$\mathrm{AIS}_{\mathrm{TTP}}$: An Artificial Immune Algorithm to solve Traveling Tournament Problems

Leslie Pérez, *Member IEEE* and María-Cristina Riff, *Member IEEE*

***Abstract*— In this paper we include new components to the Clonal Selection Algorithm to improve its search when solving the Traveling Tournament Problem. We have tested the algorithm in well-known instances of the problem, and the results obtained are very encouraging. They show a new application of an artificial immune algorithm based on the CLONALG framework.**

***Index Terms*— CLONALG, Artificial Immune Algorithm, Traveling Tournament Problem**

I. Introduction

The Traveling Tournament Problem (TTP) is an NP-hard problem and many methods have already been proposed in the literature to solve it. For bigger instances the most successful methods are those based on metaheuristics as simulated annealing, and tabu search with a post-processing local search procedure [3], [5], [9], [7]. On the other hand, Artificial Immune Systems have been applied successfully for solving very complex problems such as virus detection, some traveling salesman problem instances and 3-coloring problems, [14]. In this paper, we introduce new components to the Clonal Selection Algorithm (CLONALG),[12] for solving instances of the traveling tournament problem. Our goal is to show a new application of the immune inspired algorithms.

The contributions of this paper are:

- New components for CLONALG that help it to tackle TTP.
- A new move focused on minimizing the time away of the teams.

It is important to note that our initial goal is not to obtain the best algorithm to solve TTP but to show that an improved version of CLONALG is able to solve this kind of problem in a competitive way compared to other metaheuristics. This paper is organized as follows: In the following sections we briefly present the Traveling Tournament Problem and the CLONALG framework. The new components for the algorithm are introduced in section 4. The results obtained using our approach for solving various instances of TTP are reported in section 5. Finally, the conclusion and future work are presented in section 6.

Leslie Pérez, Universidad Técnica Federico Santa María, Chile, e-mail: Leslie.Perez@inf.utfsm.cl

María-Cristina Riff, Universidad Técnica Federico Santa María, Chile, e-mail: Maria-Cristina.Riff@inf.utfsm.cl. This work is partially supported by FONDECYT Project 1080110 and Centro Científico Tecnológico de Valparaíso (CCT-Val) FB0821

II. Traveling Tournament Problem

The Traveling Tournament Problem [1] is a sports timetabling problem which involves two issues in creating timetables: assignment feasibility (tournament structure) and optimization (reducing distance traveled by the team). The objective is, given a number of teams N and a distance matrix $D_{i,j}$, to find a double round robin tournament which minimizes the distance traveled. The *trip length* constraint establishes the minimum and the maximum number of consecutive home/away matches allowed during the tournament. Usually, the lower and upper boundaries are one and three respectively.

A solution must also satisfy the following constraints:

- No repeaters: Matches that involve the same pair of teams (i, j) must not take place in consecutive rounds.
- Mirrored: The first half of the tournament should be a single round robin tournament, the second is obtained by mirroring the first one.

The double round robin structure requires $2(N-1)$ rounds, $N/2$ matches must take place every round, each pair of teams must play twice during the tournament exchanging homes, and all teams must play only one match every round.

This problem is a real challenge to both, complete and incomplete combinatorial optimization techniques. Best complete techniques like [2] are able to find the optimum for small instances of up to 8 teams. The same authors have recently informed new results for the problem of 10 teams on their web site using a complete technique. Finding the optimum requires a long execution time. For bigger instances the most successful methods are those based on metaheuristics. A simulated annealing for TTP (TTSA) is proposed in [3] where reheats and a strategic oscillation strategy are used to vary the constraint weight parameter during the search. This method was very successful for solving TTP. Later, the authors in [7] used TTSA as individuals of a population based search method, in which waves of individuals are executed. This has improved the results obtained by TTSA. Ant colony optimization (ACO) with constraint processing techniques is proposed in [10], the results are promising and outperform other ACO approaches. A hybrid method is presented in [4] using simulated annealing and hill climbing. Tabu search has also being successfully applied to TTP in [5]. In the field of metaheuristic, an ant based hyper-heuristic is introduced in [9] and a learning hyper-heuristic is given in [11]. Both obtain good results.

