An ILS heuristic for the traveling tournament problem with predefined venues

Fabrício N. Costa · **Sebastián Urrutia** ·
Celso C. Ribeiro

Received: February 29, 2008

Abstract The Traveling Tournament Problem with Predefined Venues (TTPPV) is a single round robin variant of the Traveling Tournament Problem, in which the venue of each game to be played is known beforehand. We propose an Iterated Local Search (ILS) heuristic for solving real-size instances of the TTPPV, based on two types of local search moves and two types of perturbations. Initial solutions are derived from canonical 1-factorizations of the tournament graph or of its subgraphs. Computational results show that the new ILS heuristic performs much better than heuristics based on integer programming and improves the best known solutions for benchmark instances.

Keywords Traveling tournament problem · Sports scheduling · Iterated local search · Metaheuristics

1 Introduction and motivation

A round robin tournament is one in which each team plays against every other a fixed number of times in a given number of rounds. A team faces every other team exactly once (resp. twice) in a single round robin (SRR) tournament (resp. in a double round robin (DRR) tournament). A tournament is compact if the number of rounds is minimum and every team plays a game in every round. Every game is played in the venue of one of the opponent teams. Scheduling an SRR tournament consists in determining in which round and in which venue each game will be played.

The problem of scheduling a round robin tournament may be divided into two stages. The construction of the timetable determines the round in which each game is played. The home-away assignment (HAA) determines in which venue each game is played. Together,

F. N. Costa · S. Urrutia
Department of Computer Science, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, Belo Horizonte, MG 31270-010, Brazil
E-mail: {fabrimac,surrutia}@dcc.ufmg.br

C. C. Ribeiro
Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria 156, Niterói, RJ 24210-240, Brazil
E-mail: celso@ic.uff.br

the timetable and the home-away assignment determine the tournament schedule. A review of the literature may be found in Rasmussen and Trick (2008).

The Traveling Tournament Problem (TTP) introduced by Easton et al (2001) may be described as follows. A double round robin tournament is played by an even number n of teams indexed by $1,\ldots,n$. Each team has its own venue at its home city. All teams are initially at their home cities, to where they return after their last away game. The distance $d_{ij} \geq 0$ from the home city of team i to that of team j is known beforehand. Whenever a team plays two consecutive away games, it travels directly from the venue of the first opponent to that of the second. The problem calls for a DRR tournament schedule such that no team plays more than three consecutive home games or more than three consecutive away games, there are no consecutive games involving the same pair of teams, and the total distance traveled by the teams during the tournament is minimized.

Melo et al (2007) introduced the Traveling Tournament Problem with Predefined Venues (TTPPV). This variant of the TTP considers a single round robin tournament, in which the venues where the games take place are known beforehand. The set of games to be played is represented by ordered pairs of teams determined by the HAA. The game between teams i and j is represented either by (i,j) or by (j,i). In the first case, the game between i and j takes place at the venue of team i; otherwise, at that of team j. Therefore, for every two teams i and j, either the pair (i,j) or the pair (j,i) belongs the set of games to be played. The TTPPV consists in finding a compact single round robin schedule compatible with the HAA, such that the total distance traveled by the teams is minimized and no team plays more than three consecutive home games or three consecutive away games. This problem is a natural extension of the Traveling Tournament Problem to the case of single round robin tournaments.

Variants of this problem find interesting applications in real-life leagues whose DRR tournaments are divided into two SRR phases. Games in the second phase are exactly the same as those in the first phase, except for the inversion of their venues. Therefore, the venues of the games in the second phase are known beforehand and constrained by those of the games in the first phase. This is the case e.g. of the Chilean soccer professional league (see Durán et al (2007)) and of the German table tennis federation of Lower Saxony (see Knust (2007)).

Instances of the TTPPV with up to eight teams were solved to optimality by the integer programming formulations presented in Melo et al (2008). Since feasible solutions have not been found by a commercial solver within two hours of running time for instances with 18 or more teams, four heuristics based on the integer programming formulations were also developed. In this paper, we propose a local search based heuristic for the TTPPV to find good quality solutions for realistic size instances. Section 2 describes the proposed heuristic. Section 3 reports the computational experiments. Concluding remarks are made in the last section.

