NEW MODELS FOR THE MIRRORED TRAVELING TOURNAMENT PROBLEM

Marco Antonio Moreira de Carvalho
Instituto Tecnológico de Aeronáutica – ITA
Praça Marechal Eduardo Gomes, 50, São José dos Campos – SP, Brasil
mamc@ita.br

Luiz Antonio Nogueira Lorena
Instituto Nacional de Pesquisas Espaciais – INPE
Laboratório Associado de Computação e Matemática Aplicada – LAC
Av. dos Astronautas, 1.758, São José dos Campos – SP, Brasil
lorena@lac.inpe.br

ABSTRACT

The Mirrored Traveling Tournament Problem (mTTP) is a challenging combinatorial optimization problem which consists in generating a timetable for sports tournaments with two half series, what is equivalent to a double round-robin timetable problem. The considered objectives are to minimize the distance traveled by teams during the tournament and the new objective of minimizing the longest distance traveled, named *MinMaxTTP*. It is proposed an integer programming formulation to the mTTP and two models with dynamic constraints to its solution. Both models are based on the detection of independent sets on conflict graphs, whose use has not been reported in the literature about the problem. Real data benchmarks from a baseball tournament are used in the experiments carried out.

KEYWORDS: Traveling Tournament Problem, Sports Scheduling, Integer Programming.

RESUMO

O *Mirrored Traveling Tournament Problem* (mTTP) é um desafiador problema de otimização combinatória que consiste em realizar a programação de jogos de torneios esportivos compostos por dois turnos, equivalente a um problema de escalonamento *double round-robin*. Os objetivos considerados neste trabalho são o de minimizar a distância percorrida pelas equipes durante a realização dos jogos e o novo objetivo de minimizar a maior distância percorrida, denominado *MinMaxTTP*. Propõe-se a formulação do mTTP como um problema de programação inteira e dois modelos que utilizam restrições dinâmicas para sua solução. Os modelos se baseiam na detecção de conjuntos independentes em grafos de conflitos, cuja utilização ainda não foi registrada na literatura sobre o problema. Experimentos com *benchmarks* da literatura, formados por dados reais de um torneio de beisebol são reportados.

PALAVRAS-CHAVE: Traveling Tournament Problem, Programação de Campeonatos Esportivos, Programação Inteira.

1. Introduction

Due to its importance, the scheduling of sports tournaments (sports scheduling) has an area of research of its own within operational research. In such tournaments, teams from different locations face each other, making necessary trips to perform the matches, which implies in operational costs and can also affect athletes' performance due to the suffered fatigue. Thus, it is of interest to the tournaments organization to consider these aspects during the matches scheduling. In double round-robin tournament type, in which we are concerned, all the teams play against each other twice in two half series, alternating home rights (*i.e*. the responsibility for organizing the match). In this type of tournament, all matches are scheduled in advance, since the composition of matches does not depend on the result of others. For a review of terminology, applications, specific constraints and methods of this area, we refer the reader to Rasmussen and Trick (2008).

The Traveling Tournament Problem (TTP) (Easton, Nemhauser and Trick, 2001) consists of, given a set of n (an even number) teams and the matrix of distances between the team's hometowns, to schedule all matches of a tournament. Among the objectives, it is possible to minimize the distance traveled by the teams or minimize breaks of patterns of alternating home rights. In this work we introduce a new objective to TTP: to minimize the longest distance traveled, which we called *MinMaxTTP*. This problem is considered an operational research "discomfort" (Trick, 2009): though this is an easy problem to be stated, on the other hand it is very hard to solve even for small instances. The tournaments are divided into two half series, these in turn are divided into rounds. The constraints of TTP are:

1. All teams play against all other at each half series;
2. Each team plays only one match per round;
3. Two matches cannot occur consecutively between the same teams (no repeaters); and
4. No team can have more than three consecutive matches at home (*i.e*., as the host) or away (*i.e*., as the visitor).

