournament Problem TTP(k) [ENT01]) Let a set of n teams and a distance matrix (d_{ij}) be given. Find a feasible double round robin tournament of the teams satisfying the following condition:

1. The length of any home stand is at most k.
2. The length of any road trip is at most k.
3. Game j at i is not followed immediately by game i at j.
4. The sum of the distances traveled by the teams is minimized.

As it is the case in most real-world applications, we henceforth assume $k = 3$ throughout the paper. The third requirement is known as *no-repeater constraint.*

An example instance from [Tri11] is given in Table 1: For every team, the distance to every other team is known. Table 2 shows the corresponding optimal solution with objective 416: On day 1, team 1 plays away against team 4, then away against team 2, at home against team 3 on day 3 and so on. Note that, as demanded, there is no home stand or road trip with length larger than 3.

	Team 1	Team 2	Team 3	Team 4
Team 1	0	10	15	34
Team 2	10	0	22	32
Team 3	15	22	0	47
Team 4	34	32	47	0

Table 1 Instance *galaxy4*.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6
Team 1	-4	-2	3	4	2	-3
Team 2	3	1	4	-3	-1	-4
Team 3	-2	-4	-1	2	4	1
Team 4	1	3	-2	-1	-3	2

Table 2 Optimal solution to *galaxy4*.

1.2 Previous Work

So far, most efforts concerning the TTP have led to a variety of algorithms aiming to minimize the total distance driven by the teams. Kendall et al.

[KKRU10] provide a good overview of the work done on the TTP and sports scheduling in general. Just to mention a very few examples, hybrid algorithms with constraint programming (CP) exist by Benoist et al. [BLR01] who additionally use Lagrange relaxation. Easton et al. [ENT01] merge CP with integer programming while Henz [Hen04] combines CP with large neighborhood search. Anagnostopoulos et al. [AMHV06], Hentenryck and Vergados [vHV06], Gaspero and Schaerf [GS07] and Lim et al. [LRZ06] propose neighborhood search-based algorithms, whereas Ribeiro and Urrutia [Rib11] focus on the special class of constant distance TTP where break maximization is equivalent to travel distance minimization.

On the theoretical side, Thielen and Westphal settled the complexity by showing that the TTP is strongly NP-hard [TW11]. Miyashiro et al. [MMI08] provide a $2 + (9/4)/(n-1)$ approximation for the intensively studied special case $k = 3$ by means of the *Modified Circle Method*, a variation of the canonical schedule. In [YIMM09] Yamaguchi et al. obtain an algorithm with approximation ratio $(2k-1)/k+O(k/n)$ for $k \leq 5$ and $(5k-7)/(2k)+O(k/n)$ for $k > 5$. Again they make use of the canonical schedule, now refined such that the teams are ordered around the 'table' such that most of the distances driven are part of a near optimal traveling salesman tour which clearly has positive effects on the length of many distances traveled. As $k \leq n-1$, they showed this way that a constant factor approximation for any choice of k and n exists. However, they did not show how this factor looks exactly. This was done later by Westphal and Noparlik [WN12], whose algorithm was also able to compute new bests for all galaxy instances with at least 22 teams.

1.3 Contribution

Due to its computational difficulty, exact solution approaches already fail from a size of 12 teams on [Tri11]. In this paper we propose a combination of local search heuristics with integer programming methods to overcome local optima. In an experimental evaluation, we were able to calculate new best known solutions on most instances of the considered benchmark set.

1.4 Overview

In Section 2, we describe the algorithmic details of our heuristic approach to the traveling tournament problem, which we evaluate extensively in Section 3. Section 4 concludes the paper and gives an outlook on further research.

2 A Combined Local Search and Integer Programming Heuristic

The heuristic we consider consists of two separate phases: The local search phase, which is described in Section 2.1, and the integer programming phase as described in Section 2.2. After explaining these phases, we give an overview of the whole algorithm in Section 2.3

The motivation for combining these methods is the following: Local search heuristics are an effective means to improve solutions even for large instances, but local minima pose complications. When the search is not able to leave such a minimum anymore, it helps to use a view that does not consider the same neighborhood as before. Therefore, breaking up the current solution structure by applying integer programming methods is able to provide the local search with a fresh start from an even improved solution. Within the VLNS classification framework [AEOP02], we are using restrictions on the original problem, but they are not likely solvable in polynomial time.

