Heuristics for the Mirrored Traveling Tournament Problem

Celso C. Ribeiro[1] and Sebastián Urrutia[2]

[1] Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria 156, Niterói, RJ 24210-240, Brazil.
celso@inf.puc-rio.br

[2] Department of Computer Science, Catholic University of Rio de Janeiro, Rua Marquês de São Vicente 225, Rio de Janeiro, RJ 22453-900, Brazil.
useba@inf.puc-rio.br

Abstract. Professional sports leagues are a major economic activity around the world. Teams and leagues do not want to waste their investments in players and structure in consequence of poor schedules of games. Game scheduling is a difficult task, involving several decision makers, different types of constraints, and multiple objectives to optimize. The Traveling Tournament Problem abstracts certain types of sport timetabling issues, where the objective is to minimize the total distance traveled by the teams. In this work, we tackle the mirrored version of this problem. We first propose a fast and effective constructive algorithm. We also describe a new heuristic based on the combination of the GRASP and Iterated Local Search metaheuristics. A strong neighborhood based on ejection chains is also proposed and leads to significant improvements in solution quality. Very good solutions are obtained for the mirrored problem, sometimes even better than those found by other approximate algorithms for the less constrained non-mirrored version. Computational results are shown for benchmark problems and for a large instance associated with the main division of the 2003 edition of the Brazilian soccer championship, involving 24 teams.

1 Introduction

Professional sports leagues are a major economic activity around the world. Teams and leagues do not want to waste their investments in players and structure in consequence of poor schedules of games. Game scheduling is a difficult task with multiple constraints and objectives (involving e.g. logistic, organizational, economical, and fairness issues), as well as several decision makers (such as league officials, team managers, and TV executives).

Efficient schedules are of major interest for teams, leagues, sponsors, fans, and the media. This issue may be further complicated due to the distances involved.

In the case of the Brazilian soccer national championship, a single trip from Porto Alegre to Belém takes almost a full journey and many stops, due to the absence of direct flights to cover a distance of approximately 4000 kilometers. The total distance traveled becomes an important variable to be minimized, so as to reduce traveling costs and to give more time to the players for resting and training along a season that lasts for approximately six months. Another possible variable to be minimized is the maximum distance traveled over all teams.

Several authors in different contexts (see e.g. [1, 2, 4, 24, 26–28, 30, 31]) tackled the problem of tournament scheduling in different leagues and sports such as soccer, basketball, hockey, baseball, rugby, cricket, and football, using different techniques such as integer programming, tabu search, genetic algorithms, and simulated annealing. Another promising technique for sports timetabling is constraint programming, which was already used by Henz [19] and lead to impressive improvements in terms of computation time with respect to [24]. The results in [19] were also improved by Zhang [32], using again constraint programming. We refer to Henz [20] for recent advances in constraint programming for scheduling problems in sports. Easton et al. [10] reviewed scheduling problems in sports.

The Traveling Tournament Problem is an inter-mural championship timetabling problem that abstracts certain characteristics of scheduling problems in sports [11]. It combines tight feasibility issues with a difficult optimization problem. The objective is to minimize the total distance traveled by the teams, subject to the constraint that no team can play on a row more than three games at home or away. Since the total distance traveled is a major issue for every team taking part in the tournament, solving a traveling tournament problem may be a starting point for the solution of real timetabling applications in sports.

In this paper, we propose new, effective heuristics for solving the mirrored version of the traveling tournament problem. The contribution of the paper is twofold: first, a new, fast constructive heuristic which provides good initial solutions; and second, a hybridization of the GRASP and ILS (Iterated Local Search) metaheuristics which computes high-quality solutions. The paper is organized as follows. In the next section, the mirrored version of the traveling tournament problem is formulated in detail. In Section 3, we show how a solution of this problem may be represented as an oriented ordered 1-factorization of a complete graph. A constructive heuristic for building good feasible initial solutions is proposed in Section 4. Three simple neighborhoods and an ejection chain neighborhood are described in Section 5. A local search procedure based on these neighborhoods is described in Section 6. In Section 7, we present the new `GRILS-mTTP` heuristic for the mirrored Traveling Tournament Problem, combining the main ideas of the GRASP and ILS metaheuristics. Computational results for 12 benchmark instances and one real-life problem are reported in Section 8. Concluding remarks are made in the last section.

2 The mirrored traveling tournament problem

We consider a tournament played by n teams, where n is an even number. In a *simple round-robin* (SRR) tournament, each team plays every other exactly once in $n-1$ prescheduled rounds. The game between teams i and j is represented by the unordered pair $\{i, j\}$. There are $n/2$ games in each round. Each team plays exactly once in each round. In a *double round-robin* (DRR) tournament, each team plays every other twice, once at home and once away. A *mirrored double round-robin* (MDRR) tournament is a simple round-robin tournament in the first $n-1$ rounds, followed by the same tournament with reversed venues in the last $n-1$ rounds. A road trip is a sequence of consecutive away games for a team, while a home stand is a sequence of consecutive home games. We assume that each team in the tournament has a stadium at its home city. The distances between the home cities are known. Each team is located at its home city at the beginning of the tournament, to where it returns at the end after playing the last away game. Whenever a team plays two consecutive away games, it goes directly from the city of the first opponent to the other, without returning to its own home city.

