Ann Oper Res (2016) 239:343–354
DOI 10.1007/s10479-014-1586-6

A combined local search and integer programming approach to the traveling tournament problem

Marc Goerigk · **Stephan Westphal**

Published online: 6 April 2014

Abstract The *traveling tournament problem* is a well-known combinatorial optimization problem with direct applications 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 requirements 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

M. Goerigk (✉)
Fachbereich Mathematik, Technische Universität Kaiserslautern, Paul-Ehrlich Straße 14,
D-67653 Kaiserslautern, Germany
e-mail: goerigk@mathematik.uni-kl.de

S. Westphal
Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestr. 16-18,
D-37083 Göttingen, Germany
e-mail: s.westphal@math.uni-goettingen.de

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. (2001) 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 Werra (1981). It is based on the polygon/circle method, which was first suggested by Kirkman in 1847 Kirkman (1847). 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 Werra (1981).

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 $k = 1, \dots, n-1$ 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 TTP as described in Easton et al. (2001):

Definition 1 *(The Traveling Tournament Problem TTP(k)* Easton et al. (2001)) 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. The two games between any two teams do not follow immediately.
4. The sum of the distances traveled by the teams is minimized.

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

An example instance from Trick (2011), called *galaxy4*, is given in Table 1: For every team, the distance to every other team is known. Table 2 shows the corresponding optimal

Table 1 Instance *galaxy4*

	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 2 Optimal solution to *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

solution with objective 416, where a positive enty denotes a home game, and a negative enty denotes an away game. 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.

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. (2010) 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. (2001) who additionally use Lagrange relaxation. Easton et al. (2001, 2003) merge CP with integer programming and column generation techniques. They also developed the independent lower bound in that paper, which was later improved by Urrutia et al. (2007). Irnich used a new IP-formulation exploiting a network structure of the problem with a branch-and-price approach Irnich (2010). Melo et al. (2009) examine the quality of different IP-formulations for the case of predefined venues.

Many successful approaches use meta-heuristics to generate solutions. For example, Henz (2004) combines CP with large neighborhood search. Anagnostopoulos et al. (2006), Hentenryck and Vergados (2006), Gaspero and Schaerf (2007) and Lim et al. (2006) propose neighborhood search-based algorithms, whereas Ribeiro (2011) focus on the special class of constant distance TTP where break maximization is equivalent to travel distance minimization. For the special case of mirrored schedules, Ribeiro and Urrutia (2007) deployed a hybrid heuristic based on GRASP and ILS.

On the theoretical side, Thielen and Westphal settled the complexity by showing that the TTP is strongly NP-hard Thielen and Westphal (2011). Miyashiro et al. (2008) 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 Yamaguchi et al. (2009) 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 (2012), whose algorithm was also able to compute new bests for all galaxy instances with at least 22 teams.

1.3 Contributions and overview

Due to its computational difficulty, exact solution approaches already fail from a size of 12 teams on Trick (2011). 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.

In Sect. 2, we describe the algorithmic details of our heuristic approach to the traveling tournament problem, which we evaluate extensively in Sect. 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 Sect. 2.1, and the integer programming phase as described in Sect. 2.2. After explaining these phases, we give an overview of the whole algorithm in Sect. 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.

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 Anagnostopoulos et al. (2006) and Di Gaspero and Schaerf (2005). Given a feasible starting solution, we iteratively consider the five neighborhood search moves as presented in Table 3.

Note that if we simply chose a move from this neighborhood that yields the smallest objective value, we would have a steepest descent method. Following the tabu search approach (see, e.g. Glover et al. (1993)), we improve such a heuristic by maintaining a list of already visited solutions that may not be visited again (the tabu list). Using such a list, the local search is able to leave local optima. Specifically, the following algorithmic features were applied:

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” (see point 4) 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

Table 3 Local neighborhood for tabu search algorithm

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

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). This penalty is dynamically adapted throughout the search process, such that sequences of feasible solutions decrease the penalty, and sequences of infeasible solutions increase it.
5. **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. For specific parameter choices, we refer to the experimental setup as presented in Sect. 3.

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 Ribeiro (2011) 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. Finally, z_{ijk} $(i, j = 1, \dots, n,\ k = 1, \dots, 2n-2)$ are auxiliary variables that are used to model the driving distances of the teams.

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

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

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

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

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

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

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

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

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

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

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

We now explain this model in more detail. The objective function (1) consists of three terms: The first one models the driving distances for the first day, the last one models the distances for the last day, and the middle term models the distances for all other days. Using (2), we ensure that a team never plays against itself. Constraints (3) forces each team to play exactly one match per day (either home or away), while (4) model that every team plays each other team once away and once at home over the $2n - 2$ days. Constraints (5) and (6) ensure that home stands and road trips have a length of at most 3, while Constraints (7) encapsule the no-repeater constraint. Finally, Constraints (8–10) model the driving behavior of the teams.

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. Note that the variables y and z are already completely determined if x is given.

2.2.1 *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 a 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. This means that the solver is provided with the given solution as a warm start, and by adding Constraints (12), only the home-away-pattern can be changed. Apart from this, the solver may compute a completely different solution.