2 Heuristic approach

A factor of a graph $G=(V,E)$ is a subgraph $G'=(V,E')$ of G, with $E' \subseteq E$. G' is said to be a 1-factor of G if all its nodes have degree equal to one. A factorization of G is a set of edge-disjoint factors $\{G^1=(V,E^1),\ldots,G^p=(V,E^p)\}$, with $E^1 \cup \ldots \cup E^p = E$. A 1-factorization of G is one in which all factors are 1-factors. In an ordered 1-factorization of G, the 1-factors are taken in a fixed order.

Schedules of an SRR tournament with n (even) teams and fixed home away assignments may be represented by ordered 1-factorizations of the complete undirected graph K_n (see Ribeiro and Urrutia (2007)). Each node of this graph represents a team. An edge from node i to node j in the k-th 1-factor of an ordered 1-factorization implies that the game between teams i and j is played in the k-th round in the venue determined by the HAA.

2.1 Initial solutions

Two distinct methods are use to build initial 1-factorizations in this paper. In the canonical 1-factorization (see de Werra (1981)), the edge set of each factor $G^i = (V, E^i)$, for $i = 1, \ldots, n-1$, is defined as follows, with $V = \{1, \ldots, n\}$:

$$E^i = \{(n,i)\} \cup \{(f_1(i,k), f_2(i,k)) : k = 1, ..., n/2-1\},$$

with

$$f_1(i,k) = \begin{cases} i+k, & \text{if } i+k < n, \\ i+k-n+1, & \text{if } i+k \geq n, \end{cases}$$

and

$$f_2(i,k) = \begin{cases} i-k, & \text{if } i-k > 0, \\ i-k+n-1, & \text{if } i-k \leq 0. \end{cases}$$

A variation of the canonical 1-factorization (see de Werra (1980)) is used in this paper whenever n is divisible by four. Nodes of V are separated into two sets $A = \{a_1, ..., a_{n/2}\}$ and $B = \{b_1, ..., b_{n/2}\}$ with $n/2$ nodes each. A canonical 1-factorization is built for the complete graphs defined by each set A and B. To build the first $n/2-1$ factors of the 1-factorization of K_n, make the union of any factor of the 1-factorization associated with A with any factor of the 1-factorization associated with B. Proceed as before, using pairs of unused factors of the 1-factorizations associated with A and B until $n/2-1$ factors of the 1-factorization of K_n are obtained.

The remaining $n/2$ factors of the 1-factorization of G (corresponding to the last $n/2$ rounds of the tournament schedule) contain exclusively edges with one extremity in A and the other in B. The edge set of each 1-factor G^r is defined as follows, for each $r = n/2, \ldots, n-1$:

$$E^r = \{(a_i, b_{f_3(i,r)}) : i = 1, ..., n/2\},$$

with

$$f_3(i,r) = \begin{cases} i+r-n/2, & \text{if } i+r \leq n; \\ i+r-n, & \text{otherwise.} \end{cases}$$

Teams are randomly associated with nodes of the complete graph K_n and the 1-factors representing the rounds are ordered arbitrarily in both factorization schemes. We notice that both construction procedures may build ordered 1-factorizations that violate the limits of the maximum number of consecutive home (or away) games.

2.2 Neighborhoods

Let $V = \{1, \ldots, n\}$ be the set of teams and $R = \{1, \ldots, n-1\}$ the set of rounds of a schedule X. We assume that this schedule is represented by a matrix with n rows and $n-1$ columns, where $X(t,r)$ denotes the opponent of team $t \in V$ in round $r \in R$. A negative sign indicates that team t is playing an away game in round r. Figure 1 shows a schedule for a tournament with eight teams.

Teams	Rounds						
	1	2	3	4	5	6	7
1	-4	5	6	-2	8	7	-3
2	-6	-8	4	1	3	-5	-7
3	5	-4	-7	-6	-2	8	1
4	1	3	-2	-8	7	-6	-5
5	-3	-1	-8	-7	6	2	4
6	2	-7	-1	3	-5	4	8
7	-8	6	3	5	-4	-1	2
8	7	2	5	4	-1	-3	-6

Fig. 1 Schedule for a tournament with eight teams.

Different neighborhood structures have been used in local search procedures for scheduling round robin tournaments, see e.g. Anagnostopoulos et al (2006); Gaspero and Schaerf (2007); Ribeiro and Urrutia (2007). The basic home-away swap neighborhood used by Ribeiro and Urrutia (2007) is not considered in this study, since the home-away assignments are fixed beforehand. We consider four neighborhoods in the context of the TTPPV.

