

Applying Cooperative Coevolution to Compete in the 2009 TORCS Endurance World Championship

Luigi Cardamone, Daniele Loiacono and Pier Luca Lanzi *Member IEEE*

Abstract—The TORCS Endurance World Championship is an international competition in which programmers develop and tune their drivers to race against each other using TORCS, a state-of-the-art car racing simulator. In this work, we applied evolutionary computation to develop a driver for the 2009 edition of this competition. In particular, we focused on the optimization of the car setup of an existing driver (the winner of the 2008 edition) and applied cooperative coevolution to evolve the best car setup for the qualifying rounds of each leg of the championship.

After 10 legs involving 12 teams, our driver was able to reach the 4th position in the final standings. We believe that this is a very promising result especially if we consider that we only focused on the car setup and the other teams participated also to most of the previous four editions (gaining much domain knowledge). Overall, our results show that cooperative coevolution can be very effective in this complex optimization task producing setups that can be competitive with respect to the ones tuned by human experts. Therefore, our results also suggest that simple evolutionary computation might represent a helpful tool to human designers for improving the performance of already heavily tuned drivers.

I. INTRODUCTION

Several recent works [1], [2], [3] showed that evolutionary computation (EC) can be effectively applied to develop more interesting and challenging computer games. At the same time, computer games represent ideal testbeds which provide very complex and realistic environments. Accordingly, several scientific competitions dedicated to computer games have been recently organized at major international conferences in the evolutionary computation field [4], [5], [6], [7].

In this work, we applied evolutionary computation, specifically cooperative evolution, to improve the performance of a driver that participated to the 2009 TORCS Endurance World Championship [8]. This is a competition based on The Open Racing Car Simulator (TORCS) [9], a state-of-the-art car racing simulator, where many human competitors develop programmed drivers to race against each other on a series of Grand Prix. The championship resembles the characteristic of a real racing championship (e.g. Formula One): there are several legs, each one involving a different track; each leg has a qualifying session and a race session; during the qualifying session, the drivers race alone and

the results determine their position on the starting grid of the race session; during the race session the drivers race all together and are scored on the basis of the order of arrival. In the championship, drivers have to deal with many challenging situations (e.g. overlapping, pit stops, etc.). The championship sees the participation of several programmers (typically domain experts) who provide a very challenging context.

To participate to the championship, we started from an existing driver, called SIMPLIX, that won the 2008 TORCS Endurance World Championship. To improve the performance of this driver, we applied cooperative evolution to optimize the car setup, which represents a very crucial issue in car racing. In fact, as in Formula One, even minor changes in wing angles or suspension stiffness of a racing car can make the difference between win and lose the race.

The results of the championship show that cooperative evolution was effective to optimize the car setup. Although we started from a very fast driver (the winner of the previous edition of the championship) we achieved a performance improvement of several seconds that allowed us to reach a remarkable score in the final standings: the 4th place against 12 participants.

II. RELATED WORK

Evolutionary computation has been applied to computer games both for learning and for optimization. In this section, we present a brief overview of the published works that applied evolutionary-based optimization to computer games. For the applications to learning, we refer the reader to [10] for a broad overview and to [11], [3] for car racing games.

Evolutionary-based optimization has been applied to several computer games. The most interesting examples are probably the work of Cole et al. [12], who applied genetic algorithms to the game Counter Strike, and the work of Parker and Parker [2] who applied a similar approach to X-pilot¹.

Optimization has been applied to racing games with two main goals: to optimize the behavior of a hand-coded driver or to optimize the car setup. Most of the works focused on the optimization of a driver [13], [14], [15]. In particular, Butz et al. [13] optimized the parameters of a driver for the game TORCS using CMA-ES. de la Ossa et al. [14] applied ant colony optimization (ACO) to tune the trajectory followed by a driver in a simple 2D racing game. Tanev et al. [15] applied

Luigi Cardamone (cardamone@elet.polimi.it), Daniele Loiacono (loiacono@elet.polimi.it) and Pier Luca Lanzi (lanzi@elet.polimi.it) are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy.

Pier Luca Lanzi (lanzi@illgal.ge.uiuc.edu) is also member of the Illinois Genetic Algorithm Laboratory (IlligAL), University of Illinois at Urbana Champaign, Urbana, IL 61801, USA.

¹www.xpilot.org/

genetic algorithms to optimize the parameters of a real radio-controller car and compared its performance against human players.

