Clustering Search Approach for the Traveling Tournament Problem

Fabrício Lacerda Biajoli and 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

Abstract. The *Traveling Tournament Problem* (TTP) is an optimization problem that represents some types of sports timetabling, where the objective is to minimize the total distance traveled by the teams. This work proposes the use of a hybrid heuristic to solve the mirrored TTP (mTTP), called *Clustering Search* (*CS), that consists in detecting supposed promising search areas based on clustering. The validation of the results will be done in benchmark problems available in literature and real benchmark problems, e.g. Brazilian Soccer Championship.

1 Introduction

Scheduling problems in sports has become an important class of optimization problems in recent years. For the applications of Operational Research the management of this sporting activity type is an area very promising and little explored. The professional sport leagues represent one of the largest economical activities around the world. For several sports, e.g. soccer, basketball, football, baseball, hockey, etc, where the teams plays a double round-robin tournament among themselves, where the games are played in different places during some time period, the automating of that schedulings are necessary and very important. Other facts fortify the application of optimization techniques. Teams and leagues do not want to waste their investments in players and structure in consequence of poor scheduling of games; sports leagues represent significant sources of revenue for radio and television world networks; the scheduling interfere directly in the performance of the teams; etc. On the other hand, sport leagues generate extremely challenging optimization problems, that attract attention of the Operational Research communities.

The scheduling problems in sports are known in the literature as Traveling Tournament Problem and it was proposed by Easton et al. [7]. The TTP abstracts the salient features of Major League Baseball (MLB) in the United States and was established to stimulate research in sport scheduling. Since the challenge instances were proposed the TTP has raised significant interest. Several works in different contexts (see e.g. [2],[3],[5],[9],[13],[16],[17],[18]) tackled the problem of tournament scheduling in different leagues and sports, which contains many interesting discussion on sport scheduling. Basically, the schedule of MLB is a

conflict between minimizing travel distances and feasibility constraints on the home/away patterns. A TTP solution is a double round-robin which satisfies sophisticated feasibility constraints (e.g. no more than three away games in a road trip) and minimizes the total travel distances of the teams.

Problems of that nature contain in general many conflicting restrictions to be satisfied and different objectives to accomplish, like minimize the total road trips of the teams during the tournament, one just game per day and per team, accomplishment of certain games in stadiums and in pre-established dates, number of consecutive games played in the team's city and out, etc. To generate good schedulings, satisfying all constraints, is a very hard task. The difficulty of solution of that problem is attributed to the great number of possibilities to be analyzed, e.g., for a competition with 20 teams there are $2,9062 \text{ x } 10^{130}$ possible combinations [4].

This work proposes the application of Clustering Search (*CS) [15] to solve the mirrored version of the TTP, known as Mirrored Traveling Tournament Problem (mTTP) [16]. The *CS is a generalization of the Evolutionary Clustering Search (ECS) proposed in [14]. This paper is organized in six sections, being this the first. In the next section, the Traveling Tournament Problem and your *mirrored* version are described. In Section 3, we describe the basic ideas and conceptual components of *CS. In Section 4, the methodology is detailed, with neighborhoods and the algorithm implemented. The computational results are examined in Section 5 and conclusions are summarized in Section 6.

2 Problem description

The *Traveling Tournament Problem* was first proposed by Easton et al. in [7]. A scheduling to a *double round-robin* (DRR) tournament, played by n teams, where n is a even number, consists in a schedule where each team plays with each other twice, one game in its home and other in your opponent's home. A game between teams T_i and T_j is represented by unordered pair (i, j). That schedule needs $2(n-1)$ rounds to represents all games of the tournament. The input data consists of the number of teams (n), a symmetric matrix D, n x n, where D_{ij} represents the distance between the home cities of the teams T_i and T_j.

The cost of a team is the total distance traveled starting from its home city and return there after the tournament ending. The cost of the solution is the sum of the cost of every team.

The objective is to find a schedule with minimum cost, satisfying the following constraints:

- No more than three consecutive home or away games for any team;
- A game of T_i at T_j's home cannot be followed by the game of T_j at T_i's home;

The *Mirrored Traveling Tournament Problem* (mTTP) proposed by Ribeiro and Urrutia in [16] is a generalization of TTP that represents the common structure in Latin-America tournaments (e.g. Brazilian Soccer Championship). The main

difference is the concept of *mirrored double round-robin* (MDRR). A MDRR is a tournament where each team plays every other once in the $n-1$ rounds, followed by the same games with reversed venues in the last $n-1$ rounds.