The first neighborhood is *team swap* (N_1). For any two teams $t_1 \in V$ and $t_2 \in V$, with $t_1 \neq t_2$, the schedule obtained by swapping the opponents of teams t_1 and t_2 in all rounds of the schedule X is a neighbor of the latter in N_1.

Similarly, the second neighborhood is *round swap* (N_2). For any two rounds $r_1 \in R$ and $r_2 \in R$, with $r_1 \neq r_2$, the schedule obtained by swapping the games of schedule X in rounds r_1 and r_2 is a neighbor of X in N_2.

The third neighborhood is *partial team swap* (N_3). For any round $r \in R$ and for any two teams $t_1 \in V$ and $t_2 \in V$, with $t_1 \neq t_2$ and $|X(t_1,r)| \neq t_2$, let S be a minimum cardinality subset of rounds including round r in which the opponents of teams t_1 and t_2 are the same, i.e. $S = \{r_1, \ldots, r_k\} \subseteq R$ is minimal and such that $r \in S$ and $\{t \in V : \exists j \in S$ such that $|X(t,j)| = t_1\} = \{t \in V : \exists j \in S$ such that $|X(t,j)| = t_2\}$. Given a schedule X, a round r, and teams t_1 and t_2 defined as above, the schedule obtained by swapping the opponents of teams t_1 and t_2 in all rounds in S is a neighbor of X in N_3.

Figure 2 illustrates a move in neighborhood N_3 for a tournament with eight teams and $r = 2$, $t_1 = 1$, and $t_2 = 2$. In this case, $S = \{2,5,6,7\}$. Teams 3, 5, 7, and 8 are the opponents of teams 1 and 2 in the rounds in S. We notice that the partial team swap neighborhood N_3 is a generalization of the team swap neighborhood N_1.

The last neighborhood is *partial round swap* (N_4). For any team $t \in V$ and for any two rounds $r_1 \in R$ and $r_2 \in R$, with $r_1 \neq r_2$, let U be a minimum cardinality subset of teams including team t in which the opponents of the teams in U in rounds r_1 and r_2 are the same, i.e. $U = \{t_1, \ldots, t_k\} \subseteq V$ is minimal and such that $t \in U$ and $\{i \in V : \exists u \in U$ such that $|X(i,r_1)| = u\} = \{i \in V : \exists u \in U$ such that $|X(i,r_2)| = u\}$. Given a schedule X, a team t, and rounds r_1 and r_2 defined as above, the schedule obtained by swapping the opponents of each team in U in rounds r_1 and r_2 is a neighbor of X in N_4.

Teams	Rounds						
	1	$r=2$	3	4	5	6	7
$t_1=1$	-4	8	6	-2	-3	5	7
$t_2=2$	-6	-5	4	1	-8	-7	3
3	5	-4	-7	-6	1	8	-2
4	1	3	-2	-8	7	-6	-5
5	-3	2	-8	-7	6	-1	4
6	2	-7	-1	3	-5	4	8
7	-8	6	3	5	-4	2	-1
8	7	-1	5	4	2	-3	-6

(a) Original schedule X

Teams	Rounds						
	1	$r=2$	3	4	5	6	7
$t_1=1$	-4	5	6	-2	8	7	-3
$t_2=2$	-6	-8	4	1	3	-5	-7
3	5	-4	-7	-6	-2	8	1
4	1	3	-2	-8	7	-6	-5
5	-3	-1	-8	-7	6	2	4
6	2	-7	-1	3	-5	4	8
7	-8	6	3	5	-4	-1	2
8	7	2	5	4	-1	-3	-6

(b) New schedule after move in neighborhood N_3

Fig. 2 Move in neighborhood N_3 for a tournament with eight teams and $r = 2$, $t_1 = 1$, and $t_2 = 2$ (highlighted entries in (a) appear modified in (b) after the move).

Figure 3 shows a move in neighborhood N_4 for a tournament with ten teams and $t = 1$, $r_1 = 1$, and $r_2 = 4$. In this case, $U = \{1,6,7\}$. Teams 3, 4, and 8 are the opponents of teams in U in rounds 1 and 4. We notice that the partial round swap neighborhood N_4 is a generalization of the round swap neighborhood N_2.