In the version addressed in this paper, called Mirrored Traveling Tournament Problem (mTTP), the second half has the same sequence of matches in the first, however, the field rights are mirrored (*i.e*., inverted). Thus, constraint 3 is removed and the following constraint is added to the problem:

5. A match and its mirror can not occur in the same half series.

Note that it is not strictly necessary that the teams return to their hometown after each match (except in cases in which constraint 4 applies), resulting in the need to minimize the distance traveled, an objective considered in this approach. Additionally, it is assumed that each team is in its hometown before the first round of the first half and after the last round of the second half. Thus, a solution to the mTTP can be seen as a set of tours to be traveled by teams during the tournament, beginning and ending at their hometowns.

In this work, the mTTP is formulated as an integer programming problem in which the concept of conflict graph is used as the basis for the development of two models. To the best of our knowledge, the use of such concept has not been reported in the literature about the problem.

The remainder of this paper is organized as follows: In the next section, a literature review is performed. Then the new formulation, conceptualization and integer programming models for the mTTP are presented. In Section 4, computational experiments are detailed and results discussed. Finally, conclusions are drawn about the work and references are shown.

2. Literature Review

Although the complexity of TTP has not been defined, one can see that this is a complex problem. Among the approaches described below, some rely on grid computing and parallelism. According to the experiments analyzed, it becomes clear that even with the use of such methods, computing the solution of problems of moderate dimensions still requires days of execution.

Instances often used as benchmarks in the literature about the problem are real data from the National League of Professional Baseball Clubs (*NL*) Super 14 Rugby League (*Super*), National Football League (*NFL*) and the Brazilian Soccer Championship. These instances have 4 to 32 teams, and can be found in http://mat.gsia.cmu.edu/TOURN/. Among the sets cited, the most discussed among the works below is *NL*. In the remainder of this paper instances of this set will be referred to as *NLn*, where *n* denotes the number of teams. The largest instance solved up to date has 10 teams, and to prove optimality it took a week of running time in a grid consisting of 120 processors (Trick, 2009).

Easton, Nemhauser and Trick (2001), define the TTP and present real data instances for use as benchmarks. It is also proposed a method of constraint programming and integer programming. In preliminary experiments reported, instances *NL*4 and *NL*6 were solved, the first in a few seconds and the last in some hours in a computational environment not informed. No greater instance was solved by this method.

The use of Lagrangean relaxation is proposed by Benoist, Laburthe, and Rottembourg (2001), in which a subproblem is created for each team: the restricted traveling salesman problem. The relaxation is used in a collaborative scheme that also involves constraint programming for bounds strengthening and dynamic programming. In their experiments with the benchmarks, the approach was able to prove optimality for instances *NL*4 and *NL*6, requiring 24 hours of execution for the latter. For the *NL*10 instance, after 24 hours of running the gap obtained was 17.7%. The lower bound given by Lagrangean shown to be effective.

Easton, Nemhauser and Trick (2003) again address the TTP, this time through a parallelized implementation of the branch-and-price method, where the master problem is solved by column generation and the subproblems are solved by constraint programming. The instance *NL*8 was solved in just over four days, using 20 processors, however, the constraint of no repeaters was not considered. It was proposed the independent lower bound, a lower bound that considers a traveling salesman problem for each team. Lower bounds for cases with circular distances (adding aspects of the traveling salesman problem to the TTP) are proposed by Fujiwara *et al*. (2003).

A hybrid heuristic of GRASP (Greedy Randomized Adaptive Search Procedure) and ILS (Iterated Local Search) is presented by Ribeiro and Urrutia (2007). The results showed that in some cases the mirrored solution found was better than the best known not mirrored solution at that moment, for circular distance instances. The average gap in relation to the not mirrored solutions was 17.1%. Another important aspect is the execution time limited to 15 minutes, much lower than that reported by other methods. Four different parallelization strategies on a computational grid for this heuristic are proposed by Araújo *et al*. (2007). Using the strategy of maintaining a poll of solutions, within hours of execution it was possible to obtain better solutions than those previously reported for the mTTP for instances with circular distances and for *NL*16. A grid with 82 computers distributed in 4 clusters was used.

Urrutia, Ribeiro and Melo (2007), improve the independent lower bound considering the optimal solutions if the distances were constant between all pairs of cities. Good reduction rates were obtained for both TTP and mTTP. For instances of the *NL* set, only for instance *NL*8 no reduction in the existing bound for the mTTP was achieved. Bounds not available for the *NFL* set were also obtained.