The Traveling Tournament Problem (TTP) was first established by Easton et al. [11]. Given n (even) teams and the distances between their home cities, the TTP consists in finding a DRR such that every team does not play more than three consecutive home or away games, no repeaters (i.e., two consecutive games between the same two teams at different venues) occur, and the sum of the distances traveled by the teams is minimized. Benchmark instances are available in [29]. Some of them are based on the Major League Baseball (MLB) in the United States, while the others were generated with a particular circular structure. To date, even small benchmark instances of the TTP with $n = 8$ teams cannot be exactly solved. We refer to this problem as the non-mirrored TTP, for which both mirrored and non-mirrored solutions are feasible.

The mirrored Traveling Tournament Problem (mTTP) has an additional constraint: the games played in round k are exactly the same played in round $k+(n-1)$ for $k = 1, \ldots, n-1$, with reversed venues. Repeaters cannot occur in mirrored schedules. Mirrored tournaments are a common tournament structure in Latin America. No attempts to solve this version of the problem were found in the literature.

Some approaches based on metaheuristics were already proposed for the non-mirrored TTP. Crauwels and Van Oudheusden [5, 6] proposed ant system heuristics in which the construction procedure is coupled with a backtracking strategy and a local search algorithm, without good numerical results. Cardemil [3] developed a tabu search algorithm that obtained the best known solutions for some benchmark instances at the time of publication. Initial solutions and those used for restarts were randomly generated. Anagnostopoulos et al. [1] proposed a simulated annealing algorithm that found the best known solutions for the benchmark instances at the time of publication. Once again, the initial solutions were randomly generated. The neighborhood structure is similar to that used in this work. However, we explore our neighborhood structure using a much faster strategy. Their heuristic is time consuming and takes more than one day of computation time for some benchmark instances.

3 Solutions as ordered oriented 1-factorizations

We use the same formulation proposed by de Werra [7, 8] for representing a tournament schedule using 1-factorizations of a graph. A factor of an undirected graph $G = (V, E)$ is a graph $G' = (V, E')$ with $E' \subseteq E$. G' is a 1-factor 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 $E^1 \cup ... \cup E^p = E$. All factors in a 1-factorization of G are 1-factors. An oriented 1-factorization of G is a 1-factorization in which an orientation is set for every edge. In an ordered oriented 1-factorization of G, the 1-factors are taken in a fixed order.

An SRR tournament with n (even) teams may be represented by an ordered oriented 1-factorization of the complete undirected graph K_n. Each node of this graph represents a team. An arc from node i to node j in the k-th 1-factor of an ordered oriented 1-factorization represents the game between teams i and j being played at the home city of team j at round k. There are $(n-1)!$ ordered oriented 1-factorizations of K_n associated with each of its oriented 1-factorizations. Each ordered oriented 1-factorization represents a tournament schedule and each of its $(n-1)$ oriented 1-factors represents a round.

A mirrored double round tournament with n (even) teams may be represented as a single round robin tournament which is repeated with reversed venues. In consequence, we may also use ordered oriented 1-factorizations to represent solutions of MDRR tournaments. This representation is very useful in the study of local search neighborhoods carried out in Section 5.

4 A fast constructive heuristic for the mTTP

Good initial solutions can affect the performance of metaheuristic-based algorithms for timetabling problems. Elmohamed et al. [12] noticed that when a random initial configuration is used for timetabling problems, simulated annealing performs very poorly; however, there is a dramatic improvement in performance when a preprocessor is used to provide a good starting point for the annealing. Fast approximate algorithms are important to timetabling problems in sports, because the different and possibly contrary interests of decision makers require certain interactivity. Very often a decision maker will be unhappy with the obtained solution, and other solutions will have to be computed quickly.

We propose a three-stage constructive heuristic for the mTTP. It explores the fact that an MDRR tournament is composed by two SRR tournaments, with the second one equal to the first except by the reversed venues. In the first stage we build a schedule for an SRR tournament with n abstract teams. Stadiums are not assigned in this stage. In the second stage, the real teams are assigned to the abstract teams. Finally, in the third stage, we create an MDRR tournament assigning stadiums to the games between the real teams.

4.1 Stage 1: Build a 1-factorization

The well known "polygon method" [9] is used to build a schedule for an SRR tournament with n abstract teams, without assigning stadiums to the games.

The abstract teams $1, \ldots, n-1$ are initially placed at clockwise consecutively numbered nodes of a regular polygon with $n-1$ nodes: team 1 in node 1, team 2 in node 2, and so forth. The abstract team n is not placed in the polygon. At each round $k = 1, \ldots, n-1$, the abstract team placed in the node numbered $\ell = 2, \ldots, n/2$ plays against the one placed in the node numbered $n+1-\ell$. At every round, the abstract team placed at node 1 plays against the abstract team n. After each round, each abstract team $1, \ldots, n-1$ is moved clockwise to the immediately next node of the polygon. Figure 1 illustrates the application of this procedure for $n = 6$.

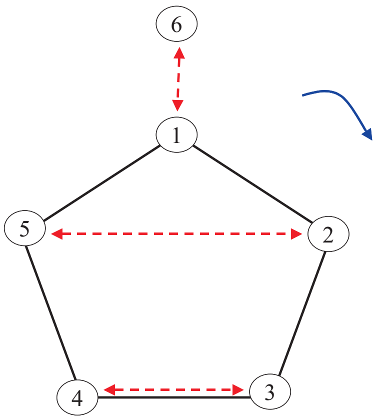

Fig. 1. Polygon method for $n = 6$ (first round).