Moves in neighbors N_1 and N_2 do not alter the current 1-factorization. In some situations, a move in neighborhood N_3 is equivalent to a move in neighborhood N_1. This is true, in particular, for the canonical 1-factorization for $n = 20$. In this case, for any round $r \in R$ and for any two teams $t_1 \in V$ and $t_2 \in V$, the minimum cardinality subset S differs from the complete set R of rounds exclusively by the round in which t_1 plays against t_2. Similarly, moves in neighborhood N_4 may be equivalent to moves in neighborhood N_2, again as it is the case for the canonical 1-factorization for $n = 20$. Therefore, if the canonical 1-factorization is used as the initial solution in these situations, any search method using only the neighborhoods N_1, N_2, N_3, and N_4 will not be able to escape from the canonical 1-factorization. However, the same does not hold for the 1-factorizations built by the second strategy, in which neighborhoods N_3 and N_4 are different from N_1 and N_2, respectively.

2.3 Local search

We propose a local search procedure exploring neighborhoods N_1 and N_2. Moves in these neighborhoods do not change the original 1-factorization. Moves in neighborhoods N_3 and N_4 will be used exclusively as perturbations, to modify the structure of the original 1-factorization built by the constructive procedure.

Teams	Rounds								
	$r_1 = 1$	2	3	$r_2 = 4$	5	6	7	8	9
$t = 1$	-3	10	7	-8	-5	2	4	9	-6
2	9	8	6	-5	-3	-1	10	-4	-7
3	1	-4	9	-6	2	-10	-7	-8	5
4	-6	3	-10	-7	8	5	-1	2	9
5	10	7	8	2	1	-4	-9	-6	-3
6	4	-9	-2	3	-10	-7	-8	5	1
7	-8	-5	-1	4	-9	6	3	-10	2
8	7	-2	-5	1	-4	-9	6	3	-10
9	-2	6	-3	10	7	8	5	-1	-4
10	-5	-1	4	-9	6	3	-2	7	8

(a) Original schedule X

	Rounds								
Teams	$r_1 = 1$	2	3	$r_2 = 4$	5	6	7	8	9
$t = 1$	-8	10	7	-3	-5	2	4	9	-6
2	9	8	6	-5	-3	-1	10	-4	-7
3	-6	-4	9	1	2	-10	-7	-8	5
4	-7	3	-10	-6	8	5	-1	2	9
5	10	7	8	2	1	-4	-9	-6	-3
6	3	-9	-2	4	-10	-7	-8	5	1
7	4	-5	-1	-8	-9	6	3	-10	2
8	1	-2	-5	7	-4	-9	6	3	-10
9	-2	6	-3	10	7	8	5	-1	-4
10	-5	-1	4	-9	6	3	-2	7	8

(b) New schedule after move in neighborhood N_4

Fig. 3 Move in neighborhood N_4 for a tournament with ten teams and $t = 1$, $r_1 = 1$, and $r_2 = 4$ (highlighted entries in (a) appear modified in (b) after the move).

Let $v(X)$ and $d(X)$ be, respectively, the number of constraint violations and the total traveled distance in the current solution X. All moves in neighborhoods N_1 and N_2 are evaluated at each local search iteration, each of them in time $O(n)$. We compute the number $v(X')$ of constraint violations and the total traveled distance $d(X')$ for each neighbor X' of the current solution X.

If there is at least one neighbor solution X' such that $v(X') \leq v(X)$ and $d(X') < d(X)$, then the current solution X is replaced by its neighbor with minimum traveled distance among all those satisfying the above condition (i.e., the current solution is replaced by a least cost neighbor which does not deteriorate the number of constraint violations). Otherwise, if no move is able to improve the traveled distance of the current solution X without increasing the number of constraint violations in the latter, then the current solution is replaced by its neighbor decreasing the most the number of constraint violations. If no such a move exists, then the local search procedure stops and the current locally optimal solution is returned.

2.4 ILS heuristic

The Iterated Local Search (ILS) metaheuristic (see Lourenço et al (2003)) proposes the use of perturbations to escape from locally optimal solutions. The method starts by constructing an initial solution and applying a local search procedure to it. The current solution is perturbed at each iteration and local search is applied to the perturbed solution. Next, the solution resulting from perturbation and local is compared with the current solution. The former is accepted as the new current solution if some predefined acceptance criterion is met. Otherwise, a new iteration formed by perturbation followed by local search is performed. The procedure stops when some stopping criterion is reached. Algorithm 1 depicts the pseudo-code with the main steps of the ILS metaheuristic.