At best of our knowledge, only Wloch et al. and the participants the Contest II of the Simulated Car Racing competition [4] held at GECCO-2009 tackled the problem of optimizing a car setup. Wloch et al. [1] applied a standard genetic algorithm to optimize the setup of a Formula One car in the commercial game Formula One Challenge. In the second contest [4] of the GECCO-2009 Simulated Car Racing competition, different methods were compared on the optimization of a car setup in TORCS [9]. The task was challenging since each evaluation was noisy and the amount of evaluation time available was limited. The competition received two entries while an additional entry was made available by the organizers. As no paper was published about the competition, we refer the interest reader to the competition website [4], to the manual [16] and to <http://cig.dei.polimi.it> for further information.

The work we present here differs from [1] in several respects: (i) we apply cooperative coevolution ([1] applied genetic algorithms); (ii) we compare several optimization algorithms ([1] considered only one); (iii) we validated the results on 10 tracks ([1] considered only two tracks); (iv) we apply evolutionary computation to develop a driver for a challenging competition where our performance are compared with the performance achieved by several expert players (in [1] the comparison with other setups, tuned by expert human players, is very limited).

The comparison with the results of the Simulated Car Racing competition is hard since no published work is available and the problem setting is much different (noisy evaluations, limited evaluation time, etc.).

III. TORCS



Fig. 1. A screenshot from TORCS.

The Open Racing Car Simulator (TORCS) [9] is a state-of-the-art open source car racing simulator which provides a sophisticated physics engine, full 3D visualization, several

tracks and models of cars, and different game modes (e.g., practice, quick race, championship, etc.). The car dynamics is accurately simulated and the physics engine takes into account many aspects of racing cars such as traction, aerodynamics, fuel consumption, etc.

Each car is controlled by an automated driver or *bot*. At each control step, a bot can access the current game state, which includes several information about the car and the track as well the information about the other cars on the track, and can control the car using the gas/brake pedals, the gear stick, and steering wheel. The game distribution includes several programmed bots which can be easily customized or extended to build new bots.

All the experiments reported in this paper have been carried out using the version 1.3.1 of TORCS.

IV. THE TORCS ENDURANCE WORLD CHAMPIONSHIP

The TORCS Endurance World Championship [8] is an international competition based on TORCS [9] in which several programmed drivers compete on a series of tracks. Each participant has one or more teams each consisting in two programmed drivers (called robots in the championship regulation) that control the same car model. The championship consists of ten legs each one taking place every 3-4 weeks. Each leg is divided into two parts: the qualifying session and the race session. During the qualifying session each robot drives for three laps and the performance is the best lap time. The results of the qualifying session are used to set up the starting grid of the race session in which the robots have to drive for approximately 500km. Due to the length of the race session, drivers have to deal with several challenging situations: pit stops, overlapping, etc.

At each leg, points are assigned to drivers, on the basis of their order of arrival: 20 points to 1st, 18 to the 2nd, 17 to the 3rd and so on. The total points gathered in all the legs form the final standings. The goal of the championship is to develop the best robot that achieves the highest score in the driver standings and in the team standings.

Several participants to the championship are domain expert in several related topic (e.g. automotive, mechanics, game programming, etc.). Thus, the championship is a very challenging contest to evaluate the potential of evolutionary computation.

A. Championship Structure

The championship is organized using the TORCS Racing Board (TRB [8]), a web platform containing all the information about the current leg (track name, deadlines, participants, etc), about the results of the completed legs, and about the current standings.

To provide a fair scoring process (and also to avoid cheating) each participant download and install all the other drivers registered to the current leg, simulates the race locally on his own machine and then uploads the results to TRB. The final performance of each robot is calculated as the median of the results submitted by the participants. This

mechanism prevents cheating since a competitor has no interest to downgrade the other cars or to improve his car as the final result is computed using the median. Then, the final results of the current leg and the updated standings are published on TRB. Note that the participants know the tracks used for the championship and therefore the robots are usually tailored for the current leg.

B. Championship Rules

Each participant must attain to the following rules: (i) each team has two cars of equal type that cannot be changed during the championship; (ii) each car of the team is allowed to have its own setup; (iii) the robots source must be licensed under the GPL; (iv) the source code must compile on g++ 3.0 or later; (v) the behavior of the robots must be deterministic; (vi) the robots are allowed to use limited computational resources².