Cheung (2008), presents a two stage method, based on 1-factorizations on the complete graphs representation of the mTTP. With this method, it was possible to solve the *NL*8 instance, using a single computer without parallelization after 3.7 days of execution. Although it was subsequently published, the integer programming formulation solved by branch-and-price proposed by Irnich (2009) is considered the first method to solve the instance *NL*8 to proven optimality, after 12 hours of execution in a single computer without parallelism. Additionally, several lower bounds for benchmarks were improved, among them, *NL*10 and *NL*12.

The Benders decomposition approach is used to obtain lower bounds for the mTTP in (Cheung, 2009), which additionally proposes a mixed integer linear programming model. Better bounds for the instances *NL*10 to *NL*16 and *NFL*16 to *NFL*24 were found, requiring 3.5 to 22.5 days of execution.

An annotated bibliography on sports scheduling detailing definitions, methodologies and applications is provided by Kendall *et al*. (2010). Among the problems addressed is the mTTP.

3. New Formulation and Models

Two new integer programming models for mTTP are presented. Both are based on independent sets detection in specific graphs that have characteristics of conflicts between its variables, however, they have different objectives. Such graphs, named conflict graphs, and the proposed models are presented below. Additionally, the ways in which distances are calculated in the models are shown, each serving a specific need.

3.1. Conflict Graphs

In conflict graphs, vertices represent instances that are adjacent to other only if they are conflicting. In the case of TTP, vertices represent matches, which are adjacent if they cannot be scheduled for the same round because they have teams in common. Applications of conflict graphs in other problems include Bin Packing with Conflicting Objects (Muritiba *et al*., 2009), Point-Feature Cartographic Label Placement, Manufacturer’s Pallet Loading, Woodpulp Stowage and Daily Photograph Scheduling of an Earth Observation Satellite (Ribeiro, 2007).

For example, consider four teams: SPO, FLA, CRU and GRE. All crossings totalize 12 ordered pairs (home, away) that define the matches. The conflict graph for this example is shown in Figure 1.

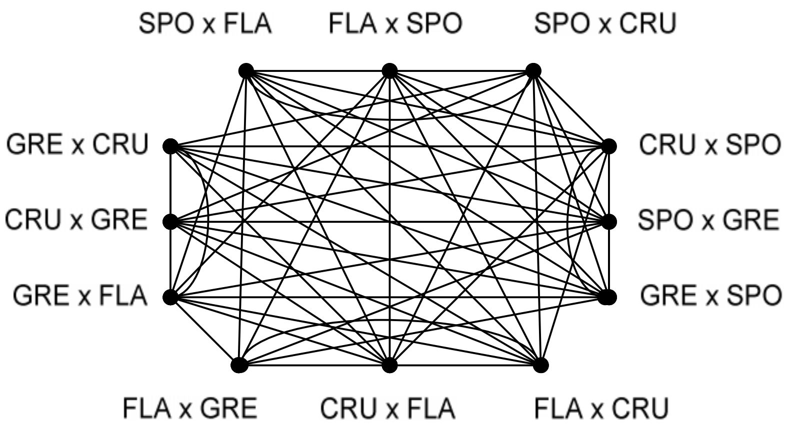

Figure 1 – Four teams conflict graph.

According to the graph, matches SPO×FLA and GRE×CRU could be scheduled to the same round, since there is no conflict between these matches. The structure of the conflict graph, therefore, suggests that independent sets (that in this case have a fixed size: $n/2$) represent matches that can be scheduled simultaneously for the same round. The mathematical models presented in the sections below are based on the detection of independent sets in conflict graphs, a new approach for the mTTP.