The objective is the same of TTP, find a schedule with minimum cost satisfying the same constraints plus an additional constraint: the games played in round R are the same played in round $R+(n-1)$ for $R = 1, 2, ..., n-1$, with reversed venues.

3 Clustering Search

The Clustering Search (*CS) [15] is a generalization of the Evolutionary Clustering Search (ECS) proposed in [14] that employs clustering for detecting promising areas of the search space. It is particularly interesting to find out such areas as soon as possible to change the search strategy over them. An area can be seen as a search subspace defined by a neighborhood relationship in metaheuristic coding space.

In the ECS, a clustering process is executed simultaneously to an evolutionary algorithm, identifying groups of individuals that deserve special interest. In the *CS, the evolutionary algorithm was substituted by distinct metaheuristics.

The *CS attempts to locate promising search areas by framing them by clusters. A cluster can be defined as a tuple $G = \{c;\ r;\ s\}$ where c, r and s are, respectively, the *center* and the *radius* of the area, and a *search strategy* associated to the cluster.

The center is a solution that represents the cluster, identifying the location of the cluster inside of the search space. Initially, the center c is obtained randomly and progressively it tends to slip along really promissing points in the close subspace. The radius r establishes the maximum distance, starting from the center, that a solution can be associated to the cluster.

For example, in combinatorial optimization, r can be defined as the number of movements needed to change a solution into another. The search strategy is a systematic search intensification, in which solutions of a cluster interact among themselves along the clustering process, generating new solutions.

The *CS consists of four conceptually independent components with different attributions: a search metaheuristcs (SM); an iterative clustering (IC); an analyzer module (AM); and a local searcher (LS).

Figure 1 shows the four components and the *CS conceptual design.

The SM component works as a full-time solution generator. The algorithm is executed independently of the remaining components and this must to be able of the continuous generation of solutions directly for the clustering process. Simultaneously, clusters are maintained to represent these solutions. This entire process works like an infinite loop, in which solutions are generated along the iterations.

IC component aims to gather similar solutions into groups, maintaining a representative cluster center for them. To avoid extra computational effort, IC is designed as an online process, in which the clustering is progressively fed by

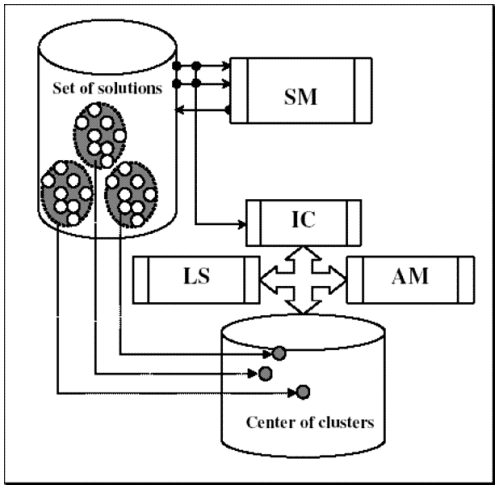

Fig. 1. *CS Components

solutions generated in each iteration of SM. A maximum number of clusters *NC* is a bound value that prevents an unlimited cluster creation. A *distance metric* must be defined, a priori, allowing a similarity measure for the clustering process.

The AM component provides an analysis of each cluster, in regular intervals, indicating a probable promising cluster. A *cluster density*, δ_i, is a measure that indicates the activity level inside the cluster *i*. For simplicity, δ_i counts the number of solutions generated by SM and allocated to the cluster i. Whenever δ_i reaches a certain *threshold*, meaning that some information template becomes predominantly generated by SM, such information cluster must be better investigated to accelerate the convergence process on it. AM is also responsible for the elimination of clusters with lower densities, allowing to create other centers and keeping framed the most active of them. The cluster elimination does not affect the set of solutions in SM. Only the center is considered irrelevant for the process.

At last, the LS component is a local search module that provides the explotation of a supposed promising search area, framed by cluster. This process can happen after AM having discovered a promising cluster and the local search is applied on the center of the cluster. LS can be considered as the particular search strategy *s* associated with the cluster.

4 Methodology

The methodology used to solve the mTTP is based on the use of the Clustering Search (*CS), explained in Section 3, with the metaheuristic Variable Neighborhood Search (VNS) [12] as the SM component of *CS, generating the approach VNS-CS. The next sections describes in full detail the methodology.