C. The 2009 Championship

The first TORCS Endurance World Championship was held in 2005 and so far five editions have been organized. The last edition was held from May 21th 2009 to December 16th 2009 and 12 teams participated. The championship was based on TORCS-1.3.1 and involved the 10 tracks depicted in Table I. We participated to the 2009 championship with the team "the PoliMi Racing Team Reloaded". All the results are available at TRB webpage [8].

D. The Robot SIMPLIX

To participate to the competition is necessary to provide a programmed driver. In this work, as starting point, we used an existing robot, called SIMPLIX, that won the championship held in 2008. The source code of SIMPLIX is publicly available at [17]. The robot SIMPLIX is very sophisticated and its behavior can be divided into three parts: (i) one that generates the trajectories; (ii) one that follows a trajectory; (iii) one that handles special situations. The generation of trajectories is performed during the initialization of the robot, before the race starts. In this phase, the robot generates a fast trajectory using some geometrical heuristics and for each point of the track the maximum speed allowed is computed. In addition, SIMPLIX generates a trajectory close to left border of the track and a trajectory close to the right border. These two additional trajectories are used to overtake. During the race, SIMPLIX controls the steer, the brake and the accelerator in order to follow the optimal trajectory within the computed speed limits. In addition, SIMPLIX has specific behaviors to deal with overtaking, overlapping, crash recovery and pit stops.

V. COOPERATIVE COEVOLUTION

Cooperative coevolution [18] is a family of algorithms that use modularity to reduce the complexity of difficult problems. It is inspired to natural ecosystems where several species compete or cooperate for their survival. When two

or more species compete they tend to produce stronger and stronger species that try to maximize their utility and minimize the utility of the opponents. When species cooperate they aim at maximizing a global utility. In this context, species can be seen as components of a bigger system whose fitness depends on the level of cooperation of its parts.

The general framework of cooperative coevolution was introduced in [18] and can be summarized in the following points: (i) a species represents a subcomponent of a complete solution; (ii) a complete solution is assembled using one member from each species; (iii) utility assignment at the species level is defined in terms of the fitness of the complete solution in which the species members participate; (iv) when required, the number of species should evolve autonomously; (v) the evolution of each species (subpopulation) is handled independently by a standard genetic algorithms [19]; (vi) each species can have a different genetic representation. Note that, the general framework does not specify how to select the members of the species in order to build complete solutions. This and other problem dependent aspects are left to the implementation.



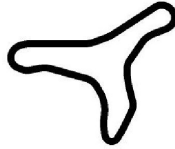
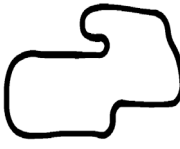


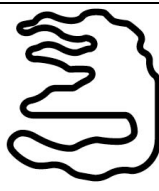


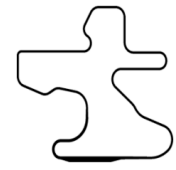
The cooperative coevolutionary algorithm used in this paper is based on [20] and it is outlined as Algorithm 1. In this work, we need to optimize a vector of real-valued parameters and so we represented each species as a numerical value. In particular, we used exactly one species to encode each parameter so to modularize the problem with the finest granularity allowed.

The inputs of Algorithm 1 are (i) the number n of sub-populations, which in our case corresponds to the number of parameters, and (ii) the size m of the populations. The overall population P is represented with a matrix with m rows and n columns as shown in Figure 2. At the beginning each sub-population is initialized independently using a uniform random distribution (line 2). Then, n complete solutions are constructed taking a single member from each sub-population. A complete solution is a row of the matrix P (line 5) which is evaluated by computing its fitness value. After the evaluation the top 25% complete solutions are copied into the matrix Top (line 8). Then the mutation and the crossover operators are applied independently to each sub-population of Top (lines 9-10). The mutation is a uniformly distributed variation. The crossover operator randomly selects two parents and computes the average value that replaces one of them. From the overall population P the worst 25% individuals are removed (line 11). The remainder of the population is randomly permuted applying a shuffle operator to each sub-population (line 12). In this way, each value can be moved from an individual to another and can cooperate with a new set of values (see Figure 2). Finally the recombined individuals contained in Top are added to P replacing those individuals removed in the previous step (line 13). With respect to [20], our approach uses elitism, that is, the best individual is copied without any modification to the next generation.

Note that each sub-population evolves independently: no

²The rules state that on a system with 256MB RAM, 1GHz CPU it should be possible to run 10 robots fluently (average > 40FPS).