2.1 Phase 1: Tabu Search

The first part of the proposed algorithm consists of a local search phase. This might be steepest descent, simulated annealing, or any other suitable (meta-)heuristic. In this work we specifically considered a tabu search heuristic that uses the standard neighborhood as described in [AMHV06] and [DGS05]. The five neighborhood search moves are presented in Table 3.

Neighborhood	Input	Effect
Swap Homes	$t_1, t_2 \in T$	Swap home/away pattern for matches between t_1 and t_2. No further adjustments necessary.
Swap Teams	$t_1, t_2 \in T$	Swap all matches of teams t_1 and t_2. Adjust opponents accordingly.
Swap Days	$d_1, d_2 \in D$	Swap two days. No further adjustments necessary.
Swap Teams Partial	$t_1, t_2 \in T, d \in D$	Swap opponents of teams t_1 and t_2 on day d. This will cause more swaps to resolve resulting conflicts.
Swap Days Partial	$t \in T, d_1, d_2 \in D$	Swap the opponents of team t on days d_1 and d_2. This will cause more swaps between these days to reestablish feasibility.

Table 3 Local neighborhood for tabu search algorithm.

Furthermore, we the following algorithm specifications are used:

1. **Neighborhood:** In every iteration, the whole neighborhood is considered. That is, all moves of the types given in Table 3 are evaluated, and the best non-tabu move is chosen.

2. **Tabu List:** The list contains whole solutions, not just partial properties. The list length is dynamic in the sense that every considered solution becomes tabu, until a new global optimum is found, which triggers the list to be cleaned.
3. **Stopping criterion:** If a prescribed number *max idle iterations* of iterations have passed without finding a new global optimum, the search is resetted: The current global optimum is restored and the tabu list cleaned. After a given number *max restarts* of resets, the search is aborted.
4. **Objective:** In order to expand the search space to infeasible solutions, we add a penalty to the original problem objective for every violated home stand, road trip and no-repeater constraint (Constraints 1.-3. in the problem definition), but do not forbid schedules that violate these constraints.
5. **Dynamic Penalty:** The infeasibility penalty is dynamically adapted throughout the search process, such that sequences of feasible solutions decrease the penalty, and sequences of infeasible solutions increase it.
6. **Dominance:** Feasible solutions in the neighborhood that are better with respect to the original objective than the current best solution are always preferred to infeasible solutions, even if their modified objective might be better.

Though there are of course many more possibilities concerning the fine-tuning of the tabu search, this approach turned out to be most promising in preliminary experiments.

2.2 Phase 2: Integer Programming

In order to leave a local optimum of the local search procedure, we apply integer programming methods that do not depend on the neighborhood as presented in Table 3. For our experiments, we use a variation of the simple formulation presented in [Rib11] with $\mathcal{O}(n^3)$ variables. The variables $x_{ijk}(i, j = 1, \dots, n,\ k = 1, \dots, 2n-2)$ represent the decision if team i plays away against team j on day k, while $y_{tij}(i, j, t = 1, \dots, n)$ denotes that team t travels from team i to team j anywhere in the schedule.

$$\min \sum_{i,j=1}^{n} d_{ij} x_{ij1} + \sum_{t,i,j=1}^{n} d_{ij} y_{tij} + \sum_{i,j=1}^{n} d_{ji} x_{ij,2n-2} \tag{1}$$

$$x_{iik} = 0 \qquad (i = 1, \dots, n,\ k = 1, \dots, 2n-2) \tag{2}$$

$$\sum_{j=1}^{n} (x_{ijk} + x_{jik}) = 1 \qquad (i = 1, \dots, n,\ k = 1, \dots, 2n-2) \tag{3}$$

$$\sum_{k=1}^{2n-2} x_{ijk} = 1 \qquad (i, j = 1, \dots, n, i \neq j) \tag{4}$$

$$\sum_{l=0}^{3}\sum_{j=1}^{n} x_{ij,k+l} \leq 3 \qquad (i=1,\dots,n,\ k=1,\dots,2n-2-3) \tag{5}$$

$$\sum_{l=0}^{3}\sum_{i=1}^{n} x_{ij,k+l} \leq 3 \qquad (j=1,\dots,n,\ k=1,\dots,2n-2-3) \tag{6}$$

$$x_{ijk} + x_{jik} + x_{ij,k+1} + x_{ji,k+1} \leq 1 \qquad (i,j=1,\dots,n,\ k=1,\dots,2n-3) \tag{7}$$

$$z_{iik} = \sum_{j=1}^{n} x_{ijk}, \qquad (i=1,\dots,n\ k=1,\dots,2n-2) \tag{8}$$

$$z_{ijk} = x_{ijk} \qquad (i,j=1,\dots,n, i \neq j,\ k=1,\dots,2n-2) \tag{9}$$

$$y_{tij} \geq z_{tik} + z_{tj,k+1} - 1 \qquad (t,i,j=1,\dots,n,\ k=1,\dots,2n-3) \tag{10}$$

$$x_{ijk}, z_{ijk}, y_{tij} \in \{0,1\} \qquad (t,i,j=1,\dots,n,\ k=1,\dots,2n-2) \tag{11}$$