Algorithm 1: Pseudo-code of the ILS metaheuristic.

```
1  d* ← ∞ ;
2  X ← BuildInitialSolution ;
3  X ← LocalSearch(X) ;
4  repeat
5      if v(X) = 0 and d(X) < d* then
6          X* ← X ;
7          d* ← d(X) ;
8      end
9      X' ← Perturbation(X) ;
10     X' ← LocalSearch(X') ;
11     X ← AcceptanceCriterion(X, X') ;
12 until stopping condition ;
```

The traveled distance associated with the best feasible solution found is initialized in line 1. The constructive procedures presented in Section 2.1 are used to build initial solutions in line 2. The local search procedure applied in lines 3 and 10 follows the strategy described in Section 2.3. Three different procedures are cyclically used for perturbing solutions in line 9: (1) a randomly generated move in neighborhood N_3, (2) a randomly generated move in neighborhood N_4, (3) a randomly generated move in neighborhood N_3 followed by a randomly generated move in neighborhood N_4. The solution X' obtained after the application of local search to the perturbed solution is accepted as the new current solution X in line 11 if and only if it satisfies one of the conditions below:

1. $v(X') < v(X)$ (the new solution X' has fewer constraint violations than the current solution X); or
2. $v(X') = v(X)$ and $d(X') < d(X)$ (the new solution X' has the same number of constraint violations as the current solution X, but the traveled distance according to schedule X' is smaller than that associated with X); or
3. if at least 100 iterations have been performed since the last update of the current solution X, $v(X') \leq v(X)$ (the number of constraint violations in the new solution X' is not greater than that in the current solution X), and $d(X') \leq 1.01 \times d(X)$ (the traveled distance associated with the new schedule X' deteriorates by at most 1% the traveled distance according with the current schedule X).

The acceptance criterion is primarily driven to finding solutions reducing the number of constraint violations and, secondly, to finding improving solutions which do not deteriorate the number of constraint violations. The best feasible solution found during the search is updated in lines 5 to 8 and returned when the stopping condition in line 12 is met.

3 Experimental results

In this section we report on the computational experiments performed to evaluate the proposed heuristic. The ILS heuristic was coded in C++ and compiled with version 4.1.2 of g++ with the optimization flag -O3. The experiments have been performed on an Intel Xeon CPU with a 3.00 GHz processor and 2 Gbytes of RAM, running under the operating system Debian GNU/Linux 4.0.

3.1 Test problems

We used in the computational experiments the same instances proposed and used by Melo et al (2008). There are a total of 40 instances, 20 of them with 18 teams and the other 20 with 20 teams. Distances are the same as in instances circ18 and circ20 of the TTP, both of them available from Trick (2007). For each instance size (i.e., the number n of teams), 20 distinct home-away assignments were created: ten out of the 20 assignments are balanced (i.e., each team plays at least $n/2-1$ games at home and at least $n/2-1$ away games), while the remaining ten assignments are unbalanced. Two instances of each size were shown to be infeasible by Melo et al (2008).

3.2 Numerical results

In the first experiment, we evaluate the impact of the use of the perturbations in neighborhoods N_3 and N_4, considering the instances with 18 teams. We compare the solution obtained by the ILS heuristic using initial solutions determined by canonical 1-factorizations with those obtained with a multi-start algorithm using the same initial solutions and the same local search procedure.

Table 1 shows the numerical results obtained after five executions of each algorithm with a time limit of 720 seconds. The first column depicts the instance name. The second, third, and forth column give the best, average, and worst traveled distances obtained by the multi-start heuristic. The next three columns show the same information for the ILS heuristic. The last column shows the improvement (i.e., the reduction) in percent in the value of the best solution obtained by the multi-start algorithm when the ILS heuristic is applied.

The use of the perturbations in neighborhoods N_3 and N_4 was essential for the performance of the ILS heuristic. The latter improved the traveled distances obtained by the multi-start algorithm by 11.50% on average and obtained better solutions for all instances. The perturbations in these neighborhoods allowed the heuristic to escape from the initial canonical 1-factorizations, performing a more thorough search on the solution space.

