Noname manuscript No.
(will be inserted by the editor)

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 moves. Initial solutions are derived from an edge coloring algorithm applied to complete graphs. We showed that canonical edge colorings should not be used as initial solutions in some situations. Instead, the use of Vizing's edge coloring method lead to considerably better results. We also establish that the solution space defined by some commonly used neighborhoods in the sport scheduling literature is not connected in the case of single round robin tournaments, which explains the hardness of finding high quality solutions to some problem instances. Computational results show that the new ILS heuristic performs much better than heuristics based on integer programming and that it improves the best known solutions for benchmark instances.

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

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 (resp. double) round robin (SRR) (resp. 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.

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

Solution methods for round robin scheduling problems usually consist of two stages Trick (2000). The construction of the timetable determines the round in which each game is played. The Home-Away Pattern (HAP) set determines in which condition (home or away) each team plays in each round. Together, the timetable and the home-away pattern set determine the tournament schedule. Literature reviews may be found in Kendall et al (2010); Rasmussen and Trick (2008).

Some problems in the literature consider the construction of both the timetable and the HAP set. 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.

In other problems, the timetable or the HAP set may be fixed and known beforehand. The Breaks Minimization Problem (Régin (2001)) and the Timetable Constrained Distance Minimization Problem (Rasmussen and Trick (2006)) are examples where the timetable is fixed and the objective is to find a HAP set that optimizes a certain objective function. The problem of finding a timetable compatible with a given HAP set appears as a subproblem in several approaches to tackle real-life league scheduling problems (Nemhauser and Trick (1998); Ribeiro and Urrutia (2007b)).

The problem of scheduling a single round robin tournament may be decomposed using a different strategy. Instead of considering a HAP set which determines the playing condition of each team in each round, a Home-Away Assignment (HAA) is considered. The latter determines the venue in which the game between each pair of teams takes place. If the timetable is fixed, the HAP set and HAA provide the same information and determine the schedule. In the following, we consider that the HAA is fixed and known beforehand.

Melo et al (2009) 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 a set J of 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 one of the ordered pairs (i,j) or (j,i) belongs to the set of games to be played. The TTPPV consists in finding a compact single round robin schedule compatible with the predefined HAA, such that the total distance traveled by the n 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.

Consider an instance of TTPPV where the set of teams is $\{1,2,3,4,5,6\}$. Let D_{ij} be the distance between each par of teams, where

$$D = \begin{pmatrix} 0 & 1 & 2 & 3 & 2 & 1 \\ 1 & 0 & 1 & 2 & 3 & 2 \\ 2 & 1 & 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 \\ 2 & 3 & 2 & 1 & 0 & 1 \\ 1 & 2 & 3 & 2 & 1 & 0 \end{pmatrix}.$$

Let

$$\begin{aligned} J = \{\ & (1,2),(1,4),(2,4),(2,6),(3,1), \\ & (3,2),(3,6),(4,3),(4,5),(4,6), \\ & (5,1),(5,2),(5,3),(6,1),(6,5)\ \}. \end{aligned}$$

be the home-way assignment. An optimal solution for this instance is given by the schedule in Figure1. Each line i of the schedule contains the sequence of matches that team i must play. A negative sign indicates an away game and the absence of sign indicates a home game. Consider, for example, the distance traveled by team 3. It performs its first two matches at home against teams 1 and 6. Next, it performs two consecutive away matches , traveling from its venue to the venue of team 4 and from there to the venue of team 5. Finally, it returns home to play its last game against team 2. Therefore, the traveled distance of team 3 is equal to $d_{34} + d_{45} + d_{53} = 1 + 1 + 2 = 4$. The total traveled distance in this solution by all teams is 36.

Teams	**Rounds**				
	1	**2**	**3**	**4**	**5**
1	-3	2	-5	-6	4
2	-5	-1	6	4	-3
3	1	6	-4	-5	2
4	6	5	3	-2	-1
5	2	-4	1	3	-6
6	-4	-3	-2	1	5

Fig. 1 An optimal solution.

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 (2010)).

