New Bounds for the Relaxed Traveling Tournament Problems using an Artificial Immune Algorithm

Leslie Pérez and María Cristina Riff
Computer Science Department, University Technical Federico Santa María
Valparaíso, Chile
{leslie.perez, maria-cristina.riff}@inf.utfsm.cl

***Abstract*—In this paper we tackle the Relaxed Traveling Tournament Problem by an artificial immune algorithm. We introduce new moves that allow the algorithm to group byes when it is useful. We have tested the algorithm in the recently proposed instances of the problem, the results obtained are very encouraging.**

I. Introduction

The Traveling Tournament Problem (TTP) is an NP-hard problem and many methods have already been already 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 [4], [6], [10], [8]. Trick and Bao propose the new form of the TTP: Relaxed Traveling Tournament Problem (RTTP), [2]. In this version the teams can have some days off between two matches. This is still a very hard problem and only problems with less than 9 teams have known upper bounds. On the other hand, Artificial Immune Systems have been applied successfully for solving very complex problems such as virus detection, some traveling salesman's problem instances and 3-coloring problems, [15]. In this paper, we introduce new components to the Clonal Selection Algorithm (CLONALG),[13] for solving instances of the traveling tournament problem. Our goal is to solve a new problem using the immune inspired algorithms.

The contributions of this paper are:

- New components based on population diversity for CLONALG that help it to tackle the Relaxed TTP.
- A new move focused on maximizing the time away of the teams whose trip costs from home are more expensive.

This paper is organized as follows: In the following sections we briefly present the Relaxed 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 Relaxed TTP are reported in section 5. Finally, the conclusion and future work are presented in section 6.

II. Relaxed 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 [3] 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 [4] 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 [8] 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 [11], the results are promising and outperform other ACO approaches. A hybrid method is presented in [5] using simulated annealing and hill climbing. Tabu search has also being successfully applied to TTP in [6]. In the field of metaheuristic, an ant based hyper-heuristic is introduced in [10] and a learning hyper-heuristic is given in [12]. Both obtain good results.

The Relaxed Travel Tournament Problem considers that each team schedule has exactly k days off. Free days are ignored for determining the lenght of a homestand or roadtrip, and in determining whether a repeater has ocurred. Some

978-1-4244-7835-4/11/$26.00 ©2011 IEEE

bounds are already known for the benchmarks for NL4, NL6 and NL8 using k=1, k=2 and k=3.

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 [13]. 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 the Relaxed TTP

In this section we introduce our artificial immune algorithm to solve the Relaxed 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, and k byes, for a round robin scheduling we define the representation TM as the Nx$(M + k)$ Tournament Matrix, where the value of each element $|tm_{ij}|$ indicates the opponent of team T_i in the round R_j. More precisely,

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

Figure 2 shows an example of the Tournament Matrix considering four teams, six rounds and three byes. In this example, that is the optimal solution for NL4 instance, team 3 will play away in the three first rounds and it has a bye in round four, eight and nine.

3	0	-4	-2	-3	0	2	4	0
0	0	3	1	4	-3	-1	0	-4
-1	-4	-2	0	1	2	4	0	0
0	3	1	0	-2	0	-3	-1	2

Fig. 2. Representation matrix (T=4)

B. Managing Constraints

It 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_l for the constraint l. In this equation $\bar{d}$ is the average distance among teams and δ_l is the penalty factor associated to the dissatisfaction of the constraint l.

$$W_l = \bar{d} * (N - 1) * \delta_l, \forall l = 1, 2. \quad (1)$$

Finally, equation 2 defines the fitness function F_{Rttp}, considering all rounds, ϕ_i is the total traveled distance of team i

and v_l is the number of violated non-structural constraints in a scheduling candidate.

$$F_{Rttp} = \sum_{i=1}^{N} \phi_i + \sum_{l=1}^{2} v_l * W_l \quad (2)$$

The distance value when a team has a bye is considered equal to zero. Thus, if team 5 has the sequence [-1 0 -3], the distance to be computed for this tuple is the distance between the locations of teams 1 and 3, instead of a sequence [-1 0 3], the distance will be the distance between the locations of team 1 and team 5. Thus the zero value is ignored for computing the fitness function.

Remark 4.1: Note that a sequence [-1 -2 0 -3 -4] is considered unfeasible, because a maximum of three away is permitted, analogously the sequence [-1 -2 0 -3 4] is a feasible one. The bye is considered as a joker day.
Using this function the algorithm quickly focuses its search to the feasible solutions by trying to find candidate solutions that reduce the penalizations.