Next, this schedule is duplicated. This preliminary schedule of an MDRR tournament is used to generate a square $n \times n$ matrix of consecutive opponents. Each entry (i, j) in this matrix is equal to the number of times the abstract teams i and j are consecutive opponents of other teams. Figure 2 displays the

$$
\begin{pmatrix}
0 & 4 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 4 & 3 & 4 & 4 & 4 & 4 & 4 \\
4 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 \\
4 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 \\
4 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 26 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 1 & 26 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 1 & 0 & 2 & 25 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 26 & 2 & 0 & 2 & 25 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 & 25 \\
4 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0 & 2 \\
4 & 2 & 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 2 & 0
\end{pmatrix}
$$

Fig. 2. Matrix of consecutive opponents for $n = 16$.

matrix of consecutive opponents built from the schedule produced by the polygon method for $n = 16$ abstract teams. This figure shows that the polygon method creates several patterns in the schedule. There are several pairs of abstract teams which are consecutively played by many other abstract teams. As an example, we notice that the abstract teams 8 and 10 are consecutive opponents of other teams 26 times along the tournament. Abstract teams which are consecutively played more times should preferably be assigned to real teams with smaller distances between their home cities.

4.2 Stage 2: Assign real teams to abstract teams

Pairs of real teams with close home cities should ideally be assigned to pairs of abstract teams which are consecutively played by many other teams. Assigning real teams to abstract teams amounts basically to solving a quadratic assignment problem.

A quick heuristic is applied in this stage. First, the pairs of abstract teams are sorted by non-increasing entries in the matrix of consecutive opponents. Next, the real teams are sorted by increasing distance to the home city of their closest rivals and assigned to abstract teams in this order. Let t_1 be the next real team to be assigned to an abstract team and let t_2 be the team with the closest home city to t_1:

– If t_2 is already assigned to an abstract team, then find the first pair (a, b) of abstract teams such that a is assigned to t_2 and b is not yet assigned to a real team. In this case, assign the abstract team b to the real team t_1.

– If t_2 is not yet assigned to an abstract team, then find the first pair (a, b) of abstract teams such that none of them is assigned to a real team. In this situation, assign either a or b to the real team t_1.

4.3 Stage 3: Assign stadiums to games

In this last stage, a stadium is assigned to each game. To minimize the total distance traveled, one should attempt to schedule as many games as possible in a road trip. We propose a two-step approach for this stage. The first step builds a feasible schedule, while the second attempts to improve this schedule by reducing the total distance traveled using a quick local search strategy.

Stadiums are randomly assigned to the games in the first round. From round 2 to $n-2$, we use the following strategy to assign a stadium to the game between t_1 and t_2. We denote by n_i the number of games that team i has consecutively played in the previous rounds, either in a home stand or in a road trip.

- Case (1): $n_{t_2} > n_{t_1}$
  - If team t_2 played its last game at home, then schedule the game to the stadium of team t_1;
  - otherwise, schedule the game to the stadium of team t_2.
- Case (2): $n_{t_2} < n_{t_1}$
  - If team t_1 played its last game at home, then schedule the game to the stadium of team t_2;
  - otherwise, schedule the game to the stadium of team t_1.
- Case (3): $n_{t_2} = n_{t_1}$
  - If team t_1 played its last game at home and team t_2 away, then schedule the game to the stadium of team t_2;
  - if team t_2 played its last game at home and team t_1 away, then schedule the game to the stadium of team t_1;
  - otherwise, schedule the game randomly to the stadiums of t_1 or t_2.

The first round of the second phase of an MDRR tournament is exactly the first round of the first phase with reversed venues. Accordingly, the last round and the first round with reversed venues should be considered as consecutive rounds. Stadiums are randomly assigned in the last round. If a stadium assignment makes the schedule infeasible, then the venue for the corresponding game is reversed. If the schedule remains infeasible, all above steps are repeated and new stadium assignments are performed from scratch.

The final step of the stadium assignment stage consists of a quick local search procedure using the schedule above created as the initial solution and the home-away swap neighborhood described in Section 5.1. The local optimum obtained by local search is returned by the constructive procedure.

5 Neighborhoods

Four neighborhood structures are defined and will be later used by the extended GRASP with ILS heuristic described in Section 7.

5.1 Home-away swap (HAS) neighborhood

Each solution in this neighborhood is obtained by reversing the stadium assignment of a single game. Since there are $n \cdot (n-1)/2$ games in each phase of a DRR tournament and the tournament is mirrored, each solution has $n \cdot (n-1)/2$ neighbors in this neighborhood. Only neighbors corresponding to feasible schedules will be visited by the local search procedure described in Section 6.

Solutions are represented as ordered oriented 1-factorizations of K_n. A move in this neighborhood consists in reversing the orientation of a single arc of an oriented 1-factor. Figure 3 illustrates a move in the HAS neighborhood for $n = 8$ teams.