TABLE I
TRACKS USED IN THE 2009 TORCS ENDURANCE WORLD CHAMPIONSHIP

Forza	E-Track 4	Dirt 3	Ruudskogen	E-Track 1
				
Aalborg	Alpine 1	CG track 2	Street 1	Alpine 2
				

Algorithm 1 Cooperative Coevolution.

```

1: procedure EVOLUTION( $m, n$ )
2:    $P = \text{Init}(P^1, P^2, P^3, \dots, P^n);$            ▷ Randomly
   initialize each sub population
3:   repeat
4:     for  $j = 1$  to  $m$  do
5:        $X_j = (P_j^1 P_j^2 P_j^3 \dots P_j^n)$ 
6:       Evaluate( $X_j$ )
7:     end for
8:      $Top \leftarrow \text{SelectBestQuarter}(P);$ 
9:     Mutate( $Top^1, Top^2, Top^3, \dots, Top^n$ );
10:    Crossover( $Top^1, Top^2, Top^3, \dots, Top^n$ );
11:    RemoveWorstQuarter( $P$ );
12:    Shuffle( $P^1, P^2, P^3, \dots, P^n$ );
13:    Add( $P, Top$ );
14:  until Maximum Number of Generations Reached
15: end procedure

```

exchange of genetic material occurs between two different sub-population. The only factor that acts as common pressure and enforce cooperation is the fitness function.

VI. EXPERIMENTAL SETUP

In this work, we applied cooperative coevolution to optimize the setup of a racing car that participated to the 2009 TORCS Endurance World Championship. In particular, we focused on the optimization of the performance for the qualifying session. As the starting point, we used the bot SIMPLIX with the car model *car1-trb1*, and we optimized the car parameters for each track of the championship.

TORCS allows to tune more than 50 car parameters, as in real racing cars: wings, suspensions, wheels, brakes, etc. Among the parameters available in a TORCS car, we chose the subset of parameters listed in Table II. On the one hand we focused on the parameters that most affect the performance of the car. On the other hand, we were forced to restrict our search on a subset of parameters because of some

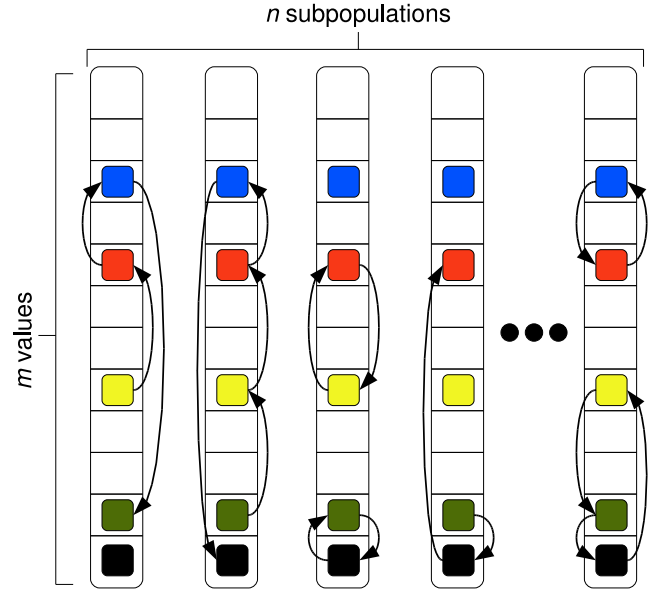


Fig. 2. Cooperative Coevolution.

practical problems: TORCS expects standard and meaningful parameter settings and when unfeasible configurations are used the game may easily crash or behave in an unpredictable manner. To overcome this issue, in the optimization, we did not consider the parameters that could cause problems.

For the optimization, we applied the cooperative coevolutionary algorithm (Section V) reported as Algorithm 1 using 16 subpopulations, one for each car parameter. We evaluated each configuration of parameters letting the robot SIMPLIX to drive 2 laps and the fitness was computed as the lap time achieved in the second lap.

All the evaluations of the individuals are performed in one long race. Thus, laps 1-2 were used to evaluate the first individual, laps 3-4 for the second one, and so on. To enable subsequent evaluations on a single race, we modified TORCS, to allow the change of parameters while the car is

TABLE II
CAR PARAMETERS CONSIDERED IN THE OPTIMIZATION PROCESS.