D. Moves

The algorithm uses seven moves. The following five moves have been proposed for a simulated annealing algorithm in [4]:

- 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_m): For a team T_i, exchanges round R_j match for R_m round match.
- Partial Swap Teams(T_i,R_j,R_m): For a round R_m, exchanges teams T_i and T_j matches.

These moves produce many changes to the candidate solution, however none of them take into account neither the byes nor traveling cost involved. Thus, we have designed the move *Swap Byes Matches* which is focused on the free-days. It tries to interchange data from two rounds: a round where teams T_1 and T_2 have a free-day. The second round is when these two teams are playing together. Figure 3 shows the application of this move between team 1 and team 3. In this case, the match between team 1 and team 3 is changed from round 5 to round 9, and now both teams have a free-day in round 5 instead of round 9. It is a very quick move and very useful, because the optimization process requires a good manage of the byes. The byes allow to extend the away period of a team. Thus, the byes positioning is very important for obtaining a good solution for our problem.

In order to guide the algorithm to search better solutions in terms of a reduction of the traveling cost, we have designed a more complex move named *Grouping Away using Byes*.

1) Grouping Away using Byes: As the cost of a team schedule corresponds to the cost of its away matches (trips), the idea is to try to group the trips of the teams with more expensive traveling costs using the byes. The procedure begins with the teams with more expensive away costs. Before explaining the move we require the following definitions:

3	0	-4	-2	**0**	0	2	4	**-3**
0	0	3	1	4	-3	-1	0	-4
-1	-4	-2	0	**0**	2	4	0	**1**
0	3	1	0	-2	0	-3	-1	2
3	0	-4	-2	**-3**	0	2	4	**0**
0	0	3	1	4	-3	-1	0	-4
-1	-4	-2	0	**1**	2	4	0	**0**
0	3	1	0	-2	0	-3	-1	2

Fig. 3. Swap Byes Matches - Teams (1,3)

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

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

Definition 4.3: (Away Pattern) Whenever we refer to an "away pattern" we define any continous sequence without character **H**.

Definition 4.4: (Feasible Away Group) A feasible away group is an away pattern that can be modified in order to increase the continous sequence size of non-character **H**.
Because the upper limit for the trip lenght is three, only feasible away tuples when we consider k=1 byes are **A**, **AA**, **A-**, **AA-**, **-A**, **-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 group identified in $PM(i,j)$.

Grouping-Away-Byes(candidate-solution, t**)**
Begin

f_a = Set of feasible away groups of team t (1)
While not change made
Randomly select an away tuple a from f_a (2)
Swap Rounds to append one feasible **A** to a (3)
EndWhile
End

Fig. 4. Grouping-Away-Byes Move

An away pattern can be grouped if it is possible to change from *home* to *away* in one of the neighbor's round. Then, a tuple is selected and the Swap Rounds move is executed in order to append one *away* more to the tuple. When the Swap Rounds can be made either to the left or to the right, the direction is randomly chosen. Figure 5 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

Team selected: 3

-2	-4	0	1	2	0	4	-1

Home/Away pattern:

A	A	-	H	H	-	H	A

Away Tuple Numeration:

1	1	1	H	H	-	H	2

Choose tuple: 1

Execute Swap Rounds

A	A	-	**A**	H	-	H	**H**

Fig. 5. Grouping-Away-Byes

round robin structure.

E. Algorithm

Our algorithm for the Relaxed TTPs is described in figure 6.

$AIS - RTTp$

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 = Hypermutation(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. 6. $AIS - RTTp$

In figure 6 line (1), Initial solutions are obtained from the following three simple steps:

- The algorithm uses the *"polygon method"* [9] 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 6, 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 [14], 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 7 shows the procedure where (sol_i) is the best new antibody obtained after applying the six moves (op_k): Swap Homes, Swap Rounds, Swap teams, Partial Swap Rounds, Partial Swap Teams and Swap Byes Matches. This new antibody can be modified by the Grouping-Away-Byes move if it improves the value of the sol_i fitness function. The key idea of the Grouping-Away-Byes is to reduce the traveling cost of the candidate solution obtained by applying the six moves. It corresponds to a refinement process.

HYPERMUTATION (P_c)
Begin

For each s_i **in** P_c
For i=0 **to** ranking
For k=1 **to** 6
s_i^* = $op_k(s_i)$
Endfor
sol_i = $Best(s_1^*, \ldots, s_6^*)$
S=Grouping-Away-Byes(sol_i);
If (S better than sol_i) **then** sol_i=S;
Endfor
Endfor
End

Fig. 7. Hypermutation Operator

2) Diversity Selection: The population of clones is composed of the union between Cells and the hypermutated 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.