As there is little hope in solving this program directly, we divide it into two smaller problems based on the provided input solution: Optimizing the *home-away-pattern* (HA-opt) and optimizing the rest of the schedule with fixed home-away-pattern (non-HA-opt). Both subproblems are described in the following.

Optimize Home-Away-Pattern. In order to optimize the home-away-pattern for a given solution, we have to fix the decisions when two teams face another. Let $\tilde{x}$ be the given solution to the x variables. We now add the Constraint (12) to the original problem:

$$x_{ijk} + x_{jik} = \tilde{x}_{ijk} + \tilde{x}_{jik} \qquad (i,j=1,\dots,n,\ k=1,\dots,2n-2) \tag{12}$$

By doing so, we fix if team i plays against team j on day k, but leave the venue open. In terms of the solution format as described in Table 2, we restrict the optimization process to the signs $+, -$ of the schedule.

The resulting problem is also known as the *Timetable Constrained Distance Minimization Problem*, and has been introduced in [RT06].

Fix Home-Away-Pattern. The resulting second partial problems consists of finding optimal team matchups, when travel and home days are fixed for every team. As before, let $\tilde{x}$ be the given solution. We add Constraints (13) and (14) to the problem formulation:

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} \tilde{x}_{ijk}, \qquad (j=1,\dots,n,\ k=1,\dots,2n-2) \tag{13}$$

$$\sum_{j=1}^{n} x_{ijk} = \sum_{j=1}^{n} \tilde{x}_{ijk}, \qquad (i=1,\dots,n,\ k=1,\dots,2n-2) \tag{14}$$

These constraints force a team to play away if this is the case for the input solution, and to play at home otherwise. For the solution format of Table 2, this means that we fix the signs, but can change the actual opponents.

2.3 Phase Combination

Having described the two phases separately, we now focus on how to combine them. In Figure 1, a diagram of the proposed algorithm structure is given.

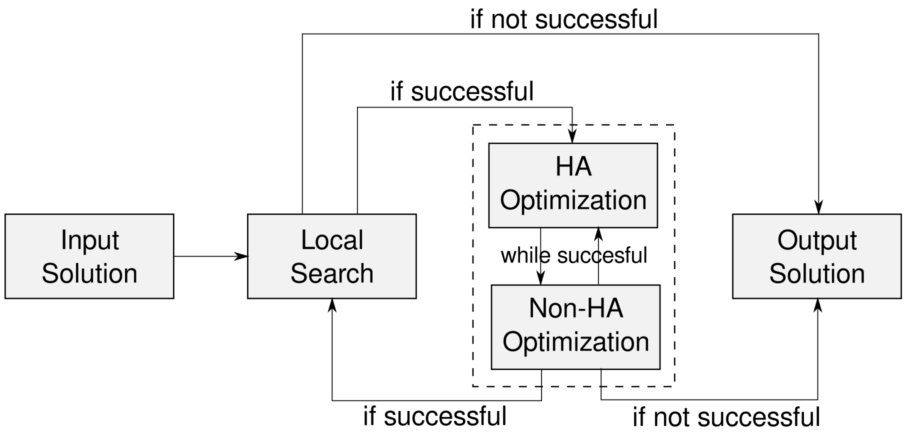

Fig. 1 Algorithm overview.

The solution process needs to be provided with a starting solution, whose creation is described in the experimental setup of Section 3. First, a local search is performed until its stopping criterion is satisfied. If it was possible to improve the solution, we pass it to the second phase, and end the solution process otherwise. In the second phase, we repeatedly optimize the home-away-pattern and its counterpart problem, until both partial problems are not able to improve the current solution anymore. If the phase was able to improve the given solution at all, we repeat phase 1, and end the algorithm otherwise.

Additionally, we propose another feature that turned out to be valuable in our experiments. The local search phase considers many solutions that may be inferior to the current best solution, but are structurally so different that a home-away optimization might create a new current best solution. Therefore, even if the local search was not able to improve its input, we feed the last found local optimum to the second phase. Only if this does not create a solution that is better than the current best, we consider the local search phase as being not successful.