978-1-4244-7376-2/10/$26.00 ©2010 IEEE

III. A DESCRIPTION OF CLONALG

Artificial Immune Systems (AIS) have been defined as adaptive systems inspired by the natural immune system and applied to problem solving. In this paper, we are interested in CLONALG, an artificial immune algorithm based on Clonal Selection Principle [12]. It follows the basic theory of the adaptive immune response to pathogens. Roughly speaking, the components of the algorithm are cells, antibodies, and a potentially dangerous invader, named antigen. The immune system reproduces those cells that are able to recognize an antigen. Cells which match well are cloned (create copies of themselves), and the better is the match, the more clones are generated. During this process, hyper-mutation is applied to clones, which may improve their antigen recognition. The better the parent cell match, the less mutations the clone suffers. Moreover, the clonal selection process also retains the best cells as memory cells, resulting in a much more efficient system in further encounters with the recognized antigen.

Figure 1 shows the CLONALG algorithm.

CLONALG
Begin

$Cells$ = Initialise population of size n_1 (1)
i=1;
Repeat
Calculate Fitness ($Cells$) (2)
B_a = Selected n best antibodies from $Cells$ (3)
P_c = Generate C clones of each antibody from B_a (4)
P_c = Mutation(P_c) (5)
$Cells$ = Selected n best antibodies from $P_c \cup Cells$ (6)
$Cells$ = $Cells$ + Generate n_2 New antibodies (7)
$i = i + 1$;
until i=Max-number-of-iterations
End

Fig. 1. Clonal Selection Algorithm

The algorithm randomly generates a population of $Cells$ and computes their fitness. The iterative process begins by constructing a sub-population B_a of size n, composed by the n best antibodies belonging to the population of $Cells$. A new population of clones P_c is constructed by generating C clones of each element on B_a. This population of clones follows a mutation process in order to improve the evaluation of the clones. The set of cells is updated including the n best antibodies from $P_c \cup Cells$ and incorporates new randomly generated antibodies (n_2) to construct the population of size n_1.

The CLONALG framework is a general guide to design artificial immune systems. When using this framework to design an immune algorithm for a specific problem, the main research task consists in defining all of its components, the same way that this is done for other techniques like genetic algorithms. To help the search of the immune algorithm, this design process must take into account the knowledge of the problem.

IV. ARTIFICIAL IMMUNE ALGORITHM FOR TTP

In this section we introduce our artificial immune algorithm to solve the TTP.

A. Representation

Definition 4.1: Given the set $S_T = \{T_1, \ldots, T_N\}$ of N teams, and the set $S_R = \{R_1, \ldots, R_M\}$ of M rounds, for a round robin scheduling we define the representation TM as the NxM Tournament Matrix, where the value of each element $|tm_{ij}|$ indicates the opponent of team T_i in the round R_j. More precisely,

- $|tm_{ij}| \in \{1, \ldots, N\}$, with $|tm_{ij}| \neq i$
- $tm_{ij} > 0$ when the match is at home
- $tm_{ij} < 0$ when the match is away.

Figure 2 shows an example of the Tournament Matrix considering four teams and six rounds. In this example team 2 will play away in the third round against team 1.

T/R	**1**	**2**	**3**	**4**	**5**	**6**
1	-2	3	2	4	-3	-4
2	1	-4	-1	3	4	-3
3	-4	-1	4	-2	1	2
4	3	2	-3	-1	-2	1

Fig. 2. Representation matrix (T=4)

B. Managing Constraints

The TTP is a very hard problem and it has many constraints to be satisfied. In our approach we consider those related to the double round robin stucture as hard constraints, and the moves take their satisfaction into account. The non-structural constraints of *No repeaters* and *trip length* are managed by a penalization on the fitness function.