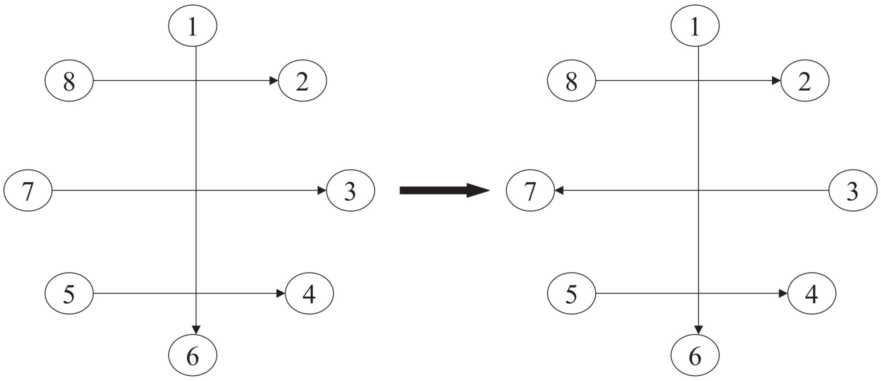

Fig. 3. Move in neighborhood HAS for $n = 8$: the venue of game $\{3, 7\}$ is changed.

5.2 Team swap (TS) neighborhood

Each solution in this neighborhood is obtained by swapping the opponents of any pair of teams t_1 and t_2 in all rounds. If a team t_3 was playing against team t_1 at home (resp. away) in a given round, then in the neighbor solution it will play against team t_2 in the same round at home (resp. away). To ensure that a feasible schedule is obtained, the stadium assigned to the game between them is reversed in the neighbor solution. There are $n \cdot (n-1)/2$ possible moves in this neighborhood and all of them correspond to feasible schedules.

Solutions are represented as ordered oriented 1-factorizations of K_n. A move in this neighborhood consists in swapping the labels of a given pair of nodes in all 1-factors of an ordered oriented 1-factorization.

5.3 Partial round swap (PRS) neighborhood

Let t_1, t_2, t_3, and t_4 be four teams and r_1 and r_2 two rounds such that games $\{t_1, t_3\}$ and $\{t_2, t_4\}$ take place in round r_1 and games $\{t_1, t_4\}$ and $\{t_2, t_3\}$ in round r_2. A move in this neighborhood consists in swapping the rounds in which these games take place. Games $\{t_1, t_4\}$ and $\{t_2, t_3\}$ are set to round r_1 and games $\{t_1, t_3\}$ and $\{t_2, t_4\}$ to round r_2. All 16 combinations of possible stadium assignments are evaluated and the best feasible combination is retained.

As before, solutions are represented as ordered oriented 1-factorizations of K_n. Let A and B be two oriented 1-factors for which there are two nodes t_1 and t_2 connected to the same nodes t_3 and t_4 in A and B. Then, the nodes connected to t_1 and t_2 in A and B may be swapped. A move in this neighborhood is completed by assigning orientations to the edges of K_n defined by the extremities of the arcs involved in the move. Figure 4 illustrates a move in the PRS neighborhood for $n = 8$ teams.

We notice that there are no moves in the PRS neighborhood of the solutions generated by the polygon method for $n = 8, 12, 14, 16, 20, 24$.

5.4 Game rotation (GR) neighborhood

Game rotation consists in enforcing an arbitrary game to be played in an arbitrary round, followed by the appropriate modifications to avoid teams playing more than one game in the same round. The modifications that have to be applied to the current solution give rise to an ejection chain move. Ejection chains are based on the notion of generating compound sequences of moves by linked steps in which changes in selected elements cause other elements to be ejected from their current state, position, or value assignment, see Glover [16, 17]. The move is finalized by the application of a fast tabu search algorithm that throws away possible infeasibilities in the sequence of home and away games generated by the ejection chain move.

Once again, solutions are represented as ordered oriented 1-factorizations of K_n. We first remove all orientations from the current ordered oriented 1-factorization, obtaining simply an ordered 1-factorization. A game rotation ejection chain move starts by inserting an edge (t_1, t_2) into a given factor. We assume