Instances of the TTPPV with up to eight teams were solved to optimality by the integer programming formulations presented in Melo et al (2009). Since feasible solutions have not been found by a commercial solver (CPLEX 10.0) in two hours of running time for instances with 18 or more teams, four heuristics based on the integer programming formulations were also developed in Melo et al (2009).

In this paper, we propose a local search based heuristic for the TTPPV to find good quality solutions for realistic size instances. The heuristic is shown experimentally to outperform previous approximate solution approaches. Besides this new heuristic itself, this paper presents other contributions: a general method for constructing initial solutions for round robin scheduling problems is proposed, a clear and formal characterization of the neighborhoods mostly often used in the literature is given, and we establish that the solution space defined by these neighborhoods is not connected, which explains the difficulties found by heuristics based on them to find high quality solutions to some problem instances.

The rest of this paper is organized as follows. Section 2 starts by the description of strategies used to build initial solutions. Neighborhoods and their properties are presented next, followed by the local search strategy. The new heuristic, based on the Iterated Local Search (ILS) metaheuristic, is also described in detail. Section 3 reports the computational experiments. Concluding remarks are made in the last section.

2 Heuristic approach

In this section we describe a local search based heuristic for solving the TTPPV. First, we discuss the construction of initial solutions. Next, we describe the properties of the neighborhoods that are used in a local search procedure. Finally, we outline the whole ILS-based heuristic procedure.

2.1 Initial solutions

An edge coloring of a graph is an assignment of colors to its edges, in such a way that, if two distinct edges are incident to the same node, they are assigned different colors. It is known, from Vizing's theorem Vizing (1964), that the minimum number of colors that can be used in any edge coloring of a graph is equal to either Δ or $\Delta+1$, where Δ is the maximum degree of any node in the graph.

Schedules of an SRR tournament with n (even) teams may be represented by edge colorings of the complete graph K_n using exactly $\Delta = n-1$ colors. Each node of this graph represents a team and each color a round. If an edge $\{i, j\}$ is colored by the k-th color, then the game between teams i and j is played in the k-th round.

The so called canonical edge coloring of K_n (also called the canonical 1-factorization, see de Werra (1981)) has been widely used to construct initial solutions for round robin tournaments. In this coloring, the edge set of each color E^i is defined as follows, for $i = 1,\ldots,n-1$:

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

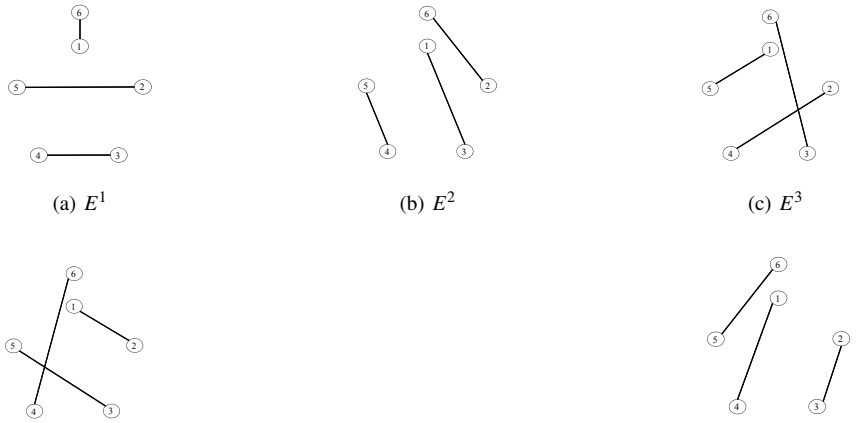
with

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

and

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

Figure 2 shows the canonical edge coloring of K_6.



(a) E^1 (b) E^2 (c) E^3

(d) E^4 (e) E^5

Fig. 2 Canonical edge coloring of K_6.