3.2 Problem Data and Variables

Both models use the same data set and share most of the variables. The problem data are:

- n: number of teams;
- p: number of rounds per half;
- m: number of possible ordered pairs of teams that make up a match;
- g_i: distance between the cities of the teams participating in the match i ($i = 1, ..., m$);
- $d_{i,k}$: distance to be traveled between matches i and k ((i, k) ∈ conflict graph); and
- $h_{i,k}$: distance traveled between the match i and k of different halves (i is a final round match of the first half and k is a first round match of the second half, (i, k) ∈ conflict graph);

The models variables are:

- $x_{i,j}$: match i takes place in round j ($i = 1, ..., m$ and $j = 1, ..., p$);
- $y_{i,k}$: match k takes place in the next round of the match i ((i, k) ∈ conflict graph);
- $w_{i,k}$: match i takes place on the last round of the first half and match k occurs in the first round of the second half;
- s_i: initial trip before match i ($i = 1, ..., m$);
- e_i: final trip after match i ($i = 1, ..., m$); and
- z: longest distance traveled during the tournament.

As shown, three different calculations of distances are used. Consider the match scheduling shown in Table 1 for the details below. The half and round of each match are shown, which are also identified by the value in column "id". The last two columns present the teams participating in each match and the city where it will take place.

Table 1 – Example of schedule.

Half	Round	id	Match	City
1st	1	1	SPO × GRE	São Paulo
		2	FLA × CRU	Rio de Janeiro
	2	3	SPO × CRU	São Paulo
		4	FLA × GRE	Rio de Janeiro
	3	5	CRU × GRE	Belo Horizonte
		6	FLA × SPO	Rio de Janeiro
2nd	4	7	GRE × SPO	Porto Alegre
		8	CRU × FLA	Belo Horizonte
	5	9	CRU × SPO	Belo Horizonte
		10	GRE × FLA	Porto Alegre
	6	11	GRE × CRU	Porto Alegre
		12	SPO × FLA	São Paulo

The first distance calculation, denoted by g_i, is the distance between the hometowns of the two teams participating of match i from the first round of the first half or from last round of the second half. For example, according to table 1 above, g_1 is calculated as the distance between Porto Alegre and São Paulo, since the GRE team will travel to São Paulo.

Since it is the mirrored version of the problem, the first half determines the second, and therefore, only the first round variables are handled. However, the distance traveled in the second half must also be taken into consideration, for the integer programming model to represent the problem completely. From this need, a second distance table is created in which, for each pair of matches, the distances between them in the first half and between its mirrors in the second half are summed.

The calculation, $d_{i,k}$ is the distance between matches of two different halves. For example, $d_{3,6}$ is the sum of distances between São Paulo and Rio de Janeiro (the travel of SPO team between matches 3 and 6) and between Belo Horizonte and São Paulo (the travel of SPO team between matches 9 and 12 – mirrors the previous matches). This calculation only occurs when two matches in different halves have one team in common. Otherwise, the distance is zero.

The third calculation, $h_{i,k}$ refers to the distance between matches of the end of the first half and the beginning of the second. For example, $h_{6,7}$ is the distance between Rio de Janeiro and Porto Alegre.

3.3 Model 1

The first model aims to minimize the total distance traveled between the scheduled matches, and is presented below.

$$Min \sum_{(i,k)\in \text{conflict graph}} d_{ik} y_{ik} + \sum_{i=1}^{m} g_i s_i + \sum_{i=1}^{m} g_i e_i + \sum_{(i,k)\in \text{conflict graph}} h_{ik} w_{ik} \quad (1)$$

subject to

$$\sum_{j=1}^{p} (x_{i,j} + x_{i+1,j}) = 1 \qquad i = 1, 3, 5, ..., m\text{-}1 \quad (2)$$

$$x_{i,j} + x_{k,j} \le 1 \qquad (i, k) \in \text{conflict graph},\ j = 1, ..., p \quad (3)$$

$$x_{i,j} + x_{k,j+1} - y_{i,k} \le 1 \qquad (i, k) \in \text{conflict graph},\ j = 1, ..., p\text{-}1 \quad (4)$$

$$x_{i,1} + x_{i+1,1} - s_i = 0 \qquad i = 1, 3, 5, ..., m\text{-}1 \quad (5)$$

$$x_{i,1} + x_{i+1,p} - e_i = 0 \qquad i = 1, 3, 5, ..., m\text{-}1 \quad (6)$$

$$x_{i,1} + x_{k,p} - w_{j,k} \le 1 \qquad (i, k) \in \text{conflict graph},\ j = \text{match } i \text{ mirror},\ i = 1, ..., m \quad (7)$$

$$x_{i,j} \in \{0,1\} \qquad i = 1, ..., m,\ j = 1, ..., p \quad (8)$$

$$y_{i,k} \in \{0,1\} \qquad (i, k) \in \text{conflict graph} \quad (9)$$

The first set of constraints (2) provides that if a match is scheduled, its mirror may not be in the same half either. The independent set detection formulation (3) does not allow conflicting matches to be scheduled for the same round. Constraints (4)-(7) calculate the distances: between rounds (4), before the first round (5), after the last round (6) and between halves (7). Constraints (8) and (9) define that the variables are binary.