Index	Parameters	Unit	Min	Max
1	Rear wing angle	deg	0	18
2	Front wing angle	deg	0	12
3	Front-rear brake repartition	SI	0.3	0.7
4	Brake max pressure	kPa	100	150000
5	Front anti-roll bar spring	lbs/in	0	5000
6	Rear anti-roll bar spring	lbs/in	0	5000
7	Front wheel ride height	mm	100	300
8	Front wheel toe	deg	-5	5
9	Front wheel camber	deg	-5	-3
10	Rear wheel ride height	mm	100	300
11	Rear wheel camber	deg	-5	-2
12	Front suspension fast bump	lbs/in	0	10000
13	Front suspension fast rebound	lbs/in	0	10000
14	Rear suspension fast bump	lbs/in	0	10000
16	Rear suspension fast rebound	lbs/in	0	10000

driving: this feature is not available in TORCS to prevent cheating. To achieve more precise evaluations, we further modified TORCS to disable fuel consumption and damages. In this way, we limited the noise that can affect the fitness function and prevented the robot from going to pit stops for low fuel or high damages.

VII. PRELIMINARY ANALYSIS

At first, before taking part to the championship, we performed a preliminary analysis to compare cooperative coevolution (Section V) against the well-known CMA-ES [21] and a real-coded genetic algorithm (GA) [19], [22]. In this analysis, we considered most of the tracks used in the 2009 TORCS Endurance World Championship.

Cooperative coevolutionary (outlined as Algorithm 1) was applied with a mutation range of 0.1 and a crossover probability of 0.1. The genetic algorithm was applied using tournament selection with a tournament size 2, single point crossover, mutation probability $1/n$, mutation range of 0.1 and crossover probability of 0.1. The parameters of both algorithms were set empirically. CMA-ES was applied using the C implementation available at [23] with all the default settings. All the three optimization algorithms were applied with a population size of 100 individuals and with a generation limit of 50.

The results are reported in Table III which shows the performance, measured as the lap time in seconds, of the best individual found by each optimization technique. Statistics are averages over 10 runs. The results show that cooperative coevolution outperforms the other techniques in 7 out of 9 tracks. In many cases, the difference with all the other algorithms is remarkable: in fact in car racing competitions (real or simulated) even half of a second represents an

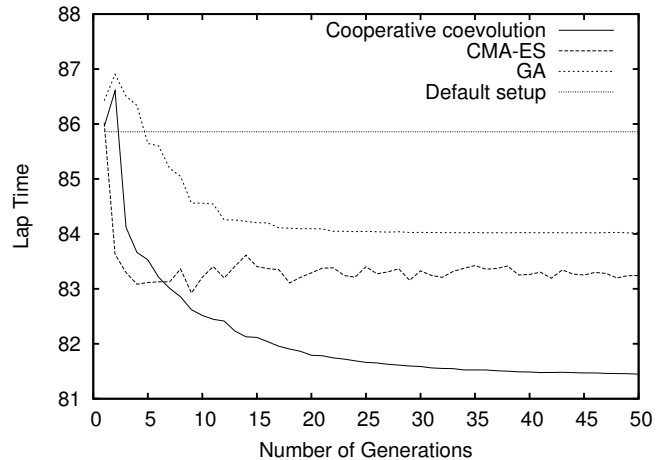


Fig. 3. Cooperative Coevolution, CMA-ES, and GA applied to the track Forza: best fitness over the number of generations. Curves are averaged over 10 runs.

important performance difference. From the results, it is possible to note that the genetic algorithm performs worse. The difference between the genetic algorithm and the other more sophisticated algorithms gives a measure of the complexity of the problem. CMA-ES performs reasonably well on all the tracks and significantly outperforms the cooperative coevolution only in the track Alpine 1.

Interesting information for comparing the three techniques can be found from the analysis of Figure 3, Figure 4 and Figure 5 that show the trends of the evolution in three different tracks. The figures report the performance (i.e., the lap time in seconds) of the best individual evolved by the three techniques in each generation. As a reference, we also report the performance of default car setup provided with SIMPLIX. All the curves are averages over 10 runs. In the optimization of track Forza (Figure 3) all the three techniques quickly reach and surpass the performance of the default setup. The CMA-ES quickly finds its best performance before generation 10 and then it cannot achieve further improvements. The GA is much slower and reaches its best performance after 25 generations. This trend suggests that the GA tends to be stuck in local minima. In contrast, cooperative coevolution continues to improve its performance until the end of the run, outperforming the two other techniques. The optimization of track Street 1 (Figure 4) is more challenging than the previous case. In fact, the best performance of the GA is a very little improvement with respect to default setup provided with SIMPLIX. The optimization of track Aalborg (Figure 5) is even much more difficult: in this case, the performance of the GA is even worse than the performance with the default setup. Both on track Street 1 and on track Aalborg, cooperative coevolution and CMA-ES present the similar behavior found in track Forza: at the beginning, the two techniques are comparable, but then, cooperative coevolution starts to significantly outperform CMA-ES.