3 Experimental Results

In this section we present experimental experience on the performance of the proposed combination of local search and integer programming techniques.

Environment All experiments were conducted on a PC with 99 GB main memory and an Intel Xeon X5650 processor, running with 6 cores at 2.66 GHz and

12MB cache. All code is written in C++ and has been compiled with g++ 4.4.3 and optimization flag -O3. For the integer programming phase, we used the Gurobi optimizer [Inc11] in version 4.5.

Setup The conducted experiment was scheduled the following way:

1. The considered benchmark set are the *galaxy* instances from [Tri11]. As instances with team size up to 10 have already been solved to optimality, we only used those which are larger.
2. We construct canonical schedules [dW81] as input solutions using the rotation scheme as described in [WN12]. Of these, only the best three are used per instance.
3. The described algorithm is run for the resulting 45 solutions. The parameter set we used is presented in Table 4. Note that the factor we multiply the infeasibility penalty of the local search with is chosen in a way that penalty increase faster than they decrease.
4. For the best solution found so far for every problem instance, we restart the algorithm with the second parameter set of Table 4.

Teams	12-14	16-22	24-36	38-40
max idle iterations, first	10,000	1,000	500	250
max restarts, first	2			
max idle iterations, second	100,000	4,000	1500	500
max restarts, second	3			
starting inf. penalty	input objective $/1,000$			
inf. penalty factor	0.97 and 1/0.93			
timelimit HA-opt	$1800s$			
timelimit Non-HA-opt	$3600s$			

Table 4 Parameter choice.

Results We summarize the achieved results in Table 5. In the first column, the instance size in terms of number of teams is given, followed by the objective value of the best currently known solution as of January 2012 in column two. We then present the initial objective values of the three best initial solutions, and the improved objective value after the first run of the algorithm, together with the corresponding number of algorithm phases. As described in the *setup* paragraph, these solutions are further improved by restarting the algorithm.

In the last column, we present a lower bound for each instance, and mark bounds that were previously unknown with an asterisk (*). These bounds are found by calculating an optimal tour for each team separately, and summing up the respective tour lengths. Finding these tours was done using a flow-based integer programming formulation with a timelimit of 3600 seconds, which was hit in only a few cases.

As can be seen, it was possible to further improve the currently best solution in 9 out of 15 cases by at least 0.1%, and up to 3.2%. There seems to be

Teams	Best Known	Initial Solutions	1st Improvement	Number of phases	2nd Improvement	LB
		8223	7642	2		
12	7197	8364	7720	2	7555 (5.0%)	6933
		8408	7639	2		
		13017	11566	2		
14	10918	13047	12008	2	11552 (5.8%)	10221
		13126	11636	2		
		16257	15769	2		
16	14900	16315	15897	2	15704 (5.4%)	13619*
		16372	16094	2		
		21635	21426	2		
18	20907	21658	21437	2	21346 (2.1%)	19050*
		21728	21346	2		
		28237	26921	4		
20	26289	28332	27121	3	26749 (1.7%)	23738*
		28368	26749	4		
		35832	35624	4		
22	35516	35882	35812	2	35584 (0.2%)	31461*
		36021	35584	2		
		45962	45671	3		
24	45728	46029	45657	2	**45657** (-0.2%)	41287*
		46130	45705	2		
		61617	58991	4		
26	60962	61634	59889	4	**58991** (-3.2%)	53802*
		61703	59894	4		
		77683	77381	2		
28	77577	77732	77320	3	**77320** (-0.3%)	69992*
		77736	77361	3		
		97270	96756	2		
30	96765	97321	96712	2	**96710** (-0.1%)	88831*
		97384	96710	4		
		122567	120053	3		
32	120683	122655	120130	4	**119996** (-0.6%)	108187*
		122661	119996	4		
		148194	147644	3		
34	147742	148223	147612	4	**147612** (-0.1%)	133976*
		148363	147763	3		
		174475	173532	3		
36	173640	174595	173716	3	**173532** (-0.1%)	158363*
		174734	173670	2		
		212706	205876	3		
38	209463	212809	204980	8	**204980** (-2.1%)	188935*
		213072	205870	7		
		249976	247017	9		
40	249002	249996	248295	3	**247017** (-0.8%)	226794*
		250081	248223	6		

Table 5 Results overview.

a connection between the instance size and the number of algorithm phases, which can be explained by the increasingly larger neighborhood, which makes it more difficult for the local search to find an actually close, better solution. Therefore, combining local search with integer programming methods is especially beneficial for large instances.