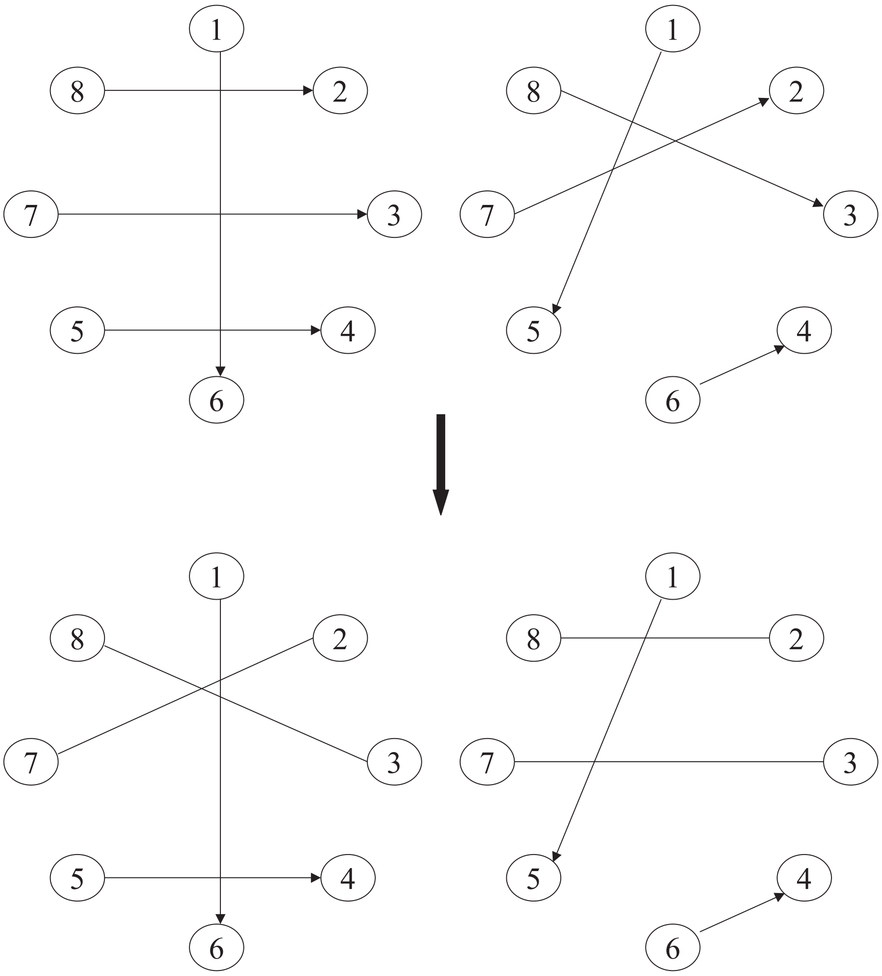

Fig. 4. Move in neighborhood PRS for $n = 8$: opponents of teams 2 and 3 in two different rounds are swapped. Stadium assignments for the games affected by the move are still to be performed.

that edges (t_1, t_3) and (t_2, t_4) belong to this factor. After the insertion of edge (t_1, t_2) both nodes t_1 and t_2 have their degrees equal to 2. In consequence, edges (t_1, t_3) and (t_2, t_4) have to be removed from this 1-factor. Now, both nodes t_3 and t_4 have their degrees equal to 0. Then, edge (t_3, t_4) is inserted and we recover a 1-factor. At this point, edges (t_1, t_2) and (t_3, t_4) appear twice in the 1-factorization under construction. Analogously, edges (t_1, t_3) and (t_2, t_4) do not appear in this 1-factorization. The procedure continues recursively, by reinserting edge (t_1, t_3) into the factor to where edge (t_3, t_4) belonged in the original ordered 1-factorization. The procedure stops once a new ordered 1-factorization is obtained after the reinsertion of edge (t_2, t_4) and the elimination of edge (t_1, t_2) duplicated in the first step of the ejection chain.

After an ejection chain move is performed, all edges corresponding to games that have not been affected by this move receive their original orientations. The orientation of each edge affected by the ejection chain move is originally set at random. If the orientation set for one of the latter edges makes the schedule infeasible, it is reversed. If the schedule remains infeasible after theses changes, we apply a tabu search heuristic aiming to minimize the total number of infeasibilities. For each team, we count an infeasibility for each additional consecutive home game or each additional consecutive away game beyond three. The tabu search heuristic makes use exclusively of the home-away swap neighborhood and stops with a feasible schedule. This procedure runs very quickly, since the number of infeasibilities usually observed after the ejection chain move is very small.

Only neighborhoods PRS and GR are able to change the structure of the 1-factorizations built by the polygon method. The GR neighborhood plays a very important role in the search of good solutions for the mTTP. Ejection chain moves are able to find solutions that are not reachable through other simpler neighborhoods. In particular, we notice that moves in the PRS neighborhood may appear after an ejection chain move is performed, in situations where none existed before. If ejection chains are not used, one may quickly get stuck at schedules with the structure of the solutions built by the polygon method.

6 Local search

Four neighborhood structures were defined in the previous section. The first three and simpler neighborhoods HAS, TS, and PRS are explored by local search. The GR ejection chain neighborhood will be explored only as a diversification move performed less frequently by the heuristic described in the next section, due to the high computation costs associated with its investigation.

We use a first-improving strategy similar to the VND procedure [18, 23] to implement the local search algorithm. TS is the first neighborhood explored. Once a local optimum with respect to this neighborhood is found, a quick local search using the HAS neighborhood is performed in search of a better stadium assignment. Next, the PRS neighborhood is investigated. As before, once a local optimum with respect to this neighborhood is found, the algorithm performs a quick local search using the HAS neighborhood in search of a better stadium assignment. This scheme is repeated until a local optimum with respect to all three neighborhoods is found. Neighbor solutions are investigated at random within each neighborhood.

7 Extended GRASP with ILS

The GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic [13, 14, 25] is a multi-start or iterative process, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. An extensive survey of the literature is given in [15].

The ILS (Iterated Local Search) metaheuristic [21, 22] starts from a locally optimal feasible solution. A random perturbation is applied to the current solution and followed by local search. If the local optimum obtained after these steps satisfies some acceptance criterion, then it is accepted as the new current solution, otherwise the latter does not change. The best solution is eventually updated and the above steps are repeated until some stopping criterion is met.

We propose a hybridization of the GRASP and ILS metaheuristics into an effective hybrid heuristic for the mTTP. Basically, we substitute the local search phase of GRASP by an ILS procedure. The pseudo-code in Algorithm 1 summarizes the main steps of the `GRILS-mTTP` heuristic for finding approximate solutions for the mirrored traveling tournament problem.

```
Procedure GRILS-mTTP();
Data   : MaxIter
Result : Solution S*
for i = 1, ..., MaxIter do
    S ← BuildGreedyRandomizedSolution();
    S̲, S ← LocalSearch(S);
    repeat
        S' ← Perturbation(S);
        S' ← LocalSearch(S');
        S ← AcceptanceCriterion(S, S');
        S* ← UpdateGlobalBestSolution(S, S*);
        S̲ ← UpdateIterationBestSolution(S, S̲);
    until ReinitializationCriterion;
end
```

Algorithm 1: Pseudo-code of a GRASP with ILS heuristic for minimization.