C. Fitness Function

In order to guide the algorithm to find good quality solutions which also satisfy the constraints, the algorithm uses equation 1 which computes the penalty function W_k for the constraint k. In this equation $\bar{d}$ is the average distance among teams and δ_k is the penalty factor associated to the dissatisfaction of the constraint k.

$$W_k = \bar{d} * (N-1) * \delta_k, \forall k = 1, 2. \quad (1)$$

Finally, equation 2 defines the fitness function F_{ttp}, considering all rounds, ϕ_i is the total traveled distance of team i and v_k is the number of violated non-structural constraints in a scheduling candidate.

$$F_{ttp} = \sum_{i=1}^{N} \phi_i + \sum_{k=1}^{2} v_k * W_k \quad (2)$$

D. Moves

The algorithm uses six moves. The following five moves have been proposed for a simulated annealing algorithm in [3]:

- Swap Homes(T_i,T_j): Exchanges teams T_i and T_j matches home (matrix elements sign).
- Swap Rounds(R_i,R_j): Exchanges rounds R_i and R_j schedules.
- Swap Teams(T_i,T_j): Exchanges teams T_i and T_j matches, except for the ones which involve themselves.
- Partial Swap Rounds(T_i,R_j,R_k): For a team T_i, exchanges round R_j match for R_k round match.
- Partial Swap Teams(T_i,R_j,R_k): For a round R_k, exchanges teams T_i and T_j matches.

These moves produce many changes to the candidate solution, however none of them take into account the traveling cost involved. Thus, in order to guide the algorithm to search better solutions in terms of a reduction of the traveling cost, we define a new move called *Away Tuple Group*.

1) A new move - Away Tuple Group: As the cost of a team schedule corresponds to the cost of its away matches (trips), the idea is to try to group its trips to reduce the traveling cost. Before explaining the move we require the following definitions:

Definition 4.2: (Pattern Matrix). Given a Tournament Matrix TM, we define the associated Pattern Matrix of size NxM, such that $\forall i = 1, \ldots, N$ and $\forall j = 1, \ldots, M$:

$$PM(i,j) = \begin{cases} \mathbf{H}, & \text{if } tm_{ij} > 0, \\ \mathbf{A}, & \text{if } tm_{ij} < 0 \end{cases}$$

Definition 4.3: (Away tuple). Whenever we refer to an "away tuple" we define any continuous sequence of character **A**.

Definition 4.4: (Feasible away tuple). A feasible away tuple is an away tuple that can be modified in order to increase the continuous **A** character sequence size.

Because the upper limit for the trip length is three, only feasible away tuples are **A** and **AA**.

Definition 4.5: (Set of grouping candidates). Given $PM(i,j)$, for each team T_i, the set of its grouping candidates is composed by all the feasible away tuples identified in $PM(i,j)$.

Figure 3 describes the new move.

Away-tuple-group(candidate-solution)
Begin

While not change made
Randomly Choose Team t from candidate-solution (1)
f_a = Set of feasible away tuples of t (2)
Randomly select an away tuple a from f_a (3)
Swap Homes to append one feasible **A** to a (4)
EndWhile
End

Fig. 3. Away Tuple Group Move

Randomly choose team: 3

-2	-4	-5	6	5	4	-6	1	2	-1	Cost: 320

Home/Away pattern:

A	A	A	H	H	H	A	H	H	A

Away Tuple Numeration:

1	1	1	H	H	H	2	H	H	3

Randomly choose tuple: 2
Execute Swap Homes

A	A	A	H	H	H	A	**A**	H	**H**

-2	-4	-5	6	5	4	-6	**-1**	2	**1**	Cost: 245

Fig. 4. Away-tuple-group move

The process selects a team and identifies the away tuples for this team. An away tuple can be grouped if it is possible to change from *home* to *away* one of its adjacent rounds. Then, a tuple is selected and the Swap Homes move is executed in order to append one *away* more to the tuple. When the Swap Homes can be made either to the left or to the right, the direction is randomly chosen. Figure 4 shows an example of the procedure which uses the *home/away* patterns of the candidate solution.