2.2.2 *Fix home-away-pattern.*

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

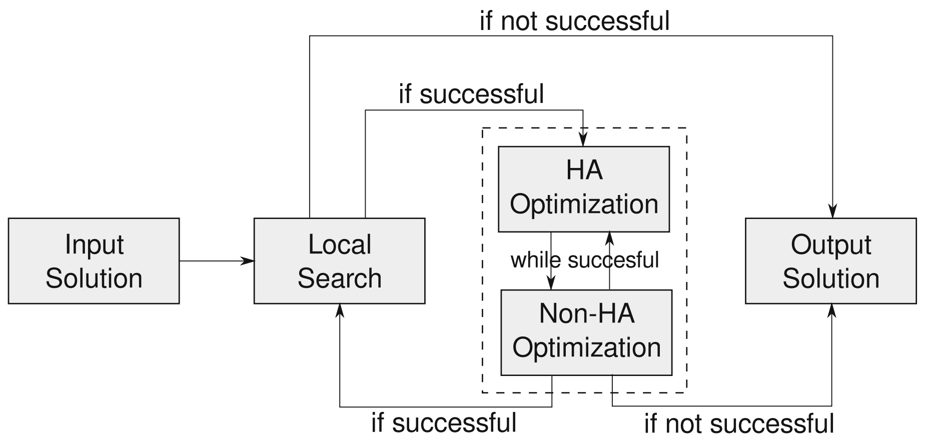

Fig. 1 Algorithm overview

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

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

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. As before, the solver is presented with $\tilde{x}$ as a warm start solution, and is free to modify it within the limitations of Constraints (13) and (14).

2.3 Phase combination

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

The solution process needs to be provided with a feasible starting solution, e.g., one of the canonical schedules (see also the experimental setup of Sect. 3). Then, the tabu 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. For the second phase, we use the best feasible solution generated in the previous phase, and repeatedly optimize the home-away-pattern and its counterpart problem as described in the previous subsection, 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.

Table 4 Parameter choice

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	1,000,000	50,000	30,000	10,000
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			

3 Experimental results

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

3.1 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 (2011) in version 4.5.

3.2 Setup

The conducted experiment was scheduled the following way:

1. The considered benchmark set are the *galaxy* instances from Trick (2011). 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 Werra (1981) as input solutions using the rotation scheme as described in Westphal and Noparlik (2012). Of these, only the best three are used per instance.
3. The described algorithm is run for the resulting 45 solutions. Using preliminary experiments to probe the performance of the algorithm for different parameter sets, we decided to use the one 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. We then intensify the search around the best solutions found so far for every problem instance. To this end, we restart the algorithm with the second parameter set of Table 4.

3.3 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

Table 5 Results overview

Teams	Best known	Initial solutions	First improvement	Number of phases	Second improvement	LB
12	7, 197	8,223	7,642	2	**7,555** (5.0 %)	6,933
		8,364	7,720	2		
		8,408	**7,639**	2		
14	10, 918	13,017	**11,566**	2	**11,552** (5.8 %)	10,221
		13,047	12,008	2		
		13,126	11,636	2		
16	14, 900	16,257	**15,769**	2	**15,704** (5.4 %)	13,619
		16,315	15,897	2		
		16,372	16,094	2		
18	20, 907	21,635	21,426	2	**21,346** (2.1 %)	19,050
		21,658	21,437	2		
		21,728	**21,346**	2		
20	26, 289	28,237	26,921	4	**26,699** (1.6 %)	23,738
		28,332	27,121	3		
		28,368	**26,749**	4		
22	35, 516	35,832	35,624	4	**35,584** (0.2 %)	31,461
		35,882	35,812	2		
		36,021	**35,584**	2		
24	45, 728	45,962	45,671	3	**45,657** (−0.2 %)	41,287
		46,029	**45,657**	2		
		46,130	45,705	2		
26	60, 962	61,617	**58,991**	4	**58,934** (−3.3 %)	53,802
		61,634	59,889	4		
		61,703	59,894	4		
28	77, 577	77,683	77,381	2	**77,320** (−0.3 %)	69,992
		77,732	**77,320**	3		
		77,736	77,361	3		
30	96, 765	97,270	96,756	2	**96,710** (−0.1 %)	88,831
		97,321	96,712	2		
		97,384	**96,710**	4		
32	120, 683	122,567	120,053	3	**119,996** (−0.6 %)	108,374
		122,655	12,0130	4		
		122,661	**119,996**	4		
34	147, 742	148,194	147,644	3	**147,612** (−0.1 %)	133,976
		148,223	**147,612**	4		
		148,363	147,763	3		
36	173,640	174,475	**173, 532**	3	**173,532** (−0.1 %)	158,549
		174,595	173,716	3		
		174,734	173,670	2		

Table 5 continued

Teams	Best known	Initial solutions	First improvement	Number of phases	Second improvement	LB
38	209,463	212,706	205,876	3	**204,497** (−2.4 %)	189,126
		212,809	**204,980**	8		
		213,072	205,870	7		
40	249,002	249,976	**247,017**	9	**247,017** (−0.8 %)	226,820
		249,996	248,295	3		
		250,081	24,8223	6		

Table 6 Results for NL instances

Teams	Best known	Our solution	LB
12	110,729	114,355	108,629
14	188,728	198,514	183,354
16	261,687	280,819	249,477

solution as of January 2012 in column two.[1] 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 the so-called independent lower bound for each instance. These bounds are found by calculating an optimal tour for each team separately, and summing up the respective tour lengths.