This model has $m\times(4\times|E|+p+2)$ variables and $m\times[1.5+p\times(6n\text{-}12)]$ constraints, where $|E|$ is the cardinality of the conflict graph edge set.

3.4 Model 2

Unlike the previous model, the second model has a different objective: to minimize the longest distance traveled, which we named *MinMaxTTP*. To the best of our knowledge, this objective is not addressed in the literature about the TTP. The model is presented below.

$$Min\ \boldsymbol{z} \qquad (10)$$

subject to

$\sum_{j=1}^{p}(x_{i,j}+x_{i+1,j})=1$	i = 1, 3, 5, ..., m-1	(11)
$x_{i,j}+x_{k,j}\leq 1$	(i, k) ∈ conflict graph j = 1, ..., p	(12)
$x_{i,j}+x_{k,j+1}-y_{i,k}\leq 1$	(i, k) ∈ conflict graph j = 1, ..., p-1	(13)
$(m-d_{i,k})y_{i,k}+z\geq m$	(i, k) ∈ conflict graph	(14)
$x_{i,1}+x_{i+1,1}-s_i=0$	i = 1, 3, 5, ..., m-1	(15)
$(m-g_i)s_i+z\geq m$	i = 1, 3, 5, ..., m-1	(16)
$x_{i,p}+x_{i+1,p}-e_i=0$	i = 1, 3, 5, ..., m-1	(17)
$(m-g_i)e_i+z\geq m$	i = 1, 3, 5, ..., m-1	(18)
$x_{i,1}+x_{k,p}-w_{j,k}\leq 1$	(i, k) ∈ conflict graph j = match i mirror i = 1, ..., m	(19)
$(m-h_{i,k})w_{i,k}+z\geq m$	(i, k) ∈ conflict graph	(20)
$x_{i,j}\in\{0,1\}$	i = 1, ..., m j = 1, ..., p	(21)
$y_{i,k}\in\{0,1\}$	(i, k) ∈ conflict graph	(22)
$z\geq 0$		(23)

As in the previous model, constraints (11) and (12) are related to conflicts between matches, and the constraints (13), (15), (17) and (19) calculate the traveled distances. Constraints (14), (16), (18) and (20) determine the longest distance traveled. Constraints (21) and (22) define that the variables are binary and the constraint (23) defines the variable is positive.

This model has m×(4×|E|+p+2)+1 variables and m×[3.5+4×|E|+(n-2)×(6p)]+1 constraints, where again, |E| is the conflict graph edge set cardinality.

3.5 Consecutive Home-Away Matches Dynamic Constraints

In the presented models, the constraint of consecutive home or away matches for each team was not directly considered. This constraint is applied dynamically to the model iteratively until we obtain a feasible solution to the problem.

At each loop, the model solution is analyzed and, if some team has more than three consecutive home or away matches, the variables related to such matches are embedded in a constraint as shown below:

$$x_{a,j} + x_{b,j+1} + x_{c,j+2} + x_{f,j+3} \leq 3 \qquad \begin{array}{l} j = 1, ..., p\text{-}3 \\ a, b, c, f = \text{matches that violate the constraint} \end{array} \qquad (24)$$

Once the constraint is added, the model is solved again and the process repeats iteratively until the solution meets all constraints. This method of applying the constraint prevents the integer programming model from becoming too large, adding only the necessary constraints.

As only variables related to the first half are manipulated (since the second half is mirrored), there still may be a violation of this constraint involving the two halves at the same time. For example, a team has two matches away in the final two rounds of the first half and again two matches away in the first two rounds of the second. It is also necessary to verify such special cases, what is done simulating the mirrors of the first rounds and confronting them with the latest rounds of the first half.