The candidate solutions generated by applying all the moves always satisfy the hardest constraints related to the double round robin structure.

E. Algorithm

Our algorithm for the TTPs is described in figure 5.

AIS_{TTP}
Begin

$Cells$ = Initialise population of size n_1 (1)
i=1;
Repeat
Calculate Fitness ($Cells$) (2)
$n = r_c * n_1$ (3)
B_a = Select n best antibodies from $Cells$ (4)
P_c = Generate C clones of each antibody from B_a (5)
P_c = Hyper-mutation(P_c) (6)
$P_c = P_c \cup Cells$ (7)
$Cells$ = Diversity_selection(P_c, n) (8)
$Cells$ = $Cells$ + Generate n_2 New antibodies (9)
$i = i + 1$;
until i=Max-number-of-iterations
End

Fig. 5. AIS_{TTP} for Traveling Tournament Problem

In figure 5 line (1), Initial solutions are obtained from the following three simple steps:

- The algorithm uses the *"polygon method"* [8] to create a single round robin structure (graph 1-factorization), then
- this structure is mirrored and

- homes are randomly assigned to complete the double round robin tournament.

In line (3) in figure 5, r_c is the clonation rate which represents the percentage of the repertoire to be selected. After that the algorithm obtains the set of clones generated using the affinity proportional equation 3, proposed in [13], where f_c is the clonal factor, n_1 the repertoire size and l the antibody ranking.

$$C = \sum_{l=1}^{n} round(\frac{f_c * n_1}{l}) \quad (3)$$

1) Hyper-mutation: The hyper-mutation procedure, according to CLONALG, should be inversely proportional to the antibody affinity. In our approach, we use the antibody ranking as an hyper-mutation level indicator. It determines the number of moves to apply to each clone. Figure 6 shows the procedure where (sol_i) is the best new antibody obtained after applying the five moves (op_k). This new antibody can be modified by the Away-tuple-group move if it improves the value of the sol_i fitness function. The key idea of the Away-tuple-group is to reduce the traveling cost of the candidate solution obtained by applying the five moves. It corresponds to a refinement process.

HYPER-MUTATION (P_c)
Begin

For each s_i **in** P_c
  For i=0 **to** ranking
    For k=1 **to** 5
      s_k^* = $op_k(s_i)$
    Endfor
    sol_i = $Best(s_1^*, \ldots, s_5^*)$
    S=Away-tuple-group(sol_i);
    If (S better than sol_i) **then** sol_i=S;
  Endfor
Endfor
End

Fig. 6. Hyper-mutation Operator

2) Diversity Selection: The population of clones is composed of the union between Cells and the hyper-mutated population P_c. The selection to construct the new set of cells is different from the original CLONALG. In our approach, the selection does not only search the best antibodies but it also considers that an antibody to be included in the selected_cells set must also be as different as possible to those already belonging to this set. That means, it is not sufficient to be a good antibody in terms of the fitness function. Moreover, the new antibody must also have different TM values.

The diversity for each antibody in P_c is measured by the average of the hamming distance between its home/away pattern and the home/away pattern of the antibodies that are already members of the selected_cells set. The goal of this procedure, is to select those which are the most different in terms of their home/away patterns from the best antibodies.

Diversity_selection(P_c, n)
Begin

new_cell= best antibody in P_c
P_c=P_c-new_cell
selected_cells=new_cell
i=1;
While $i < n$ **do**
  new_cell=empty
  dt= Set initial diversity selection threshold
  Compute diversity for each antibody in P_c
  While new_cell is empty **do**
    new_cell = best and dt diverse antibody from P_c
    If new_cell is empty **Then**
      dt= Reduce diversity threshold
    Else P_c = P_c - new_cell
    EndIf
  EndWhile
  Add new_cell to selected_cells
  $i = i + 1$;
EndWhile
End

Fig. 7. Diversity Selection

A diversity threshold is defined. It is reduced when none of the antibodies has the required diversity level. This procedure avoids a premature convergence of the algorithm.