In order to obtain a greater variety of initial solutions, we describe a different method to construct edge colorings of K_n. We first recall that the number n of teams is even. Using the constructive algorithm directly derived from the proof of Vizing's theorem, we color the edges of the complete graph K_{n-1} using $n-1$ colors. Since $n-1$ is odd, there is a unique color in this coloring that is not used to color the edges incident to each node. Then, we add a new node to K_{n-1} and a new edge connecting each node v of K_{n-1} to the new node. The new edge incident to each node v of K_{n-1} is colored with the color that was not used in the other edges incident to v. This leads to an edge coloring of K_n using exactly $n-1$ colors.

In contrast to the procedure used to construct canonical colorings, the method above described is general in the sense that it is capable to produce any edge coloring of the complete graph K_n using $n-1$ colors, including the canonical coloring itself. Vizing's algorithm colors the edges of the graph one by one. If the nodes are scanned and colored in different orders, it is able to produce any coloring of K_{n-1} using $n-1$ colors. In consequence, it may construct any edge coloring of K_n using $n-1$ colors. In this paper we implemented Vizing's algorithm scanning and coloring the edges of the graph in a random order. Although this procedure is not new in graph theory, to the best of our knowledge it was not used before to build solutions to round robin scheduling problems.

We notice that both construction procedures may build round robin schedules that violate the limits on the maximum number of consecutive home or away games. Infeasibilities will be repaired during local search.

2.2 Neighborhoods

Let $V = \{1,...,n\}$ be the set of teams and $R = \{1,...,n-1\}$ 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 3 shows a schedule for a single round robin 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. 3 Schedule for a single round robin 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 (2007a). The basic home-away swap neighborhood used by Ribeiro and Urrutia (2007a) 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.

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.

Neighborhoods N_1 and N_2 are not used by the heuristic proposed in this work and have only theoretical relevance for this paper.

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. $S = \{r_1,...,r_k\} \subseteq R$ is minimum 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 4 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. $U = \{t_1,...,t_k\} \subseteq V$ is minimum 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.

Figure 5 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 neighborhoods N_1 and N_2 generate solutions whose underlying colorings are isomorphic to that of the current solution before the move. In some situations, it may happen

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. 4 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).

that every move in neighborhood N_3 be equivalent to a move in neighborhood N_1. This was experimentally observed e.g. for the canonical edge coloring when $n \in \{12,14,20\}$. In these cases, 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, it is the case for the canonical edge coloring when $n \in \{12,14,20\}$. Therefore, if the canonical edge coloring is used as the initial solution in these situations, any search method using only neighborhoods N_1, N_2, N_3, and N_4 will not be able to visit solutions whose underlying colorings are different from the canonical edge coloring. However, the same does not hold for other colorings built by the general method, for which neighborhoods N_3 and N_4 are not equivalent to N_1 and N_2, respectively.

In the following, we show that any move in neighbor N_2 is equivalent to a sequence of moves in neighborhood N_4.

We first observe that each move in N_4 with parameters t, r_1, and r_2 involves a subset of the games affected by a move in N_2 using parameters r_1 and r_2. In fact, the teams involved in the move in N_4 are those in the subset U and their opponents in rounds r_1 and r_2, which are computed during the move (see above). We also remark that if, instead of using team t as a move parameter, we use another team $t_1 \in U$, the subset U_1 of teams computed during the move turns out to be equal to U and, in consequence, it does not change the move. If, instead of using team t, we use another team t_2 that is an opponent of a team in U in round r_1 or r_2, the subset U_2 of teams computed during the move turns out to be equal to the set of opponents of teams in U in rounds r_1 and r_2 and the set of opponents of teams in U_2 turns

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

Teams	Rounds								
	$r_1 = 1$	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. 5 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).

out to be equal to U. Therefore, once again the move is the same. Finally, if instead of using team t as a move parameter, we use another team t_3 that does not belong to U neither is an opponent of a team in U in round r_1 or r_2, the subset U_3 of teams computed during the move turns out to be disjoint to U and to the set of opponents of teams in U in rounds r_1 and r_2. Hence, in this case the move involves different games between rounds r_1 and r_2. Therefore, the set of teams may be partitioned into subsets $P_1, \ldots, P_p$, so as that for any $i = 1, \ldots, p$ the moves in neighborhood N_4 with parameters t, r_1, and r_2 are the same for every $t \in P_i$. To conclude, we remark that any sequence of p moves in N_4 involving rounds r_1 and r_2 and one team from each subset $P_i, i = 1, \ldots, p$, is equivalent to a move in N_2 using r_1 and r_2 as parameters.