The outer *for* loop in algorithm 1 performs `MaxIter` GRASP iterations. Each iteration starts by the construction of an initial solution S by algorithm `BuildGreedyRandomizedSolution`, which implements the three-stage construction procedure described in Section 4. The greedy criterion used in the second stage (see Section 4.2) is randomized by applying $n/8$ swaps in the order in which the real teams would be assigned to the abstract teams.

Next, algorithm `LocalSearch` is applied to the initial solution built in the previous step, returning a new current solution S. This solution is also used to initialize the best solution $\underline{S}$ in the current GRASP iteration. The VND-like local search procedure described in Section 6 is used to implement the `LocalSearch` local search algorithm.

The inner *repeat* loop starts by applying a perturbation to the current solution S using algorithm `Perturbation` and obtaining a new solution S'. This algorithm implements a randomly generated move within the GR ejection chain neighborhood described in Section 5.4. Algorithm `LocalSearch` is applied to S'.

The new solution S' is accepted or not as the new current solution, depending on the result of an acceptance criterion implemented by algorithm `AcceptanceCriterion` using a threshold parameter β. This parameter is initialized with 0.001 at the beginning of each GRASP iteration and reinitialized with this same value every time the current solution changes. Solution S' is accepted as the new current solution if its cost is lower than $(1 + \beta)$ times the cost of the current solution S. If the current solution does not change after a fixed number of iterations (we used $12n$ in our implementation), the value of β is doubled (i.e., the acceptance criterion is relaxed).

The best overall solution S^* and the best solution in the current GRASP iteration are eventually updated and a new cycle starts with the perturbation of the current solution, until a reinitialization criterion implemented by algorithm `ReinitializationCriterion` is met. In this case, a new GRASP iteration starts if 50 deteriorating moves have been accepted since the last time the best solution found in this GRASP iteration was updated. Reinitialization occurs if too many perturbations followed by local search are performed without improving the best solution in the current GRASP iteration. We notice that a GRASP iteration is not interrupted if the current solution S is still being improved.

8 Computational results

The algorithms described in this paper were coded in `C++` and compiled with version 2.96 of the `gcc` compiler with the optimization flag `-O3`. The experiments were performed on a Pentium IV machine with a 2.0 GHz clock and 512 Mbytes of RAM memory.

8.1 Test problems

Two classes of test problems were originally proposed and described by Easton et al. [11].

The first class is formed by circle instances, artificially generated for testing if traveling tournament problems are easier when the associated traveling salesman problem instances are trivial. The name circ$n$ is used to denote a circle instance with $4 \leq n \leq 20$ teams. Nodes are placed at equal unit distances along a circumference and are labeled $0, 1, \ldots, n-1$. There is an edge between nodes i and $i+1 \mod n$, for $i = 0, 1, \ldots, n-1$. The distance between nodes i and j (with $i > j$) is given by the length of the shortest path between them and is equal to the minimum of $i - j$ and $j - i + n$. National League instances were generated using the distances between the home cities of subsets of teams playing in the National League of the MLB. We denote by nl$n$ a national league instance with $4 \leq n \leq 16$. All these instances are available at `http://mat.gsia.cmu.edu/TOURN/`. We discarded the smaller instances with $n = 4$ and $n = 6$ in our experiments.

Furthermore, we also generated a new real-life instance named br2003.24 and available at `http://www.esportemax.org/`, defined by the home cities of the 24 teams playing in the main division of the 2003 edition of the Brazilian soccer championship.

8.2 Numerical results

We performed three computational experiments, concerning the constructive heuristic described in Section 4 and the GRILS-mTTP extended GRASP with ILS heuristic.

In the first experiment, the randomized version of the constructive heuristic was run 1000 times with different seeds for each instance. We report the numerical results in Table 1. For each instance, we first report the worst, average, and best solution value found over the 1000 runs. These values are followed by the relative gap in percent between the values of the best solution found by the constructive heuristic and the best known solution for the non-mirrored instance at the time of writing. We also give the total computation time in seconds over all runs.

The constructive heuristic is very fast. If we consider e.g. the nl16 instance, as many as 1000 runs can be performed and different schedules generated in approximately one second. The average gap with respect to the best known solutions for the non-mirrored instances at the time of writing is 17.1%. The constructive heuristic runs in computation times which are smaller by several orders of magnitude with regard to those observed for other heuristics for the non-mirrored problem. The solutions quickly obtained by our constructive procedure are often even better than those found after a few days of computation time by the ant colony with backtrack and local search heuristic of Crauwels and Van Oudheusden [6].

In the second experiment, the GRILS-mTTP heuristic was applied only once to each test problem. The maximum computation time of the extended GRASP with ILS heuristic was set at 15 minutes. For each instance, we report in Table 2 the value of the best known solution for the non-mirrored instance at the time

Instance	worst	average	best	gap (%)	time (secs.)
circ8	214	180	156	18.2	0.16
circ10	390	344	306	20.5	0.27
circ12	652	573	486	15.7	0.45
circ14	1012	892	748	9.7	0.67
circ16	1508	1329	1138	16.6	0.94
circ18	2086	1880	1584	11.5	1.26
circ20	2818	2532	2234	17.1	1.67
nl8	59485	50478	44902	13.0	0.18
nl10	90530	80103	71092	19.3	0.30
nl12	167414	146365	127534	13.6	0.48
nl14	293675	266216	241361	27.0	0.71
nl16	436137	382032	329990	23.5	1.01
br2003.24	827611	741369	628930	—	2.63

Table 1. Computational results: constructive heuristic (1000 runs).

of writing and the value of the solution found by the GRILS-mTTP heuristic. These values are followed by the relative gap in percent between the values of the best solution found by the new heuristic and the best known solution for the non-mirrored instance at the time of writing. We also give the iteration counter when the best solution was obtained by the GRILS-mTTP heuristic and the corresponding computation time.