The same experiment was performed on the instances with 20 teams, whose numerical results are displayed in Table 2. In this case, the ILS heuristic performed worst than the multi-start algorithm. This is due to the fact that neighborhoods N_3 and N_4 are equivalent to neighborhoods N_1 and N_2, respectively. Consequently, the perturbations do not allow the ILS heuristic to escape from the initial canonical 1-factorization. Furthermore, since neighborhoods N_1 and N_2 are used in the local search procedure, the perturbation moves are of the same type of those used in the local search, implying a great risk of returning to the same solution after the local search.

To overcome the above difficulty, derived from the use of canonical 1-factorizations as initial solutions, the second construction strategy described in Section 2.1 was used for the

Instance	multi-start			ILS			Improvement (%)
	Best	Average	Worst	Best	Average	Worst	
circ18abal	948	1077	1192	850	859.2	870	10.34
circ18bbal	948	1078	1206	844	860	876	10.97
circ18cbal	942	1075	1200	856	859.6	864	9.13
circ18dbal	944	1070	1184	842	853.6	870	10.81
circ18ebal	938	1070	1198	838	862.8	878	10.66
circ18fbal	940	1074	1202	834	852	876	11.28
circ18gbal	934	1071	1206	842	851.2	864	9.85
circ18hbal	918	1059	1184	838	872	948	8.71
circ18ibal	944	1074	1208	848	862.4	878	10.17
circ18jbal	950	1070	1208	830	846.4	854	12.63
circ18anonbal	992	1107	1264	862	880.8	908	13.10
circ18bnonbal				infeasible			
circ18cnonbal				infeasible			
circ18dnonbal	962	1096	1220	854	872	882	11.23
circ18enonbal	1000	1113	1256	854	874.4	888	14.60
circ18fnonbal	1012	1115	1212	864	874	900	14.62
circ18gnonbal	998	1106	1242	856	868.8	876	14.23
circ18hnonbal	978	1109	1222	860	881.2	904	12.07
circ18inonbal	988	1111	1224	860	876.8	906	12.96
circ18jnonbal	950	1085	1204	858	874.8	888	9.68
Average	960.33	1086.67	1212.89	849.44	865.67	885.00	11.50

Table 1 Numerical results for the multi-start and ILS heuristics using canonical 1-factorizations (18 teams).

Instance	multi-start			ILS			Improvement (%)
	Best	Average	Worst	Best	Average	Worst	
circ20abal	1312	1494	1640	1344	1363	1378	-2.44
circ20bbal	1304	1482	1634	1356	1368	1380	-3.99
circ20cbal	1336	1487	1632	1352	1362	1374	-1.20
circ20dbal	1300	1485	1660	1354	1367	1378	-4.15
circ20ebal	1316	1486	1640	1352	1379	1390	-2.74
circ20fbal	1310	1481	1640	1348	1365	1378	-2.90
circ20gbal	1296	1477	1650	1346	1369	1390	-3.86
circ20hbal	1314	1491	1644	1354	1376	1396	-3.04
circ20ibal	1342	1490	1658	1348	1367	1388	-0.45
circ20jbal	1322	1484	1658	1330	1354	1380	-0.61
circ20anonbal	1450	1534	1606	1432	1449	1476	1.24
circ20bnonbal	1356	1522	1662	1376	1409	1426	-1.47
circ20cnonbal	1392	1536	1652	1402	1430	1452	-0.72
circ20dnonbal	1426	1547	1658	1432	1444	1464	-0.42
circ20enonbal	1426	1552	1664	1446	1458	1474	-1.40
circ20fnonbal				infeasible			
circ20gnonbal	1412	1544	1670	1420	1446	1478	-0.57
circ20hnonbal				infeasible			
circ20inonbal	1338	1511	1648	1390	1406	1416	-3.89
circ20jnonbal	1332	1508	1648	1358	1387	1406	-1.95
Average	1349.11	1506.17	1648.00	1374.44	1394.39	1412.44	-1.92

Table 2 Numerical results for the multi-start and ILS heuristics using canonical 1-factorizations (20 teams).

instances with $n = 20$. Table 3 shows the results obtained with this alternative construction strategy for the instances with 20 teams. They show that the use of the modified initial 1-factorizations allowed the ILS heuristic to escape from the initial solutions through the perturbations in neighborhoods N_3 and N_4. Without the use of perturbations, the multi-start