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. 8. Diversity Selection

The diversity for each antibody in P_c is measured by the average of the hamming distance between its home/away

TABLE I
BEST TRAVEL DISTANCE FOR NLX INSTANCES

BEST	NL8	NL10	NL12	NL14	NL16
CLONALG	40414	66714	124997	230010	333347
$AIS - RTT_p$	39955	60760	122074	210740	320137

TABLE II
BEST TRAVEL DISTANCE FOR SUPERX INSTANCES

BEST	S8	S10	S12	S14
CLONALG	190510	348814	512652	715078
$AIS - RTT_p$	182862	322030	500790	696472

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 more different in terms of their home/away patterns from the best antibodies. 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++.

Three 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, considering k=0 as it is usual in TTP.
- To show our results in the Relaxed TTP instances.

A. Comparing different versions of CLONALG for RTTP using k=0

The tests are carried out with two algorithms:

- The original version of CLONALG, and
- our approach which includes both the diversity procedure and the new moves: Swap Byes Matches and Grouping Away Byes.

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

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.

TABLE III
BEST PARAMETERS

	Clonal rate	Clonal Factor	Replacement Rate
NL8	0.9	0.9	0.2
NL10	0.8	0.7	0.1
NL12	0.4	0.7	0.2
NL14	0.5	0.8	0.8
NL16	0.7	0.9	0.3
S8	0.5	0.4	0.5
S10	0.7	0.6	0.4
S12	0.7	0.8	0.4
S14	0.9	1	0.5

TABLE IV
AVERAGE TRAVEL DISTANCE FOR NLX INSTANCES

Avg.	NL8	NL10	NL12	NL14	NL16
CLONALG	43632	70272	134677	249319	350140
$AIS - RTT_p$	41692	68523	132440	253984	346183

We can conclude that both the diversity strategy and the new move has 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 both algorithms. We remark from these tables that for NL14 and S14 the average travel distances obtained by $AIS - RTT_p$ are greater than those obtained by CLONALG. However, the best solutions found for these instances are better for $AIS - RTT_p$. The algorithms use different parameter values for 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 have been tuned by considering all of the values of their range, and selecting the best one for each algorithm and for each instance. The parameter values found for $AIS - RTT_p$ are reported in table III.

From these tables, we can observe that the new move allows the reduction of the traveled distance of the best solutions. Moreover, the standard deviation of the best solutions found by the $AIS - RTT_p$ is also reduced as it can be observed from the box-plots in figures 9 and 10. For each instance the y-axis represents the difference, in percentage, between the best solution we obtain and the best known solution.

TABLE V
AVERAGE TRAVEL DISTANCE FOR SUPERX INSTANCES

Avg.	S8	S10	S12	S14
CLONALG	231043	383632	562817	790466
$AIS - RTT_p$	215149	372340	549660	791848

B. Comparison with the state of the art algorithms using k=0

Tables VI and VII report the best values obtained using our approach with 8000 iterations for the smaller problems and 60000 iterations for the biggest one. The solutions found by well known existing techniques are also included in the tables.

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

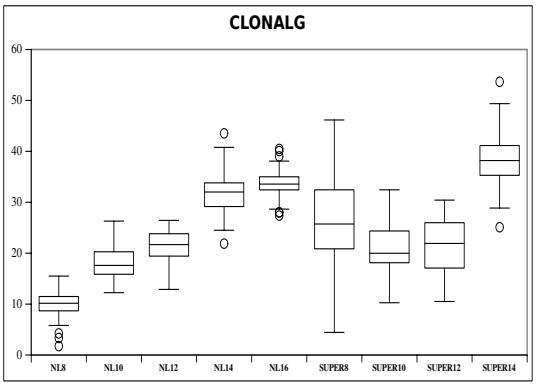

Fig. 9. CLONALG Box-plot

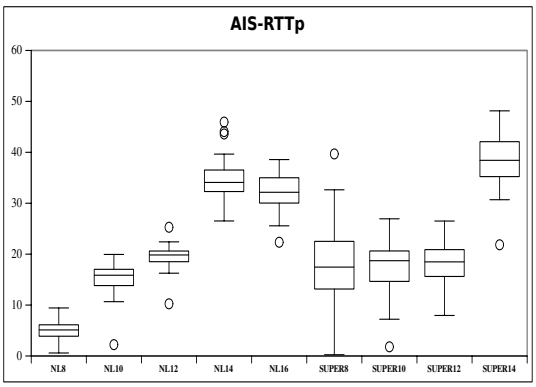

Fig. 10. $AIS - RTT_p$ Box-plot

the processor configurations used to run the tests. For the hardest problem (NL16), one run of our algorithm required 169 seconds on Corei7-720 CPU for 5000 iterations.