4. Computational Experiments

Experiments were carried out involving some of the instances available in the literature described earlier in Section 2. The computing environment used consists of an Intel Pentium 4 3.20 GHz frequency and 1016 MB of RAM memory under Windows XP Professional Edition operational system. The models were solved using CPLEX 12.

In the first part of the experiment the models were solved directly, and then strategies for accelerating the solution were included. A limit of 48 hours for the execution time was established. This limit is smaller than those seen in other works that address the TTP by exact methods, however, the results reported here are preliminary. Table 2 presents the data obtained by direct solution of the original models by CPLEX. For each model, the results, running times and number of loops required to solve the instances are presented. The last column of the table shows the best results known for each instance.

Table 2 - Original models results.

Instance	Model 1			Model 2			Benchmark
	Result	Runtime (s)	# Loops	Result	Runtime (s)	# Loops	
*NL*4	8276	0.36	0[1]	8429	0.48	0[1]	8276[2]
*NL*6	*	*	*	*	*	*	26588
*NL*8	*	*	*	*	*	*	41928
*NL*10	*	*	*	*	*	*	63832

[1] Not applied. [2] Optimal Solution. [*] Solution not found within time limit.

The direct solution of the models proves to be impossible within the time limit set, since in this condition only the smallest instance could be solved. In both models, such instance, *NL*4, was solved in less than a minute, while for the instance with two more teams, *NL*6, the the 48 hours of execution was exceeded. The fast growth of execution times shows the great difficulty in solving the problem.

In order to accelerate the solve-and-constrain loops, two strategies are proposed. The first aims to interrupt the CPLEX execution after obtaining the first feasible solution, instead of waiting for the complete solution. Table 3 presents the results obtained by applying this strategy.

Table 3 - Results interrupting the execution of CPLEX on the first feasible solution found.

Instance	Model 1			Model 2			Benchmark
	Result	**Runtime (s)**	**# Loops**	**Result**	**Runtime (s)**	**# Loops**	
*NL*4	8413	0.08	0[1]	8413	0.03	0[1]	8276[2]
*NL*6	30555	0.63	4	30555	0.92	4	26588
*NL*8	56599	55.16	29	56599	34.09	29	41928
*NL*10	88556	145.76	208	91219	3458.25	685	63832

[1] Not applied. [2] Optimal Solution.

For the first three instances, the results were similar for the two models, which also required the same number of iterations and generated exactly the same schedule. This fact in models of different objectives indicates that there is a difficulty in moving among different solutions, also evidenced by the fact that a change in one match makes the whole schedule infeasible. For the last instance, model 1 obtained a better solution in less time and fewer iterations. The deterioration of the solutions when compared to the previous approach solutions was expected, since CPLEX was interrupted, however, this deterioration may have been outstanding, given the distance of over 38% when compared to the benchmarks.

The second acceleration strategy controls the interruption of the model solution according to the gap between the current solution and the bound used by CPLEX. Therefore, it is possible to control the solution deterioration level and run time growth. Table 4 presents the results for different gap values.

Table 4 – Results interrupting the CPLEX execution accordingly to the gap.

gap: 95%

Instance	Model 1			Model 2			Benchmark
	Result	**Runtime (s)**	**# Loops**	**Result**	**Runtime (s)**	**# Loops**	
*NL*4	8569	0.05	0[1]	8413	0.03	0[1]	8276[2]
*NL*6	31068	3.59	8	30555	0.95	4	26588
*NL*8	55955	635.08	69	56599	34.08	29	41928
*NL*10	88556	122.86	208	91219	3418.22	685	63832

gap: 90%

Instance	Model 1			Model 2			Benchmark
	Result	**Runtime (s)**	**# Loops**	**Result**	**Runtime (s)**	**# Loops**	
*NL*4	8569	0.03	0[1]	8413	0.03	0[1]	8276[2]
*NL*6	28514	5.23	8	30555	2.27	4	26588
*NL*8	*	*	*	56599	463.36	29	41928
*NL*10	*	*	*	91219	123057.31	685	63832

gap: 50%

Instance	Model 1			Model 2			Benchmark
	Result	**Runtime (s)**	**# Loops**	**Result**	**Runtime (s)**	**# Loops**	
*NL*4	8276	1.00	0[1]	8596	0.47	0[1]	8276[2]

[1] Not applied. [2] Optimal Solution. * Solution not found within time limit.