4.1 Representation of a solution

The representation of a schedule is a table indicating the opponents of the teams, where each line corresponds to a team and each column corresponds to a round. The opponent's representation is given by the pair (i, j), where i represents the team T_i and j represents the round r_j (e.g., the opponent of the team T_1 in round r_2 is given by $(1, 2)$). If (i, j) is positive, the game takes place at T_i's home, otherwise at T_i's opponent home.

In this work only the $n-1$ first rounds (*first half*) are represented, because the $n-1$ last rounds (*second half*) are the mirror with reversed venues and all alteration in the first half affects the second half (see Figure 2).

	First half					Second half				
T_i/r_k	1	2	3	4	5	6	7	8	9	10
1	-6	4	2	-5	-3	6	-4	-2	5	3
2	-5	3	-1	-4	-6	5	-3	1	4	6
3	4	-2	-5	6	1	-4	2	5	-6	-1
4	-3	-1	6	2	-5	3	1	-6	-2	5
5	2	-6	3	1	4	-2	6	-3	-1	-4
6	1	5	-4	-3	2	-1	-5	4	3	-2

Fig. 2. Representation of a Schedule

4.2 The neighborhood

Five different movements have been defined to compose distinct kinds of neighborhood, named *Home-away swap*, *Team swap*, *Round swap*, *Partial Round swap* and *Games swap*, from a schedule S. The neighborhood of a schedule s is the set of the schedules (feasibles and infeasibles) which can be obtained by applying one of these five types of movements.

Home-away swap This move swaps the home/away roles of a game involving the teams T_i and T_j. The application of the move *Home-away swap* in a solution s obtain a solution s', with a single game swapped, by reversing the game's place. In other words, if team T_i plays at home with T_j (T_j plays away) in s, then T_j plays at home and T_i plays away in s'.

Team swap This move swaps the schedule of two teams, T_i and T_j. Only the games where T_i and T_j play against each other are not swapped. For example, if a team T_L was playing against team T_i at home in a round r_k of s, then in the neighbor solution s' it'll play against team T_j in the same round (r_k) at home.

Round Swap Given two rounds r_i and r_j, the application of the Round Swap in a solution s obtain a solution s' where all the opponents of the all teams were swapped in these two rounds. For example, if a team T_i was playing against the teams T_j and T_k in the rounds r_m and r_L, respectively, of s, then in the neighbor solution s' the team T_i plays against the teams T_j and T_k in the rounds r_L and r_m, respectively.

Partial Round swap Consider the teams T_i, T_j, T_k and T_L and the rounds, r_m and r_z, such that games $\{T_i, T_k\}$ and $\{T_j, T_L\}$ take place in round r_m and games $\{T_i, T_L\}$ and $\{T_j, T_k\}$ take place in round r_z. The application of the move Partial Round Swap consists in swapping the rounds in which these games take place. The games $\{T_i, T_k\}$ and $\{T_j, T_L\}$ are set to round r_z and the games $\{T_i, T_L]$ and $\{T_j, T_k\}$ are set to round r_m.

Games swap This move consists in selecting an arbitrary game and enforcing it to be played in a round, followed by the necessary modifications to avoid teams playing more than one game in the same round. In consequence, more games would be swapped to maintain the feasible of the schedule. The modifications that have to be applied to the current schedule 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 [16].

	First half					Second half				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	6	7	8	9	10
1	-6	4	2	-5	-3	6	-4	-2	5	3
2	-5	3	-1	-4	-6	5	-3	1	4	6
3	4	-2	-5	6	1	-4	2	5	-6	-1
4	-3	-1	6	2	-5	3	1	-6	-2	5
5	2	-6	3	1	4	-2	6	-3	-1	-4
6	1	5	-4	-3	2	-1	-5	4	3	-2

	First half					Second half				
T_i/r_k	**1**	**2**	**3**	**4**	**5**	6	7	8	9	10
1	**-5**	**-3**	2	**4**	**-6**	**5**	**3**	-2	**-4**	**6**
2	**-6**	**4**	-1	**-5**	**3**	**6**	**-4**	1	**5**	**-3**
3	-4	**1**	-5	**6**	**-2**	4	**-1**	5	**-6**	**2**
4	3	**-2**	6	**-1**	**5**	-3	**2**	-6	**1**	**-5**
5	**1**	-6	3	**2**	**-4**	**-1**	6	-3	**-2**	**4**
6	**2**	5	-4	-3	**1**	**-2**	-5	4	3	**-1**

Fig. 3. Schedule before (left) and after (right) the application of *Games swap*

The figure 3 shows the schedule produced by the application of *Game swap* move. Note that several games are swapped, when the team T_3 was enforcing to play with team T_1 in round r_2.