Instance	multi-start (new construction)			ILS (new construction)			Improvement (%)
	Best	Average	Worst	Best	Average	Worst	
circ20abal	1288	1485	1668	1170	1200	1232	9.16
circ20bbal	1320	1486	1664	1200	1204	1218	9.09
circ20cbal	1324	1477	1666	1192	1211	1238	9.97
circ20dbal	1330	1479	1656	1172	1207	1230	11.88
circ20ebal	1302	1485	1660	1202	1212	1228	7.68
circ20fbal	1304	1471	1640	1202	1208	1224	7.82
circ20gbal	1304	1470	1640	1188	1197	1204	8.90
circ20hbal	1314	1472	1644	1190	1224	1242	9.44
circ20ibal	1296	1467	1646	1172	1202	1222	9.57
circ20jbal	1328	1487	1648	1182	1194	1204	10.99
circ20anonbal	1430	1511	1618	1208	1225	1238	15.52
circ20bnonbal	1316	1510	1670	1222	1234	1246	7.14
circ20cnonbal	1400	1535	1666	1190	1229	1268	15.00
circ20dnonbal	1416	1544	1656	1234	1256	1278	12.85
circ20enonbal	1382	1538	1674	1232	1244	1266	10.85
circ20fnonbal				infeasible			
circ20gnonbal	1414	1536	1656	1244	1256	1280	12.02
circ20hnonbal				infeasible			
circ20inonbal	1322	1480	1660	1208	1217	1226	8.62
circ20jnonbal	1320	1493	1668	1194	1215	1240	9.55
Average	1339.44	1495.89	1655.56	1200.11	1218.61	1238.00	10.34

Table 3 Numerical results for the multi-start and ILS heuristics using the modified canonical 1-factorizations (20 teams).

algorithm gets stuck at the initial 1-factorization and explores a very small fraction of the solution space.

Comparing the results in Tables 2 and 3, we notice that the proposed ILS heuristic performed much better using the modified canonical 1-factorization as a initial solutions. The distances traveled in the best solutions found using the modified canonical 1-factorizations are 12.68% smaller in average than those found using the canonical 1-factorizations.

Finally, we compare the results found by the ILS heuristic with those presented by Melo et al (2008), obtained in a computational environment similar to that used in this work. The initial solutions for the ILS algorithm were built with the canonical 1-factorization for the instances with 18 teams, while the modified canonical 1-factorizations were used for the instances with 20 teams.

Table 4 displays the numerical results. The first column gives the instance name. The second column shows the best solution value among those found by the four algorithms described by Melo et al (2008) after two hours of running time. The next two columns give the traveled distance in the best solution obtained with a single run of the ILS heuristic for 30 seconds, together with the corresponding improvement over the best result in Melo et al (2008). The next two columns show the same information for the best solution found after five runs with a time limit of 720 seconds each.

The ILS heuristic proposed in this work clearly outperformed the algorithms described by Melo et al (2008). Table 4 shows that running the ILS heuristic for 30 seconds improved the best solution by at least 8.48% for every instance.