TABLE III

BEST PERFORMANCE ACHIEVED BY COOPERATIVE COEVOLUTION, GA AND CMA-ES ON EACH TRACK OF THE CHAMPIONSHIP. RESULTS ARE AVERAGES OVER 10 RUNS

Track	Cooperative Coevolution	GA	CMA-ES
Forza	81.44 \pm 0.09	84.01 \pm 0.84	82.08 \pm 0.68
E-Track 4	93.81 \pm 0.08	96.14 \pm 1.17	94.99 \pm 0.24
Dirt 3	53.58 \pm 0.28	55.47 \pm 0.5	53.25 \pm 0.48
Ruudskogen	56.39 \pm 0.09	57.89 \pm 0.74	57.02 \pm 0.41
E-Track 1	62.34 \pm 0.18	65.65 \pm 2.46	64.01 \pm 1.03
Aalborg	66.22 \pm 0.25	70.08 \pm 1.5	66.68 \pm 0.33
Alpine 1	115.21 \pm 1.29	118.94 \pm 1.55	113.56 \pm 0.38
CG track 2	45.87 \pm 0.03	46.83 \pm 0.72	46.08 \pm 0.11
Street 1	70.49 \pm 0.16	73.36 \pm 2.36	71.24 \pm 0.75

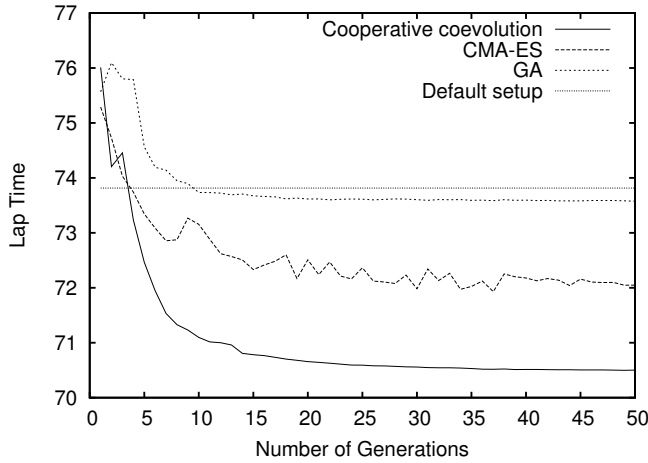


Fig. 4. Cooperative Coevolution, CMA-ES, and GA applied to the track Street1: best fitness over the number of generations. Curves are averaged over 10 runs.

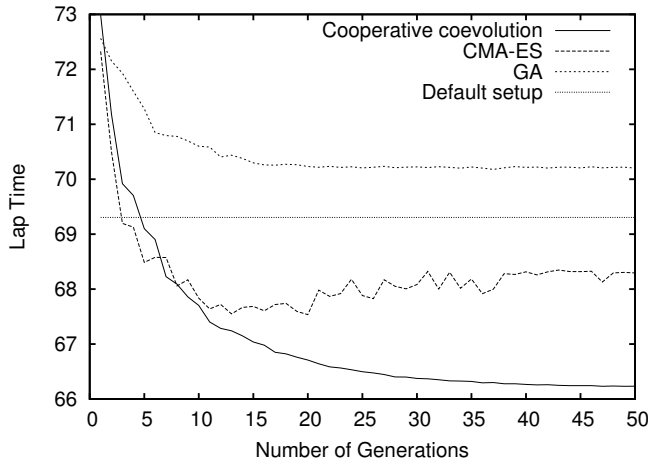


Fig. 5. Cooperative Coevolution, CMA-ES, and GA applied to the track Aalborg: best fitness over the number of generations. Curves are averaged over 10 runs.

VIII. PERFORMANCE IN THE TORCS ENDURANCE WORLD CHAMPIONSHIP

Our preliminary analysis suggests that cooperative evolution is very effective to optimize the setup of a racing car in TORCS. We applied this approach to develop a driver for the 2009 TORCS Endurance World Championship. As starting point, we used the robot SIMPLIX and for each leg of the championship we optimized the default car setup, focusing on the performance during the qualifying session.