V. Experiments and Results

The hardware platform for the experiments was a PC Intel Corei7-920 with 4GB RAM under the Linux Mandriva 2010 operating system. The algorithm has been implemented in C++.
Two sets of experiments have been done:

- To compare the original CLONALG with our approach
- To give a comparison among our approach and the best known ones in this research domain.

A. Comparing different versions of CLONALG

The tests are carried out with three algorithms:

- the first one is the original version of CLONALG,
- the second one corresponds to another version of CLONALG which only includes our diversity procedure (DCLONALG), and
- our approach (AIS_{TTP}) which includes both the diversity procedure and the new Away-tuple-group move.

The first goal of these tests is to evaluate the improvement obtained using our approach respect to the original algorithm. The second goal is to evaluate the impact of the new move on the algorithm. For these experiments we have compared the algorithms using tests instances from the National League of The Major League Baseball in United States (NL) and from the Rugby League composed of teams from New Zealand, Australia, and South Africa (SUPER).

1) Tests set-up: The algorithms use a population size of 10. We consider both non-structural constraints to be of equal importance to satisfy, and they have therefore the same

TABLE I

BEST TRAVEL DISTANCE FOR NLX INSTANCES

BEST	NL8	NL10	NL12	NL14	NL16
CLONALG	40414	66714	124997	230010	333347
DCLONALG	40156	62669	121904	220657	319490
AIS_{TTP}	39947	62103	120572	213854	311053

TABLE II

BEST TRAVEL DISTANCE FOR SUPERX INSTANCES

BEST	S8	S10	S12	S14
CLONALG	190510	348814	512652	715078
DCLONAG	182457	323475	498088	684693
AIS_{TTP}	182409	322845	497668	665674

weight factor of 0.5. The three algorithms have been run 50 times, each of them using 1000 iterations.

The algorithm uses three other parameters: r_c (clonation rate), f_c (clonal factor) and r_r (replacement rate, related to the number of new antibodies included at each iteration). These parameters are tuned by considering all of the values of their range, and selecting the best one for each algorithm and for each instance.

The best results obtained by each type of algorithm when solving NL instances are in table I. For SUPER instances the best results are in table II.

We can conclude that both the diversity strategy and the Away-tuple-group move have been useful for the algorithm to find better solutions than the original CLONALG. Tables IV and V show the average of the solutions obtained for the three algorithms using the different parameter values for r_c, f_c and r_r reported in table III.

From these tables, we can observe that the new move allows the reduction of the average traveled distance of best solutions. Moreover, the standard deviation of the best solutions found by the algorithm is also strongly reduced as it can be observed from the box-plots in figures 8 and 9. Note that the scale of y-axis in the boxplots are not the same, because their boundary values are quite different. For each instance the y-axis represents the difference, in percentage, between the best solution we obtain and the best known solution.

TABLE III

BEST PARAMETERS

	Clonal rate	Clonal Factor	Replacement Rate
NL8	0.7	0.6	0.3
NL10	0.7	1	0.3
NL12	0.8	0.8	0.3
NL14	0.8	0.8	0.2
NL16	0.5	1	0.4
S8	1	0.7	0.2
S10	0.9	0.7	0.5
S12	0.9	1	0.2
S14	0.6	1	0.2

TABLE IV

AVERAGE TRAVEL DISTANCE FOR NLX INSTANCES

Avg.	NL8	NL10	NL12	NL14	NL16
CLONALG	43632	70272	134677	249319	350140
DCLONALG	42149	68807	132629	242220	348946
AIS_{TTP}	40347	63561	123447	220113	319965

TABLE V

AVERAGE TRAVEL DISTANCE FOR SUPERX INSTANCES

Avg.	S8	S10	S12	S14
CLONALG	231043	383632	562817	790466
DCLONAG	208883	365855	544520	775981
AIS_{TTP}	184241	330198	503210	699172

B. Comparison with the state of the art algorithms

Tables VI and VII report the best values obtained using our approach with 8000 iterations for the smaller problems and 10000 iterations for the biggest one. The solutions found by well known existing techniques are also included in the tables. Our results are very encouraging, and as far we know it is the first immune based approach proposed to solve TTP. Moreover, it is a new application using the CLONALG framework for solving this hard and challenging problem.