A similar argument shows that a sequence of moves in N_3 is equivalent to a move in N_1.

2.3 Local search

We have shown in the last section that any solution that can be reached by a move in N_1 (resp. N_2) can also be reached by a sequence of one or more moves in N_3 (resp. N_4). Therefore, the set of solutions connected through neighborhoods N_1 and N_2 is included in that connected through neighborhoods N_3 and N_4. Since neighborhoods N_3 and N_4 generalize neighborhoods N_1 and N_2 and solutions in the two former neighborhoods can be computed

as quickly as those in the two latter, we propose a local search procedure exploring only neighborhoods N_3 and N_4.

Let $v(X)$ and $d(X)$ be, respectively, the number of constraint violations and the total traveled distance in the schedule X corresponding to the current solution. All moves in neighborhoods N_3 and N_4 are evaluated at each local search iteration, each of them in time $O(n)$. The number $v(X')$ of constraint violations and the total traveled distance $d(X')$ are computed 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 by 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 followed by local search 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. One of the constructive procedures presented in Section 2.1 (i.e., the canonical coloring or Vizing's algorithm) is used to build initial solutions in line 2. To build an edge coloring, teams are randomly associated to nodes. The local search procedure applied in lines 3 and 10 follows the strategy described in Section 2.3. The perturbation in line 9 consists in a randomly generated sequence of two, three, or four moves in $N_3 \cup N_4$. 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 $d(X') \leq 1.01 \cdot d(X)$ (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 ILS heuristic, which 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 Core 2 Quad CPU with a 2.50 GHz processor and 2 Gbytes of RAM, running under the operating system Debian GNU/Linux 4.0.

The same 40 instances considered by Melo et al (2009) were used in the computational experiments: 20 instances with 18 teams and 20 instances with 20 teams. Distances are the same as in instances circ18 and circ20 of the TTP, both of them available from Trick (2009). 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 $n/2-1$ home games 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 (2009).

In the first experiment, we compare the performance of the heuristic when using the canonical edge coloring and an edge coloring produced by the general Vizing method to build the initial solution. Tables 1 and 2 show the numerical results obtained by ten runs of the ILS heuristic with a time limit of five minutes using each of the edge coloring methods to build the initial solutions, for instances with 18 and 20 teams, respectively. The first column in each table shows the instance name. The second, third, and fourth columns give the best, average, and worst solution values obtained using Vizing's algorithm to obtain the initial edge colorings. The next three columns show the same information for the algorithm using the canonical coloring to build initial solutions. The last column shows the improvement in the average solution value obtained by using Vizing's algorithm in the construction phase to replace the canonical coloring.

The results obtained by the ILS heuristic using any of the two initial edge colorings are quite similar for the instances with 18 teams. However, the use of Vizing's edge coloring method considerably improved the best solution values for the instances with 20 teams. The solutions obtained by Vizing's algorithm improved those obtained from canonical colorings by at least 17.5%, with an average improvement of 19.0%. This is due to the fact that, for $n =$

Instance	Canonical			Vizing			
	Average	Best	Worst	Average	Best	Worst	Improvement (%)
circ18abal	824.4	812	844	821.6	782	834	0.3
circ18bbal	824.8	810	842	826.2	800	848	-0.2
circ18cbal	820.8	802	842	820.2	802	846	0.1
circ18dbal	827.0	816	836	816.4	802	828	1.3
circ18ebal	819.8	812	836	823.8	804	836	-0.5
circ18fbal	822.4	800	840	822.4	814	832	0.0
circ18gbal	820.2	806	832	819.6	806	842	0.1
circ18hbal	817.0	804	832	816.2	796	842	0.1
circ18ibal	814.4	798	832	823.0	810	850	-1.1
circ18jbal	816.4	804	832	815.8	798	838	0.1
circ18anonbal	836.4	810	858	827.8	818	838	1.0
circ18dnonbal	836.4	822	848	838.8	812	860	-0.3
circ18enonbal	842.4	814	864	839.2	808	862	0.4
circ18fnonbal	837.2	816	856	842.0	828	854	-0.6
circ18gnonbal	823.4	806	842	840.8	816	864	-2.1
circ18hnonbal	849.2	832	870	835.8	822	870	1.6
circ18inonbal	845.8	828	860	847.6	830	866	-0.2
circ18jnonbal	826.2	808	836	819.2	800	850	0.8
Average	828.0	811.1	844.6	827.6	808.2	847.8	0.1