The GRILS-mTTP heuristic is very robust. The largest gap observed between the value of the mirrored solution found by the heuristic and the best known solution value for the non-mirrored problem at the time of writing was 9.5%. In particular, the new heuristic found mirrored solutions which improved the best known solution values for the less constrained non-mirrored instances circ18 and circ20 by respectively 3.9% and 1.4%.

The computation times compare favorably with those observed for other heuristics. The computation time to find the best solution value reported by Anagnostopoulos et al. [1] e.g. for the non-mirrored instance nl14 amounts to 418358.2 seconds (i.e., almost five days) on an AMD Athlon machine at 1544 MHz, while the time given for our heuristic was only 15 minutes on a similar machine (i.e., approximately 0.2%).

Concerning the real-life instance br2003.24, we notice that the total distance traveled in the official schedule used in the 2003 edition of the Brazilian soccer championship amounted to 1,048,134 kilometers. The value of the solution found by the GRILS-mTTP heuristic was only 506,433 kilometers, corresponding to a reduction of 52% in the total distance traveled.

In the final experiment, we isolated and quantified the additional improvement obtained through the introduction of the GR ejection chain neighborhood. We implemented a pure GRASP heuristic using the same local search procedure

Instance	non-mirrored	mirrored	gap (%)	iteration	time (secs.)
circ8	132	140	6.1	5	1.4
circ10	254	276	8.7	289	276.0
circ12	420	456	8.6	7	8.5
circ14	682	714	4.7	1	1.1
circ16	976	1004	2.9	13	115.3
circ18	1420	(*) 1364	-3.9	19	284.2
circ20	1908	(*) 1882	-1.4	40	578.3
nl8	39721	41928	5.6	1	0.7
nl10	59583	63832	7.1	471	643.9
nl12	112298	120655	7.4	22	24.0
nl14	190056	208086	9.5	30	69.9
nl16	267194	285614	6.9	31	514.2
br2003.24	—	506433	—	17	742.5

(*) New best solution value for the non-mirrored instance

Table 2. Computational results: `GRILS-mTTP` heuristic.

used by the `GRILS-mTTP` heuristic. This local search procedure does not make use of the ejection chain neighborhood. As before, the same computation time of 15 minutes is given to both algorithms. For each instance, we report in Table 3 the value of the best solution and the time to the best solution (in seconds) for each heuristic.

The contribution of the ejection chain neighborhood is clear. The solutions found by the full `GRILS-mTTP` heuristic are systematically better than or equal to those found by the pure GRASP. The average reduction in the solution value observed with the use of the `GRILS-mTTP` heuristic with respect to the pure GRASP algorithm is 4.2%. The computation times to the best solutions found by each algorithm also compare favorably for the `GRILS-mTTP` heuristic.

9 Concluding remarks

In this work, we investigated the yet unexplored mirrored version of the traveling tournament problem (mTTP). This tournament structure is very common in Latin American tournaments.

The first contribution of this paper is a very fast constructive heuristic for the mirrored traveling tournament problem. It runs in approximately 1/1000 of a second on a Pentium IV machine with a 2.0 GHz clock for the largest benchmark instances. The average gap with respect to the best known solutions for the non-mirrored instances at the time of writing is 17.1%. The constructive heuristic runs in computation times which are smaller by several orders of magnitude with respect to those of other heuristics. The solutions quickly obtained are

Instance	Pure GRASP		GRILS-mTTP	
	value	time (s)	value	time (s)
circ8	154	1.6	140	1.4
circ10	282	755.7	276	276.0
circ12	458	145.0	456	8.5
circ14	714	746.3	714	1.1
circ16	1072	107.9	1004	115.3
circ18	1446	563.3	1364	284.2
circ20	2008	620.2	1882	578.3
nl8	43742	8.8	41928	0.7
nl10	64515	360.4	63832	643.9
nl12	121477	72.3	120655	24.0
nl14	215358	637.7	208086	69.9
nl16	297004	662.3	285614	514.2
br2003.24	576129	660.9	506433	742.5

Table 3. Computational results: pure GRASP vs. GRILS-mTTP heuristic.

sometimes even better than those found in a few days of computation time by implementations of metaheuristics such as ant colonies.

This constructive heuristic provides a quite effective tool for building good initial solutions for more sophisticated algorithms.

We have also investigated different neighborhood structures for local search procedures. Three simple neighborhoods and an ejection chain neighborhood were described. The latter plays a very important role in the search of good solutions for the mTTP, since the ejection chain moves are able to find solutions that are not reachable through other neighborhoods. Admissible moves in other neighborhoods may appear after an ejection chain move is performed, in situations where none existed before. If ejection chains are not used, algorithms based on local improvement strategies may easily get stuck at schedules with certain structures that cannot be further improved. The computational results have showed the importance of the use of this neighborhood.