The results of the qualifying sessions are presented in Table IV which reports, for each track of the championship (i) the lap times of the robot SIMPLIX with the default setup and with the optimized setup; (ii) the rank achieved from our team in the qualifying session; (iii) the best lap time among all the drivers using the same car of our robot (*car1-trb1*); (iv) the lap time and the name of the best team of the qualifying session. All the results, for both the qualifying and the race sessions together with additional statistics (damage, pit stops, etc.), are reported in details on the TRB website [8].

The results show that cooperative coevolution is effective also in a very competitive contest like the 2009 championship. By applying only the optimization of the car setup, we were able to achieve the top positions of the qualifying standings among the twelve teams participating. In particular, if we compare our performance with the drivers that use our same car model, the results shows that our lap times are faster in 4 tracks out of 10. These results are very promising if we consider that several teams, in addition to the car setup, tune also the behavior of the robot (speed limits and trajectories) specifically for the current track. It is important to mention that, in the track Alpine 2, we achieved the best performance scoring a lap time many seconds faster than any other driver and we set the new record for the track (considering all the championships from 2005 to 2009). From the analysis of this result, we noted that, our driver was able to exploit the high barriers present in Alpine 2 to tackle some difficult turns. The setup optimized with cooperative coevolution resulted in a more stable car that used the collisions with the barriers

TABLE IV

BEST LAPS ACHIEVED FROM SIMPLIX BEFORE AND AFTER THE OPTIMIZATION COMPARED WITH THE RESULTS OF THE 2009 TORCS ENDURANCE WORLD CHAMPIONSHIP

Track	SIMPLIX	Optimized	Rank	Best Car1	Best	Best Team
Forza	1:25.858	1:22.690	8	1:20.308	1:18.754	wdbee racing
E-Track 4	1:42.902	1:34.852	5	1:34.462	1:33.118	wdbee racing
Dirt 3	1:02.912	0:53.984	3	0:57.241	0:53.127	wdbee racing
Ruudskogen	1:01.964	0:55.730	5	0:55.106	0:55.106	Green Monster Racing
E-Track 1	1:05.192	1:01.848	4	1:01.606	1:00.989	wdbee racing
Aalborg	1:09.304	1:06.440	6	1:04.798	1:02.330	wdbee racing
Alpine 1	2:00.868	1:57.029	6	1:56.050	1:55.933	wdbee racing
CG track 2	0:49.636	0:45.675	3	0:46.884	0:45.575	wdbee racing
Street 1	1:13.814	1:10.130	3	1:10.840	1:09.868	wdbee racing
Alpine 2	1:32.682	1:19.768	1	1:24.534	1:19.768	PoliMi Racing Team

to drive much faster in the slow parts of the track. This result really impressed the other participants (see [24]).

To provide a good setup for the racing session, it is important to take into account also the car damage. This is less important during the qualifying where the crucial aspect is the lap time. The situation changes in the race sessions during which, if the total amount of damage is significant, the performance of the car gets worse and the driver is forced to go to pit stop. Thus, for each leg, we analyzed the performance of the driver in the qualifying session. If the optimized setup produced little or no damage the same setup was used also for the racing session. If the amount of damage is significant, we performed another optimization process using a fitness function that took in account also the car damage. The fitness was computed as the lap time minus the damage produced during the evaluation. This provided drivers that were more robust with respect to damage but would have been too slow for the qualifying session.

Finally, in Table V, we report the final results of the championship. The results are very promising, as we reached the 4th position against several teams of cars tuned with human expertise, especially considering that the score is based on the racing session and that we optimized only the car setup. So, simple optimization might help improving the performance of human designed and tuned drivers. In fact, although SIMPLIX has already a rather competitive performance, cooperative evolution allowed to further enhance its lap times of several seconds.

IX. CONCLUSIONS AND FUTURE WORKS

In this work, we applied evolutionary computation to optimize the car setup of our driver for the 2009 TORCS Endurance World Championship. At first, we performed a preliminary analysis and compared three evolutionary techniques (CMA-ES, cooperative evolution and real-coded GAs) in several tracks. The results suggested that cooperative evolution could outperform the other methods and also improve the performance of the driver winner of 2008 championship, which we used as starting point.