4.3 Clustering Search for mTTP

A *CS metaheuristc is now described: VNS-CS (*Variable Neighborhood Search Clustering Search*). In this approach, the component SM is a hybrid metaheuristic that combines VNS and VND. As it is already known, VNS [12] starts from the initial solution and at each iteration, a random neighbor is selected in the

$N_{(k)}(s)$ neighborhood of the current solution. That neighbor is then submitted to some local search method. If the solution obtained is better than the current, update the current and continue the search of the first neighborhood structure. Otherwise, the search continues to the next neighborhood. The VNS stopped when the maximum number of iterations since the last improvement is satisfied.

The local optimum within a given neighborhood is not necessarily an optimum within other neighborhoods, and a change of neighborhoods can also be performed during the local search phase. This local search is then called Variable Neighborhood Descent (VND) [12].

In this work, we used the VNS with the three movements as follow, in this order: *Games swap*, *Round swap* and *Partial Round swap*. We used the VND as a local search method of the VNS, and it is composed by the other two movements proposed, in this order: *Team swap* and *Home-away swap*.

The IC is the CS's core, working as a classifier, keeping in the system only relevant information, and driving a search intensification in the promising search areas. Initially, a maximum number of clusters (NC) is defined. In our case, we used $NC = 20$. Solutions generated by VNS are passed to IC that attempts to group as known information, according to *distance metric*. In this paper, the distance metric was the number of different games between the solution and the center of the cluster, not considering the home-away definition(A x B equals B x A).

If the information is considered sufficiently new, it is kept as a center in a new cluster. Otherwise, redundant information activates the closest center c_i (center c that minimizes the distance metric), causing some kind of pertubation on it.

The information will be considered similar if there is at least 50% of coincidence between the games of the two compared solutions.

The perturbation means an assimilation process, in which the center of the cluster is update by the new generated solution. Here, we used the Path-Relinking method [8], that generates several points (solutions) taken in the path connecting the solution generated by VNS and the center of the cluster. Since each point is evaluated by the objective function, the assimilation process itself is an intensification mechanism inside the clusters. The new center c_i is the best evaluated solution sampled in the path.

The AM is executed whenever an solution is assigned to a cluster, verifying if the cluster can be considered promising. A cluster becomes promising when reaches a certain density λ_t,

$$\lambda_t \geq PD \left[\frac{NS}{NT_c} \right] \tag{1}$$

where, NS is the number of solutions generated in the interval of analysis of the clusters, NT_c is the number of cluster in the iteration t, and PD is the desirable cluster density beyond the normal density, obtained if NS was equally divided to all clusters. In this work, we used $NS = 300$, $PD = 2$ and $NT_c = 20$, chosen after several tests.

The component LS has been activated when the AM discover a promising cluster. The LS implemented was the Iterated Local Search (ILS) [11]. This metaheuristic starts from a locally optimal feasible solution. A random pertubation (movement *Game swap*) is applied to the current solution and followed by a local search similar to VNS, with the movements *Team swap*, *Partial Round swap* and *Home-away swap*. 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 for 1000 iterations.

Figure 4 shows the Pseudo Code of the algorithm implemented.

```
Procedure VNS-CS(f(.), N(.), s_0)
1   Iter ← 0;              { Current iteration }
2   C_active ← ∅;          { Define if a cluster is active }
3   CP ← False;            { Define if a cluster is promising }
4   s ← s_0;               { Current Solution }
5   while (Iter < IterMax) do
6       k ← 1;
7       while (k ≤ 3) do
8           Generate any neighbor s' ∈ N_(k)(s);
9           s'' ← VND(s');
10          C_active ← Component-IC(s'');
11          CP ← Component-AM(C_active);
12          if(CP = True) then
13              Component-LS(C_active);
14          if (f(s'') < f(s)) then
15              s ← s'';
16              k ← 1;
17          else
18              k ← k + 1;
19          end-if;
20      end-while;
21      Iter ← Iter + 1;
22  end-while;
end-VNS-CS;
```

Fig. 4. Algorithm *VNS-CS*

5 Experiments and computational results

The algorithm was coded in C++ and was run on *Pentium IV* 3.0 *GHz clock with* 512 *Mbytes of RAM memory.*

The benchmark's instances, described in [7] and adapted to the mirrored form in [16] was used to validate the results. A real-life instance (br2003.24), where 24 teams playing in the main division of the 2003 edition of Brazilian Soccer Championship was also tested. These test problems are available to download in http://mat.gsia.cmu.edu/TOURN/. The parameters of the methods were empirically chosen, after several simulations.