Finally, we proposed an extended GRASP with ILS heuristic for the mirrored traveling tournament problem. The GRILS-mTTP heuristic obtained very good solutions for the mTTP. For two benchmark instances, this new algorithm obtained solutions that are even better than the previously best known solutions for the less constrained non-mirrored instances. The computation times compare favorably with those observed for other heuristics. The GRILS-mTTP heuristic is much faster than a simulated annealing procedure which is currently the best heuristic for the non-mirrored problem. As an example, our heuristic is approximately 500 times faster for instance nl14.

The new GRILS-mTTP heuristic can be used by decision makers interested in quickly computing and comparing good different tournament schedules in real life applications.

It is interesting to notice that the total distances traveled in the best known solutions for the mTTP instances are not much larger than those for the associated non-mirrored instances. Furthermore, for two benchmark instances we were able to find mirrored solutions that were better than the best known solutions for the non-mirrored instances at the time of writing.

We have also generated and made available a new real-life instance, defined by the home cities of the 24 teams playing in the main division of the 2003 edition of the Brazilian soccer championship. The GRILS-mTTP heuristic produced a mirrored solution for this instance for which the total distance traveled amounts to almost 50% of that of the schedule actually implemented by the organizers. This reduction amounts to up to 1.5 million dollars in terms of potential savings in airfares.

References

1. A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados. A simulated annealing approach to the traveling tournament problem. In *Proceedings of CPAIOR'03*, 2003.
2. B.C. Ball and D.B. Webster. Optimal schedules for even-numbered team athletic conferences. *AIIE Transactions*, 9:161–169, 1997.
3. A. Cardemil. Optimización de fixtures deportivos: Estado del arte y un algoritmo tabu search para el traveling tournament problem. Master's thesis, Universidad de Buenos Aires, Departamento de Computación, Buenos Aires, 2002.
4. D. Costa. An evolutionary tabu search algorithm and the NHL scheduling problem. *INFOR*, 33:161–178, 1995.
5. H. Crauwels and D. Van Oudheusden. A generate-and-test heuristic inspired by ant colony optimization for the traveling tournament problem. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, pages 314–315, Gent, 2002.
6. H. Crauwels and D. Van Oudheusden. Ant colony optimization and local improvement. In *Workshop of Real-Life Applications of Metaheuritics*, Antwerp, 2003.
7. D. de Werra. Geography, games and graphs. *Discrete Applied Mathematics*, 2:327–337, 1980.
8. D. de Werra. Scheduling in sports. In P. Hansen, editor, *Studies on Graphs and Discrete Programming*, volume 11 of *Annals of Discrete Mathematics*, pages 381–395. North-Holland, 1981.
9. J. Dinitz, E. Lamken, and W.D. Wallis. Scheduling a tournament. In C.J. Colbourn and J. Dinitz, editors, *Handbook of Combinatorial Designs*, pages 578–584. CRC Press, 1995.
10. K. Easton, G.L. Nemhauser, and M. Trick. Sports scheduling. In J.T. Leung, editor, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, pages 52.1–52.19. CRC Press, 2004.

11. K. Easton, G.L. Nemhauser, and M.A. Trick. The traveling tournament problem: Description and benchmarks. In T. Walsh, editor, *Principles and Practice of Constraint Programming*, volume 2239 of *Lecture Notes in Computer Science*, pages 580–584. Springer, 2001.
12. S. Elmohamed, P. Coddington, and G. Fox. A comparison of annealing techniques for academic course scheduling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference for the Practice and Theory of Automated Timetabling*, volume 1408 of *Lecture Notes in Computer Science*, pages 92–114. Springer, 1998.
13. T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.
14. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
15. P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.
16. F. Glover. New ejection chain and alternating path methods for traveling salesman problems. In R. Sharda O. Balci and S. Zenios, editors, *Computer Science and Operations Research: New Developments in Their Interfaces*, pages 449–509. Elsevier, 1992.
17. F. Glover. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics*, 65:223–253, 1996.
18. P. Hansen and N. Mladenović. Developments of variable neighborhood search. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, 2002.
19. M. Henz. Scheduling a major college basketball conference revisited. *Operations Research*, 49:163–168, 2001.
20. M. Henz, T. Müller, and S. Thiel. Global constraints for round robin tournament scheduling. *European Journal of Operational Research*, 153:92–101, 2004.
21. O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.
22. O. Martin, S.W. Otto, and E.W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
23. N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.
24. G.L. Nemhauser and M.A. Trick. Scheduling a major college basketball conference. *Operations Research*, 46:1–8, 1998.
25. M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.
26. R.A. Russell and J.M. Leung. Devising a cost-effective schedule for a baseball league. *Operations Research*, 42:614–625, 1994.
27. J.A.M. Schreuder. Combinatorial aspects of construction of competition Dutch professional football leagues. *Discrete Applied Mathematics*, 35:301–312, 1992.
28. J.M. Thompson. Kicking timetabling problems into touch. *OR Insight*, 12:7–15, 1999.
29. M.A. Trick. Challenge traveling tournament instances. Online document at http://mat.gsia.cmu.edu/TOURN/, last visited on October 11, 2003.
30. M.B. Wright. Scheduling English cricket umpires. *Journal of the Operational Research Society*, 42:447–452, 1991.

31. J.T. Yang, H.D. Huang, and J.T. Horng. Devising a cost effective basketball scheduling by evolutionary algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1660–1665, Honolulu, 2002.
32. H. Zhang. Generating college conference basketball schedules by a SAT solver. In *Proceedings of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing*, pages 281–291, Cincinnati, 2002.