Table 1 Numerical results obtained by the heuristic (18 teams).

Instance	Canonical			Vizing			
	Average	Best	Worst	Average	Best	Worst	Improvement (%)
circ20abal	1381.0	1360	1396	1135.0	1118	1158	17.8
circ20bbal	1365.8	1340	1396	1134.4	1108	1164	16.9
circ20cbal	1371.6	1348	1390	1119.6	1074	1142	18.4
circ20dbal	1380.8	1362	1392	1140.8	1118	1168	17.4
circ20ebal	1379.0	1364	1396	1130.6	1108	1164	18.0
circ20fbal	1374.4	1362	1388	1125.0	1098	1150	18.1
circ20gbal	1373.8	1350	1386	1123.0	1104	1142	18.3
circ20hbal	1380.4	1350	1410	1131.4	1106	1164	18.0
circ20ibal	1376.6	1366	1392	1129.0	1112	1156	18.0
circ20jbal	1377.4	1370	1384	1121.2	1112	1138	18.6
circ20anonbal	1476.0	1430	1512	1150.0	1124	1188	22.1
circ20bnonbal	1420.6	1396	1442	1154.6	1122	1184	18.7
circ20cnonbal	1441.8	1420	1460	1150.2	1132	1170	20.2
circ20dnonbal	1455.6	1412	1478	1155.2	1122	1172	20.6
circ20enonbal	1477.2	1452	1496	1176.2	1156	1198	20.4
circ20gnonbal	1466.2	1434	1500	1159.6	1122	1192	20.9
circ20inonbal	1400.6	1362	1420	1141.8	1122	1166	18.5
circ20jnonbal	1387.4	1362	1408	1134.4	1116	1148	18.2
Average	1404.8	1380.0	1424.8	1139.6	1115.2	1164.7	18.8

Table 2 Numerical results obtained by the heuristic (20 teams).

20, neighborhoods N_3 and N_4 are equivalent to N_1 and N_2, respectively, if the canonical edge coloring is used. Since moves in N_1 and N_2 do not change the edge coloring associated with the current schedule, in this case moves in N_3 and N_4 do not change it either. In consequence, the ILS heuristic gets stuck by the structure of the canonical edge coloring and explores just a small portion of the solution space. This phenomenon does not happen when an edge coloring obtained by Vizing's algorithm is used.

To conclude, we compare the average result obtained by the ILS heuristic in five minutes of processing time with those presented in Melo et al (2009) and obtained in a computational environment similar to the one used in this work. Table 3 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 (2009) after two hours of running time. The next two columns give the average traveled distance in the solutions found by ten executions of the ILS heuristic for five minutes, together with the corresponding average improvement over the best result in Melo et al (2009). The ILS heuristic clearly outperformed the algorithms in Melo et al (2009): the average cost of the solution found running the ILS heuristic for five minutes improved in the best known solution values by at least 20.9% for every instance and by 24.5% on average.