Even for a small variation of gap values (from 95% to 90%), the execution time grows very fast. For a gap set at 95%, model 1 has the worst solutions, execution times and larger number of iterations compared to the previous strategy, except for instance *NL*10, for which there was only a small change of the runtime. For a gap of 90% the model achieved improved solutions for *NL*6 instance, with little variation in execution time, but was not able to solve larger instances within the time limit.

Model 2 indicates again a great difficulty in moving among solutions, obtaining a single solution both when the gap is set at 95% or 90%, with a very high difference in execution times.

Lowering the gap to 50% it is possible to solve only the smallest instance in both models, reaching its optimum value in a second for model 1 and a worse value for the model 2. However, the objective of the latter model is to minimize the longest distance traveled, the introduced *MinMaxTTP*, which in fact occurs for this instance, but affects the total distance traveled.

The comparison of the number of dynamic constraints added to each model is performed in Table 5. For each model, the values are presented according to the CPLEX interruption strategy. The *NL*4 instance does not appear in the table, since it has 4 teams and no possible solution can violate the constraint of three home or away consecutive matches.

Table 5 – Number of dynamic constraints added to the models per instance

Instance	Model 1			Model 2		
	Feasibility	Gap 95%	Gap 90%	Feasibility	Gap 95%	Gap 90%
*NL*6	16	35	39	16	16	16
*NL*8	151	475	*	151	151	151
*NL*10	1658	1658	*	5190	5190	5190

[*] Solution not found within time limit.

Model 1 adds a different number of dynamic constraints in accordance with the strategy used (except for instance *NL*10). This number increases as the gap is reduced, indicating that different solutions are visited, requiring greater effort to obtain better solutions. Conversely, model 2 remains constant the number of constraints added, suggesting that the same solutions are generated, regardless of the strategy used.

Comparing the two models, one can see that there is an alternation in obtaining the best results and execution times according to the context used, so that we cannot determine the dominance of one over the other.

According to the data presented, the great conflict between solution quality and run times is made evident, with little difference between the different criteria established for speeding up execution, in addition to moving between solutions prove to be reduced.

5. Concluding Remarks

The Traveling Tournament Problem (TTP) has practical application in the preparation of sports tournaments timetables in their various modalities, an investigation area of its own in the field of operational research. In relation to the classical problems of the field, the TTP was defined recently, and attracts attention for being a difficult problem to solve, although it can be easily stated. The objectives vary from minimizing the distance traveled by the teams during the tournament and minimizing breaks on travel patterns and matches as host. The variation addressed in this paper focuses on cases where the tournaments are divided into two similar half series, where the same matches occur, but inverted in relation to the host team. This variant is called the Mirrored Traveling Tournament Problem (mTTP). The objectives considered are to minimize the distance traveled by the teams and a new objective proposed: to minimize the longest distance traveled.

An integer programming formulation for mTTP based on the detection of independent sets in conflict graphs where the vertices represent the possible matches between teams and adjacencies indicate that two matches cannot be scheduled for the same round was presented. To the best of our knowledge, this concept had not been reported in works related to the Traveling Tournament Problem.

Two models were proposed, which share the same conceptual basis, however, they have different objectives: the first seeks to minimize the sum of the distances traveled, and the second seeks to minimize the longest distance traveled, which we called *MinMaxTTP*. The latter, not found in other studies. Both models use dynamic constraints to control the excess of consecutive home or away matches, a constraint of the problem. Thus, the solution was performed iteratively:

the problem is solved, dynamically constrained and solved again until we obtain a final feasible solution.

In preliminary computational experiments involving benchmarks of the literature, the original models were solved directly by the solver used, and later, strategies to accelerate the iteractive process were included. The first strategy was to interrupt the solver after obtaining a feasible solution to the model used, and the second was to stop the solver according to the gap between the current solution and the expected solution. The data collected during the experiments indicate the difficulty of addressing the problem, ocurring little difference in results obtained by different solution strategies. The execution time grows very quickly while the sizes of the instances or the quality of the solutions grow slowly.