Of course, many more experimental setups are possible to pursue: As an example, we not only examined the best three initial solutions of galaxy22, but all 22 of them. As a result a solution of objective 35467 was found, which is 0.1% below the current best solution.

4 Conclusion and Outlook

We proposed an algorithm to the traveling tournament problem that is able to overcome local optima arising in local search heuristics by solving integer optimization subproblems. Its applicability is demonstrated by an extensive experiment on a well-known benchmark set, resulting in new best known solutions for all instances of size greater or equal to 24. Experiments with further instance sets are currently being conducted.

It seems promising to use further parameter setups and local search heuristics to find better solutions than presented in this work. Also, we plan to use the described algorithmic ideas to include *robustness* issues in the schedule design, i.e., to find tournament schedules that are insensitive to disruptions like bad weather conditions that may increase travel time between two venues, or even render a stadium unusable for certain days.

References

[AMHV06] Aris Anagnostopoulos, Laurent Michel, Pascal Van Hentenryck, and Yannis Vergados. A simulated annealing approach to the traveling tournament problem. *J. Scheduling*, 9(2):177–193, 2006.

[AEOP02] Ravindra K. Ahuja, zlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(13):75 – 102, 2002.

[BLR01] T. Benoist, L. Laburthe, and B. Rottembourg. Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems. In *Proceedings of the 3rd International Workshop on the Integration of AI and OR Techniques (CP-AI-OR)*, pages 15–26, 2001.

[DGS05] Luca Di Gaspero and Andrea Schaerf. A tabu search approach to the traveling tournament problem. In *Proceedings of the 6th Metaheuristics International Conference (MIC-2005)*, Vienna, Austria, August 2005. Available as electronic proceedings.

[dW81] D. de Werra. Scheduling in Sports. In P. Hansen, editor, *Studies on graphs and integer programming*, volume 11, pages 381–395. Annals of Discrete Mathematics, North Holland, 1981.

[ENT01] K. Easton, G. Nemhauser, and M. Trick. The traveling tournament problem description and benchmarks. *Principles and Practice of Constraint ProgrammingCP 2001*, pages 580–584, 2001.

[GS07] Luca Di Gaspero and Andrea Schaerf. A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, 13:189–207, April 2007.

[Hen04] M. Henz. Playing with constraint programming and large neighborhood search for traveling tournaments. In *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT)*, pages 23–32, 2004.

[Inc11] Gurobi Optimization Inc. Gurobi optimizer, 2011. Version 4.5.

[Kir47] Thomas P. Kirkman. On a problem in combinations. *The Cambridge and Dublin Mathematical Journal*, 2:191–204, 1847.

[KKRU10] Graham Kendall, Sigrid Knust, Celso C. Ribeiro, and Sebastián Urrutia. Invited review: Scheduling in sports: An annotated bibliography. *Comput. Oper. Res.*, 37:1–19, January 2010.

[LRZ06] A. Lim, B. Rodrigues, and X. Zhang. A simulated annealing and hill- climbing algorithm for the traveling tournament problem. *European Journal of Operations Research*, 174:1459 – 1478, 2006.

[MMI08] R. Miyashiro, T. Matsui, and S. Imahori. An approximation algorithm for the traveling tournament problem. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT)*, 2008.

[Rib11] Celso C. Ribeiro. Sports scheduling: Problems and applications. *International Transactions in Operational Research*, 2011.

[RT06] Rasmus Rasmussen and Michael Trick. The timetable constrained distance minimization problem. In J. Beck and Barbara Smith, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3990 of *Lecture Notes in Computer Science*, pages 167–181. Springer Berlin / Heidelberg, 2006.

[Tri11] Michael Trick. Challenge traveling tournament problems benchmark, 2011. http://mat.gsia.cmu.edu/TOURN/.

[TW11] C. Thielen and S. Westphal. Complexity of the traveling tournament problem. *Theoretical Computer Science*, 412(4-5):345–351, 2011.

[vHV06] P. van Hentenryck and Y. Vergados. Traveling tournament scheduling: A systematic evaluation of simulated annealing. *LNCS*, 3990:228–243, 2006.

[WN12] S. Westphal and K. Noparlik. A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, 2012.

[YIMM09] D. Yamaguchi, S. Imahori, R. Miyashiro, and T. Matsui. An improved approximation algorithm for the traveling tournament problem. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC)*, volume 5878 of *LNCS*, pages 679–688, 2009.