Table VIII resumes the average execution time reported by the authors of the existing techniques (we include those with reported time in the papers). We also include a table IX with the processor configurations used to run the tests. For the hardest problem (NL16), one run of AIS_{TTP} required 169 seconds on Corei7-720 CPU for 5000 iterations.

VI. CONCLUSION

We have introduced an immune-inspired algorithm based on the CLONALG framework to solve instances of the traveling tournament problem. One difficulty found when applying the original CLONALG framework is that the algorithm get stuck in local optima very quickly, thus its capacity of exploration is strongly reduced to a few number of iterations. To solve this issue, we have defined a new way

TABLE VI

COMPARISON WITH THE STATE OF THE ART ALGORITHMS, BEST SOLUTION FOUND FOR NLX INSTANCES

	NL8	NL10	NL12	NL14	NL16
PBSA [7]	-	-	110729	188728	261687
TTSA [3]	39721	59583	111248	188728	263772
TS [5]	-	59583	111483	190174	270063
L. HYP [11]	39721	59583	112873	196058	279330
SA + HC [4]	39721	59821	115089	196363	274673
ACO [10]	39721	60399	115871	203205	292214
VNS-CS [6]	41928	65193	120906	208824	287130
ANT HYP. [9]	40361	65168	123752	225169	321037
This Paper	39721	61351	120531	206434	298246
Best known	**39721**	**59436**	**110729**	**188728**	**261687**
GAP %	*0*	*3.2*	*8.9*	*9.4*	*13.4*

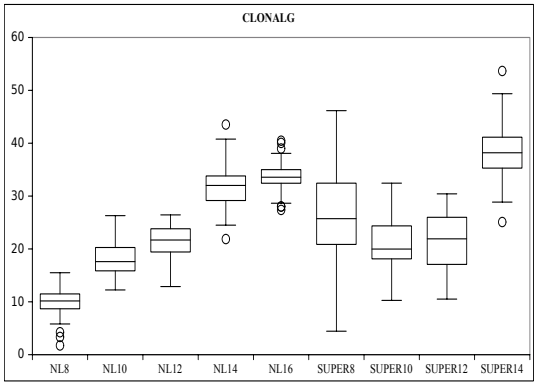

Fig. 8. CLONALG Box-plot

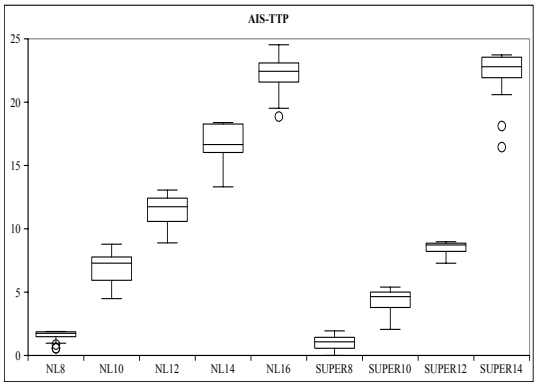

Fig. 9. AIS_{TTP} Box-plot

for selecting memory cells. Our approach is not only focused on having a good set of cells in memory in terms of their fitness function, but also on their structural diversity. We also propose a new move that does not depend on an immune approach. It can be used by other techniques to reduce the travel costs when the algorithm makes a change. We note that the algorithm requires different parameter values for each instance of the TTP to be solved. In order to reduce the hard time consuming task of tuning, we plan to define a dynamic parameter control strategy, which allows the algorithm to self-adapt its parameter values during its search, no matter which instance it is solving.

ACKNOWLEDGMENT

The authors would like to thank Dr. Xavier Bonnaire for his helpful remarks.