Table 1 shows the results for the considered instances. For each instance is reported the best solution found by the algorithms proposed by [16] and [10] (they obtained the best known solutions) and the minimum computation time of this last algorithm, the solutions obtained by this approach, the relative gap in percent between our solutions and the best solution between [16] and [10], and the last column presents the total computation times in seconds. The results represented by "–" wasn't considered by the authors.

Table 1. Computational results

Instances	Best by [16]	Best by [10]	$Time_{[10]}$	VNS-CS	$gap(\%)$	$Time$
circ4	20	–	–	20	0%	6
circ6	72	–	–	72	0%	9
circ8	140	140	0.2	140	0%	438
circ10	272	272	28160	276	1,47%	4951
circ12	456	432	93.1	446	3,14%	1275
circ14	714	696	53053.5	702	0,86%	3672
circ16	978	968	38982.7	978	1,02%	826
circ18	1306	1352	178997.5	1352	3,40%	1153
circ20	1882	1852	59097.9	1882	1,59%	1189
nl4	8276	–	–	8276	0%	5
nl6	26588	–	–	26588	0%	7
nl8	41928	41928	0.1	41928	0%	1030
nl10	63832	63832	477.2	65193	2,09%	228
nl12	120655	119608	15428.1	120906	1,07%	2630
nl14	208086	199363	34152.3	208824	4,53%	796
nl16	279618	279077	55640.8	287130	2,80%	3201
con4	17	–	–	17	0%	4
con6	48	–	–	48	0%	5
con8	80	80	0.1	81	1,23%	39
con10	130	130	0.1	130	0%	92
con12	192	192	0.3	193	0,52%	299
con14	253	253	6.0	255	1,18%	131
con16	342	342	2.7	343	0,29%	407
con18	432	432	8.1	433	0,23%	1265
con20	524	522	1106.3	525	0,57%	1102
br2003.24	503158	–	–	512545	1,83%	798

The results presented demonstrate that the proposed methodology can be competitive, because was possible to obtain values near of the best results known in the literature, getting to reduce to zero the gap in six of the seventeen instances and being near of reducing to zero in some others. The worse gap was $4,53\%$.

The main aspect of the results is the computation time. See that the VNS-CS can reach solutions with similar quality quicker than the algorithm proposed in [10], e.g., whilst the algorithm of the literature took some days (more than two days) to reach the best solutions for the instance circ18, the VNS-CS reached high-quality solutions, near the best known, in few minutes.

The result of the real-life instance was very interesting. First, because it is the larger instance in the literature, with 24 teams. Second, because was observed a reduction of 51.09% in the total distance traveled, where in the official schedule the teams traveled 1.048.134 km and in the schedule found by the work they traveled 512.545 km only.

6 Conclusions

In this work was investigate the Mirrored Traveling Tournament Problem, first published in [16], with a implementation of a new way of detecting promising search areas based on clustering: the Clustering Search (*CS) [15]. Together with other search metaheuristics, working as full-time solution generators, *CS attempts to locate promising search areas by solution clustering. The clusters work as sliding windows, framing the search areas and giving a reference point to problem-specific local search procedures, besides an iterative process, called assimilation.

A metaheuristic based on *CS was proposed to solve the all mTTP instances available in literature: VNC-CS (*Variable Neighborhood Search Clustering Search*).

Five different neighborhood structures for local search was investigated: three simple neighborhood, *Home-away swap*, *Round swap* and *Team swap*; and two more complicated, *Partial Round swap* and *Game swap*.

The approach become very promising, when the reported results are analyzed. Seventeen benchmark instances was tested and VNS-CS approach have achieved similar and sometimes superior performance than the works presented in literature. One real-life instance, the 2003 edition of Brazilian Soccer Championship, was also tested, with reduction of 51.09% of the total distance traveled by the official schedule. Although not all real constraints were analyzed, but only the main constraints, the obtained result for this instance was very interesting.