Future work involves the inclusion of cuts in the proposed models to accelerate the solution and experiments with time limit of execution similar to those found in the literature, as well as implementation of metaheuristics for supplementary results. Bounds based on a Lagrangian relaxation-type with decomposition into clusters will also be subject of future research. Finally, the fact that the objective of minimizing the longest distance traveled (here called *MinMaxTTP*) is not addressed in the literature can also be explored.

6. Acknowledgements

This work was funded by the State of São Paulo Research Foundation – FAPESP, process 2009/51831-9 (first author), and partially funded by the National Counsel of Technological and Scientific Development – CNPq, processes 471837/2008-3 and 300692 / 2009-9 (second author).

References

Araújo, A., Boeres, M. C., Rebello, V. E., Ribeiro, C. C. (2007). Exploring grid implementations of parallel cooperative metaheuristics: a case study for the mirrored traveling tournament problem. In: Doerner K., Gendreau M., Greistorfer P., Gutjahr W., Hartl R., editors. *Metaheuristics: progress in complex systems optimization*. Berlin: Springer; 297–322.

Benoist, T., Laburthe, F., Rottembourg, B. (2001). Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In: *Proceedings of the third international workshop on integration of AI and OR techniques in constraint programming for combinatorial optimization problems*, 15–26.

Cheung, K. K. H. (2008). Solving mirrored traveling tournament problem benchmark instances with eight teams. *Discrete Optimization*, 5:138–43.

Cheung, K. K. H. (2009). A Benders approach for computing lower bounds for the mirrored traveling tournament problem. *Discrete Optimization*, 6:189–196.

Easton, K., Nemhauser, G. L., Trick, M. A. (2001). The travelling tournament problem: description and benchmarks. In: Walsh, T., editor. *Principles and practice of constraint programming*. Lecture notes in computer science, vol. 2239. Berlin: Springer, 580–5.

Easton, K., Nemhauser, G. L., Trick, M. A. (2003). Solving the travelling tournament problem: a combined integer programming and constraint programming approach. In: Burke, E., de Causmaecker, P., editors. *The 4th international conference on the practice and theory of automated timetabling*. Lecture notes in computer science, vol. 2740. Berlin: Springer, 100–9.

Fujiwara, N., Imahori, S., Matsui, T., Miyashiro, R. (2007). Constructive algorithms for the constant distance traveling tournament problem. In: Burke, E. K., Rudová, H., editors. *Practice and theory of automated timetabling VI*. Lecture notes in computer science, vol. 3867. Berlin: Springer, 135–46.

Irnich, S. A new branch-and-price algorithm for the traveling tournament problem. *Technical Report*, OR-01-2009, RWTH Aachen; 2009.

Kendall, G., Knust, S., Ribeiro, C. C., Urutia, S. *(2010).* Scheduling in sports: An annotated bibliography, *Computers and Operations Research*, 37, 1-19.
Muritiba, A. E. F., Iori, M., Malaguti, E., Toth, P. (2009). Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, Articles in Advance, 1-15.
Rasmussen, R. V., Trick, M. A. (2008). Round robin scheduling – a survey. *European Journal of Operational Research*, 188, 617-636.
Ribeiro, C. C., Urrutia, S. (2007). Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, (179):775–87.
Ribeiro, G. M. (2007). Relaxação Lagrangeana com divisão em clusters para alguns problemas de otimização modelados em grafos de conflitos. *Tese* (*Doutorado*) – Instituto Nacional de Pesquisas Espaciais. São José dos Campos.
Trick, M. (2009). Michael Trick's Operations Research Blog. Operations Research: *Embarrassments and the Traveling Tournament Problem*. 2009. Disponível em: http://mat.tepper.cmu.edu/blog/?p=1020. Acesso em 13 de Abril de 2010.
Urrutia, S., Ribeiro, C. C., Melo, R. A. (2007). A new lower bound to the traveling tournament problem. In: *Proceedings of the IEEE symposium on computational intelligence in scheduling*, Honolulu. IEEE, 15–8.