Instance	Best in Melo et al (2009)	ILS	Improvement (%)
circ18abal	1106	821.6	25.7
circ18bbal	1100	826.2	24.9
circ18cbal	1038	820.2	21.0
circ18dbal	1096	816.4	25.5
circ18ebal	1074	823.8	23.3
circ18fbal	1060	822.4	22.4
circ18gbal	1100	819.6	25.5
circ18hbal	1094	816.2	25.4
circ18ibal	1102	823.0	25.3
circ18jbal	1078	815.8	24.3
circ18anonbal	1124	827.8	26.4
circ18dnonbal	1060	838.8	20.9
circ18enonbal	1092	839.2	23.2
circ18fnonbal	1098	842.0	23.3
circ18gnonbal	1098	840.8	23.4
circ18hnonbal	1110	835.8	24.7
circ18inonbal	1104	847.6	23.2
circ18jnonbal	1102	819.2	25.7
circ20abal	1520	1135.0	25.3
circ20bbal	1530	1134.4	25.9
circ20cbal	1470	1119.6	23.8
circ20dbal	1464	1140.8	22.1
circ20ebal	1526	1130.6	25.9
circ20fbal	1546	1125.0	27.2
circ20gbal	1536	1123.0	26.9
circ20hbal	1516	1131.4	25.4
circ20ibal	1544	1129.0	26.9
circ20jbal	1484	1121.2	24.4
circ20anonbal	1502	1150.0	23.4
circ20bnonbal	1522	1154.6	24.1
circ20cnonbal	1488	1150.2	22.7
circ20dnonbal	1510	1155.2	23.5
circ20enonbal	1574	1176.2	25.3
circ20gnonbal	1540	1159.6	24.7
circ20inonbal	1516	1141.8	24.7
circ20jnonbal	1516	1134.4	25.2
Average			24.5

Table 3 Comparison between the average results obtained by the ILS heuristic in five minutes with the best previously known solutions.

4 Concluding remarks

We proposed an ILS heuristic for the traveling tournament problem with predefined venues. A general strategy for building initial solutions was developed and evaluated. We showed that canonical edge colorings should not be used as initial solutions in some situations. Instead, the use of Vizing's edge coloring method lead to considerably better results, improving the best solution values for the instances with 20 teams by 19% on average.

Two neighborhoods were investigated and explored by the ILS heuristic. These neighborhoods allow the heuristic to escape from locally optimal solutions. We have also given a characterization of some neighborhoods often used in the literature. We established that the solution space defined by these neighborhoods is not connected, which explains the difficulties found by local search heuristics based on them to find high quality solutions to some problems.

The new ILS heuristic clearly outperformed the previous heuristics in the literature, improving the best known solution values by at least 20.9% for every benchmark problem after five minutes of running time. The average reduction over all feasible instances amounted to 24.5%. 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) and CNPq research grant 302560/2007-6. 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

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

Knust S (2010) Scheduling non-professional table-tennis leagues. European Journal of Operational Research 200:358–367

Lourenço HR, Martin OC, Stutzle T (2003) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics, Springer, pp 321–353

Melo RA, Urrutia S, Ribeiro CC (2009) The traveling tournament problem with predefined venues. Journal of Scheduling 12:607–622

Nemhauser G, Trick M (1998) Scheduling a major college basketball conference. Operations Research 46:1–8

Rasmussen R, Trick M (2006) The timetable constrained distance minimization problem. In: Beck J, Smith B (eds) Integration of AI and OR Techniques in Constraint Programming

for Combinatorial Optimization Problems, Springer, Lecture Notes in Computer Science, vol 3990, pp 167–181

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

Régin J (2001) Minimization of the number of breaks in sports scheduling problems using constraint programming. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 57:115–130

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

Ribeiro CC, Urrutia S (2007b) Scheduling the Brazilian soccer championship. In: Burke E, Rudová H (eds) Practice and Theory of Automated Timetabling VI, Springer, Lecture Notes in Computer Science, vol 3867, pp 149–159

Trick M (2000) A schedule-then-break approach to sports timetabling. In: Burke E, Erben W (eds) PATAT, Springer Lecture Notes in Computer Science, vol 2079, Springer-Verlag, pp 242–253

Trick M (2009) Challenge traveling tournament instances. Online reference at `http://mat.gsia.cmu.edu/TOURN/`, last visited on August 19, 2009.

Vizing VG (1964) On an estimate of the chromatic class of a p-graph (in russian). Metody Discret Analiz 3:25–30

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