Instance	Previous best	ILS (30 seconds)		ILS (five runs of 12 minutes)	
		Distance	Improvement (%)	Distance	Improvement (%)
circ18abal	1106	914	17.36	850	23.15
circ18bbal	1100	914	16.91	844	23.27
circ18cbal	1038	950	8.48	856	17.53
circ18dbal	1096	932	14.96	842	23.18
circ18ebal	1074	936	12.85	838	21.97
circ18fbal	1060	900	15.09	834	21.32
circ18gbal	1100	880	20.00	842	23.45
circ18hbal	1094	948	13.35	838	23.40
circ18ibal	1102	952	13.61	848	23.05
circ18jbal	1078	938	12.99	830	23.01
circ18anonbal	1124	942	16.19	862	23.31
circ18bnonbal	infeasible				
circ18cnonbal	infeasible				
circ18dnonbal	1060	942	11.13	854	19.43
circ18enonbal	1092	950	13.00	854	21.79
circ18fnonbal	1098	944	14.03	864	21.31
circ18gnonbal	1098	968	11.84	856	22.04
circ18hnonbal	1110	962	13.33	860	22.52
circ18inonbal	1104	950	13.95	860	22.10
circ18jnonbal	1102	928	15.79	858	22.14
circ20abal	1520	1316	13.42	1170	23.03
circ20bbal	1530	1326	13.33	1200	21.57
circ20cbal	1470	1286	12.52	1192	18.91
circ20dbal	1464	1298	11.34	1172	19.95
circ20ebal	1526	1280	16.12	1202	21.23
circ20fbal	1546	1266	18.11	1202	22.25
circ20gbal	1536	1288	16.15	1188	22.66
circ20hbal	1516	1290	14.91	1190	21.50
circ20ibal	1544	1314	14.90	1172	24.09
circ20jbal	1484	1290	13.07	1182	20.35
circ20anonbal	1502	1342	10.65	1208	19.57
circ20bnonbal	1522	1340	11.96	1222	19.71
circ20cnonbal	1488	1352	9.14	1190	20.03
circ20dnonbal	1510	1358	10.07	1234	18.28
circ20enonbal	1574	1358	13.72	1232	21.73
circ20fnonbal	infeasible				
circ20gnonbal	1540	1376	10.65	1244	19.22
circ20hnonbal	infeasible				
circ20inonbal	1516	1298	14.38	1208	20.32
circ20jnonbal	1516	1348	11.08	1194	21.24
Average	1303.89	1127.11	13.62	1024.78	21.49

Table 4 Comparative results: best solutions.

4 Concluding remarks

In this paper, we proposed an ILS heuristic for the traveling tournament problem with predefined venues. Two construction methods for building initial solutions were devised and evaluated. Four neighborhoods were investigated and explored by the ILS heuristic. Two of the neighborhoods allow the heuristic to escape from canonical 1-factorizations. We have showed that canonical 1-factorizations should not be used to produce initial solutions in some situations, for which the modified 1-factorizations are very appropriate and should be used.

The new ILS heuristic clearly outperformed the previous heuristics in the literature, improving the best known solution values by at least 8.48% for every benchmark problem after 30 seconds of running time. The average reduction over all feasible instances amounted to 13.62%. The running times needed to find solutions improving those in the literature are often as small as three seconds. Even better results can be obtained if longer running times are accepted.

Acknowledgements Fabrício N. Costa is supported by a FAPEMIG grant. Sebastián Urrutia is partially supported by FAPEMIG (Edital Universal). Celso C. Ribeiro is partially supported by CNPq research grants 301694/2007-9 and 485328/2007-0, and by FAPERJ research grant E-152.522/2006.

References

Anagnostopoulos A, Michel L, Hentenryck PV, Vergados Y (2006) A simulated annealing approach to the traveling tournament problem. Journal of Scheduling 9:177–193

Durán G, Noronha TF, Ribeiro CC, Souyris S, Weintraub A (2007) Branch-and-cut for a real-life highly constrained soccer tournament scheduling problem. In: Burke E, Rudová H (eds) Practice and Theory of Automated Timetabling VI, Springer, Lecture Notes in Computer Science, vol 3867, pp 174–186

Easton K, Nemhauser G, Trick M (2001) The traveling tournament problem: Description and benchmarks. In: Walsh T (ed) Principles and Practice of Constraint Programming, Springer, Lecture Notes in Computer Science, vol 2239, pp 580–584

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

Knust S (2007) Scheduling non-professional table-tennis leagues. Tech. Rep. 270, Osnabrücker Schriften zur Mathematik

Lourenço HR, Martin OC, Stutzle T (2003) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Springer

Melo RA, Urrutia S, Ribeiro CC (2007) Scheduling single round robin tournaments with fixed venues. In: Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications, Paris, pp 431–438

Melo RA, Urrutia S, Ribeiro CC (2008) The traveling tournament problem with predefined venues. To appear in Journal of Scheduling.

Rasmussen R, Trick M (2008) Round robin scheduling - A survey. European Journal of Operational Research 188:617–636

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

Trick M (2007) Challenge traveling tournament instances. Online reference at `http://mat.gsia.cmu.edu/TOURN/`, last visited on August 10, 2007.

de Werra D (1980) Geography, games, and graphs. Discrete Applied Mathematics 2:327–337

de Werra D (1981) Scheduling in sports. In: Hansen P (ed) Studies on Graphs and Discrete Programming, North Holland, Annals of Discrete Mathematics, vol 11, pp 381–395