Finally, this work explores the mirrored instances of TTP, because its represents common structure in Latin-America tournaments.

For further research there are a variety of open issues that need to be addressed, e.g., to study other neighborhood to obtain high-quality solutions. In the same way the real championships has many other restrictions that should be analyzed: treatment of the classic games, allocation of stadiums, do not consider distance metric, but the airfares, and many others real constraints.

References

1. Anagnostopoulos, A.; Michel, L.; Van Hentenryck; P., Vergados, Y.: A Simulated Annealing Approach to the Traveling Tournament Problem. Proceedings of Cpaior'03 (2003)
2. Biajoli, F. L.; Chaves, A. A.; Mine, O. M.; Souza, M. J. F.; Pontes, R. C.; Lucena, A.; Cabral, L. F.: Scheduling the Brazilian Soccer Championship: A Simulated Annealing Approach. Fifth International Conference on the Practice and Theory of Automated Timetabling, Patat2004, Pittsburgh, USA, (2004) 433-437
3. Biajoli, F. L.; Lorena, L. A. N.: Mirrored Traveling Tournament Problem: An Evolutionary Approach. Lecture Notes in Artificial Intelligence Series, v. 4140, p. 208-217, 2006.
4. Concílio, R.; Zuben, F. J.: Uma Abordagem Evolutiva para Geração Automática de Turnos Completos em Torneios. Revista Controle e Automação. Vol. 13, n.2, (2002) 105-122
5. Costa, D.: An Evolutionary Tabu Search Algorithm and the NHL Scheduling Problem. Infor. Vol. 33, n. 3, (1995) 161-178
6. Dinitz, J.; Lamken, E.; Wallis, W. D.: Scheduling a Tournament. In C.J. Colbourn and J. Dinitz, editors, Handbook of Combinatorial Designs, CRC Press, (1995) 578-584
7. Easton , K.; Nemhauser, G.; Trick, M.: The Traveling Tournament Problem Description and Benchmarks. Seventh International Conference on the Principles and Practice of Constraint Programming, CP'99, (2001) 580-589
8. Glover, F.: Tabu search and adaptive memory programing: Advances, applications and challenges. In: Interfaces in Computer Science and Operations Research. [S.l.]: Kluwer, 1996. p. 175.
9. Hamiez, J. P.; Hao, J. K.: Solving The Sports League Scheduling Problem with Tabu Search. Lecture Notes In Artificial Intelligence 2148, Springer (2001), 24-36
10. Hentenryck, P. V.; Vergados, Y. Traveling tournament scheduling: A systematic evaluation of simulated annealling. In: CPAIOR. [S.l.: s.n.], 2006. p. 228243.
11. Loureno, H. R.; Martin, O.; Sttzle, T: Iterated local search. In: . Handbook of Metaheuristics. Norwell: Kluwer Academic Publishers, 2002. p. 321353.
12. Mladenovic, N.; Hansen, P. Variable neighborhood search. Computers and Operations Research, v. 24, p. 10971100, 1997.
13. Nemhauser, G.; Trick, M.: Scheduling a Major College Basketball Conference. Operations Research, Vol. 46(1), (1998) 1-8
14. Oliveira, A. C. M.: Algoritmos Evolutivos Hbridos com Deteco de Regies Promissoras em Espaos de Busca Contnuo e Discreto. 200 p. Tese (Doutorado) Instituto Nacional de Pesquisa Operacional (INPE), So Jos dos Campos - SP, 2004.
15. Oliveira, A. C. M.; Lorena, L. A. N.: Pattern sequencing problems by clustering search. Springer Lecture Notes in Artificial Intelligence Series, v. 4140, p. 218 227, 2006. 57, 78
16. Ribeiro, C.C.; Urrutia S.: Heuristics for the Mirrored Traveling Tournament Problem. Fifth International Conference on the Practice and Theory of Automated Timetabling, Patat2004, Pittsburgh, USA, (2004) 323-342
17. Schönberger, J.; Mattfeld, D. C.; Kopfer, H.:Memetic Algorithm Timetabling for Non-Commercial Sport Leagues. European Journal of Operational Research, Vol. 153(1), (1989) 102-116
18. Zhang, H.: Generating College Conference Basketball Schedules by a Sat Solver. Proceedings Of The Fifth International Symposium on the Theory and Applications of Satisfiability Testing, Cincinnati, (2002) 281-291