C. Relaxed TTP instances

In table X we report the best results obtained using our approach for solving NL instances and the best known solutions. The results has been obtained using 100 runs. The parameter values obtained by tuning are clonation rate and clonation factor equals one and replace rate equal 0.3. We have also found feasible solutions for the bigger NL instances with the results in table XI.

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

	NL8	NL10	NL12	NL14	NL16
PBSA [8]	-	-	110729	188728	261687
TTSA [4]	39721	59583	111248	188728	263772
TS [6]	-	59583	111483	190174	270063
L. HYP [12]	39721	59583	112873	196058	279330
SA + HC [5]	39721	59821	115089	196363	274673
ACO [11]	39721	60399	115871	203205	292214
VNS-CS [7]	41928	65193	120906	208824	287130
ANT HYP. [10]	40361	65168	123752	225169	321037
This Paper	39721	60340	119872	209558	313043
Best known	**39721**	**59436**	**110729**	**188728**	**261687**
GAP %	*0*	*3.2*	*8.9*	*9.4*	*13.4*

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

	S8	S10	S12	S14
L. Hyp. [12]	182409	318421	467267	599296
This Paper	182409	318363	485678	646521
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 [8]	-	-	1501	2491	12858
TTSA [4]	1639	27818	150328	77587	476191
TS [6]	-	1445	1330	2937	5885
SA + HC [5]	52497	67619	82322	96822	111935

TABLE IX
COMPUTER CONFIGURATIONS

PBSA [8]	Cluster of 60 Intel dualcore, dualprocessor Dell
TTSA [4]	AMD Athlon 64, 2Ghz
TS [6]	AMD Athlon, 1.5 Ghz
SA + HC [5]	Pentium 4, 2.53 Ghz 512 RAM

TABLE X
COMPARISON WITH BEST-KNOWN SOLUTIONS FOR RTTP

	NL4, k=1	NL4, k=2	NL4, k=3
Best Known	8160	8160	8044
This Paper	8160	8160	8044
	NL6, k=1	NL6, k=2	NL6, k=3
Best Known	23124	22557	22557
This Paper	23124	22557	22557
	NL8, k=1	NL8, k=2	NL8, k=3
Best Known	39128	38761	38670
This Paper	39128	38761	38680

TABLE XI
NEW FEASIBLE SOLUTIONS FOR RELAXED NLX INSTANCES

NL10	k=1	k=2	k=3
This Paper	59425	59373	59582
NL12	k=1	k=2	k=3
This Paper	117680	119067	116082
NL14	k=1	k=2	k=3
This Paper	209616	209137	205690
NL16	k=1	k=2	k=3
This Paper	307125	300188	297426

As far as we know these are the first reported results for these instances. We can not prove that they are the optimal values, but they are feasible solutions. We have also implemented an algorithm for checking the feasibility of the solutions found by our approach, using various k values.

VI. CONCLUSION

We have introduced an immune-inspired algorithm to solve instances of the relaxed traveling tournament problem which is a new challenge for optimization algorithms. 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 two moves that do not depend on an immune approach. The moves can be used by other techniques to reduce the travel costs when the algorithm makes a change and to use the knowledge of the team byes. We note that the algorithm requires different parameter values that depend on both the size of the instance and the type of problem (number of byes) at hand. Our approach outperforms the algorithm CLONALG in the tested instances. Moreover, $AIS - RTT_p$ has been able to find eight of the nine best known solutions for the relaxed travel tournament instances. We also give in this paper new bounds for the bigger NL instances. 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

This work is partially supported by Centro Científico Tecnológico de Valparaíso (CCT-Val) FB0821. The authors would like to thank Dr. Xavier Bonnaire for his helpful remarks.

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] Challenge Travel Tournament Problems(Relaxed), January 17, 2011 from mat.gsia.cmu.edu/TOURN/relaxed/

[3] 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.

[4] 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.

[5] 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, Elsevier, 2006.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] M-C. Riff, M. Zuniga and E. Montero, *A graph-based immune inspired constraint satisfaction search*, Neural Computing and Applications Journal, volume 18, number 8 pages 1133-1142, Springer 2010.