TABLE VII

COMPARISON WITH THE STATE OF THE ART ALGORITHMS, BEST SOLUTION FOUND FOR SUPERX

	S8	S10	S12	S14
L. Hyp. [11]	182409	318421	467267	599296
This Paper	182409	321896	490868	657577
Best known	**182409**	**316329**	**463876**	**571632**
GAP %	*0*	*1.8*	*5.8*	*15.0*

TABLE VIII

AVERAGE TIME IN SECONDS

Avg.	NL8	NL10	NL12	NL14	NL16
PBSA [7]	-	-	1501	2491	12858
TTSA [3]	1639	27818	150328	77587	476191
TS [5]	-	1445	1330	2937	5885
SA + HC [4]	52497	67619	82322	96822	111935

TABLE IX

COMPUTER CONFIGURATIONS

PBSA [7]	Cluster of 60 Intel dualcore, dualprocessor Dell
TTSA [3]	AMD Athlon 64, 2Ghz
TS [5]	AMD Athlon, 1.5 Ghz
SA + HC [4]	Pentium 4, 2.53 Ghz 512 RAM

REFERENCES

[1] K. Easton, G. Nemhauser and M. Trick, *The Traveling Tournament Problem Description and Benchmarks*, Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, 2001.

[2] D. Uthus, P. Riddle and H. Guesgen, *DFS* and the Traveling Tournament Problem*, Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems; pages 279-293, Springer-Verlag, 2009.

[3] A. Anagnostopoulos, L. Michel, P. Van Hentenryck and Y. Vergados, *A simulated annealing approach to the traveling tournament problem*, Journal of Scheduling; volume 19 pages 177-193, Kluwer Academic Publishers, 2006.

[4] A. Lim, B. Rodrigues and X. Zhang, *A simulated annealing and hill-climbing algorithm for the traveling tournament problem*, European Journal of Operational Research; volume 127 pages 1459-1478, 2006.

[5] L. Di Gaspero and A. Schaerf, *A composite-neighborhood tabu search approach to the traveling tournament problem*, Journal of Heuristics; volume 13 pages 189-207, Kluwer Academic Publishers, 2007.

[6] L. Nogueira and F. Lacerda, *Clustering Search Approach for the Traveling Tournament Problem*, Mexican International Conference on Artificial Intelligence; LNCS 4827 pages 83-93, 2007.

[7] P. Van Hentenryck and Y. Vergados, *Population-based simulated annealing for traveling tournaments*, Proceedings of the 22nd national conference on Artificial intelligence; pages 267-272, 2007.

[8] C. Ribeiro and S. Urrutia, *Heuristics for the mirrored traveling tournament problem*,European Journal of Operational Research; volume 127 pages 775-787, 2007.

[9] G. Kendall and V. Berghe, *An Ant Based Hyper-heuristic for the Travelling Tournament*, In proceedings of IEEE Symposium of Computational Intelligence in Scheduling; pages 19-26, 2007.

[10] D. Uthus, P. Riddle and H. Guesgen, *An ant colony optimization approach to the traveling tournament problem*, Proceedings of the 11th Annual conference on Genetic and evolutionary computation; pages 81-88, 2009.

[11] M. Misir, T. Wauters, K. Verbeeck and G. Vanden Berghe, *A New Learning Hyper-heuristic for the Traveling Tournament Problem*, The VIII Metaheuristics International Conference; 2009.

[12] L. N. De Castro and J. Timmis, *Artificial Immune System: A New Computational Intelligence Approach*, Springer-Verlag, 2002.

[13] L. N. De Castro and F. J. Von Zuben, *Learning and optimization using the clonal selection principle*, IEEE Transactions on Evolutionary Computation; number 3 volume 6 pages 239-251, 2002.

[14] M-C. Riff, M. Zuniga and E. Montero, *A graph-based immune inspired constraint satisfaction search*, Neural Computing and Applications Journal, in press., 2010, DOI:10.1007/s00521-010-0390-8.