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 3.3 %. There seems to be 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 tor the local search to find an actually close, better solution. Therefore, combining local search with integer programming methods can be especially beneficial for large instances.

Typically, the integer programs for HA-opt and non-HA-opt were not solved to optimality within the given time limit. The only exception was HA-opt for galaxy12 and galaxy14; for all other instances, we only used the best solution found by the IP solver for the next iteration. Furthermore, it turns out in this experiment that non-HA-opt was not able to improve the current solution even once. All improvements in the integer programming phase where gained by using HA-opt. However, as this was not the case with preliminary experiments, we included both steps in the setup.

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.

Finally, this experimental setup can also directly be applied to other sets of the benchmark Trick (2011). For a comparison, we considered the instances NL12, NL14, and NL16 with the same experimental setup as before. Due to the relatively small instance sizes, we were not able to improve the currently best known solutions, see Table 6.

[1] After completing the presented experiments, some results were further improved; partially by applying the presented algorithm to a differently generated set of starting solutions.

4 Conclusion and outlook

We proposed an algorithm to the TTP 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 10 out of 15 instances.

It seems promising to use not only further parameter setups and local search heuristics to systematically find better solutions than presented in this work, but also to use different sets of starting solutions, which is currently under research. 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

Anagnostopoulos, A., Michel, L., Van Hentenryck, P., & Vergados, Y. (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, *9*(2), 177–193.

Benoist, T., Laburthe, L., & Rottembourg, B. (2001). 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)*. (pp. 15–26).

Di Gaspero, L., & Schaerf, A. (2005). A tabu search approach to the traveling tournament problem. In *Proceedings of the 6th Metaheuristics International Conference (MIC-2005), Austria*. Vienna, Available as electronic proceedings.

Di Gaspero, L., & Schaerf, A. (2007). A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, *13*, 189–207.

de Werra, D. (1981). Scheduling in Sports. In P. Hansen (Ed.), *Studies on graphs and integer programming* (Vol. 11, pp. 381–395). North Holland: Annals of Discrete Mathematics.

Easton, K., Nemhauser, G., & Trick, M. (2001). The traveling tournament problem description and benchmarks. *Principles and Practice of Constraint Programming CP*, *2001*, 580–584.

Easton, K., Nemhauser, G., & Trick, M. (2003). Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In E. Burke & P. Causmaecker (Eds.), *Practice and Theory of Automated Timetabling IV* (Vol. 2740, pp. 100–109)., Lecture Notes in Computer Science Springer: Berlin Heidelberg.

Glover, F., Taillard, E., & Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, *41*(1), 1–28.

Henz, M. (2004). 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)*. (pp. 23–32).

Gurobi Optimization Inc.,. (2011). Gurobi optimizer. Version 4.5.

Irnich, S. (2010). A new branch-and-price algorithm for the traveling tournament problem. *European Journal of Operational Research*, *204*(2), 218–228.

Kirkman, T. P. (1847). On a problem in combinations. *The Cambridge and Dublin Mathematical Journal*, *2*, 191–204.

Kendall, G., Knust, S., Ribeiro, C. C., & Urrutia, S. (2010). Invited review: Scheduling in sports: An annotated bibliography. *Computers and Operations Research*, *37*, 1–19.

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

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

Melo, R. A., Urrutia, S., & Ribeiro, C. C. (2009). The traveling tournament problem with predefined venues. *Journal of Scheduling*, *12*, 607–622.

Ribeiro, C. C. (2011). Sports scheduling: Problems and applications. *International Transactions in Operational Research*, *19*, 201–226.

Ribeiro, C. C., & Urrutia, S. (2007). Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, *179*(3), 775–787.

Trick, M. (2011). Challenge traveling tournament problems benchmark. http://mat.gsia.cmu.edu/TOURN/. Accessed Jan 2012.

Thielen, C., & Westphal, S. (2011). Complexity of the traveling tournament problem. *Theoretical Computer Science*, *412*(4–5), 345–351.

Urrutia, S., Ribeiro, C.C., Melo, R.A. (2007). A new lower bound to the traveling tournament problem. In Computational Intelligence in Scheduling, 2007. SCIS '07. IEEE Symposium on, (pp. 15–18).

van Hentenryck, P., & Vergados, Y. (2006). Traveling tournament scheduling: A systematic evaluation of simulated annealing. *LNCS*, *3990*, 228–243.

Westphal, S., & Noparlik, K. (2012). A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*. doi:10.1007/s10479-012-1061-1.

Yamaguchi, D., Imahori, S., Miyashiro, R., & Matsui, T. (2009). 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, (pp. 679–688).