TABLE V

TEAM STANDINGS OF THE 2009 TORCS ENDURANCE WORLD CHAMPIONSHIP

Pos	Team	Car Type	Points
1	wdbee racing	Car6	335.5
2	dummy drivers	Car1	310
3	Team JeDI	Car2	291.5
4	the PoliMi Racing Team Reloaded	Car1	229
5	Nexus Six Racing	Car3	192
6	Alba	Car1	147.5
7	Team Dots 2008	Car4	146.5
8	EJRacing	Car5	115
9	Green Monster Racing	Car1	95.5
10	USRobotics	Car7	90.5
11	Berniw Racing	Car5	63.5
12	Hymie Tech	Car4	57.5

Then, we applied cooperative evolution to optimize the car setup for the qualifying session of each leg of the 2009 championship. After each qualifying round, we checked the amount of damage our car suffered. If the car damage was limited the same setup was used also for the race session. Otherwise, if the damage was substantial, we repeated the optimization process to generate a new car setup for the racing session using a fitness function which took into account also the car damage.

Our results show that: (i) in the qualifying session, our lap times are competitive: in particular, when we consider the drivers using our same car model, we note that our lap times are better 4 times out of 10; (ii) in the final standings, we achieved the 4th position against 12 participants; (iii) in some tracks the optimized setup resulted in interesting behaviors that really impressed the other participants [24].

Overall, results suggest that the use of even simple evolutionary computation methods can effectively improve an already high performing human-developed driver tuned with domain knowledge. The methodology applied in this work appears to be a valid and simple tool to help human designers to improve the performance of their drivers.

Although, with the optimization of the car setup, we achieved good results, winning the championship would

have required also the optimization of the driver behavior. Thus, our future work will involve the optimization of other important factors such as the overtaking policy, the pit stop strategy, the shape of the trajectory and the speed associated with each point of the trajectory.

REFERENCES

- [1] K. Wloch and P. J. Bentley, "Optimising the performance of a formula one car using a genetic algorithm," in *Proceedings of Eighth International Conference on Parallel Problem Solving From Nature*, 2004, pp. 702–711.
- [2] G. Parker and M. Parker, "Evolving parameters for xpilot combat agents," in *Proc. IEEE Symposium on Computational Intelligence and Games CIG 2007*, 2007, pp. 238–243.
- [3] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Evolving competitive car controllers for racing games with neuroevolution," in *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2009, pp. 1179–1186.
- [4] "Gecco-2009 simulated car racing competition — contest 2: Optimizing car setup," <http://www.sigevo.org/gecco-2009/competitions.html> and <http://www.sigevo.org/gecco-2009/docs/SCR-GECCO2009-Contest2.pdf>.
- [5] "Simulated car racing competitions." [Online]. Available: <http://cig.dei.polimi.it/>
- [6] "Ieee cec-2009 simulated car racing competition." [Online]. Available: <http://www.cec-2009.org/competitions.shtml>
- [7] "Ieee cig-2009 simulated car racing competition." [Online]. Available: <http://www.ieee-cig.org/cig-2009/competitions/>
- [8] "the torcs racing board website." [Online]. Available: <http://www.berniw.org/trb/>
- [9] "The open racing car simulator website." [Online]. Available: <http://torcs.sourceforge.net/>
- [10] J. Togelius, "Optimization, imitation and innovation: Computational intelligence and games," Ph.D. dissertation, Department of Computing and Electronic Systems, University of Essex, Colchester, UK, 2007.
- [11] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Evolving drivers for torcs using on-line neuroevolution," Illinois Genetic Algorithms Laboratory, Tech. Rep. TR No.: 2009008, 2009.
- [12] N. Cole, S. J. Louis, and C. Miles, "Using a genetic algorithm to tune first-person shooter bots," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004, pp. 139–145.
- [13] M. V. Butz and T. D. Lonneker, "Optimized sensory-motor couplings plus strategy extensions for the torcs car racing challenge," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 317–324.
- [14] L. de la Ossa, J. Gamez, and V. Lopez, "Improvement of a car racing controller by means of ant colony optimization algorithms," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, Dec. 2008, pp. 365–371.
- [15] I. Tanev, M. Joachimczak, and K. Shimohara, "Evolution of driving agent, remotely operating a scale model of a car with obstacle avoidance capabilities," in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2006, pp. 1785–1792.
- [16] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Car setup optimization. competition software manual," Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 2010, Tech. Rep. 2010.1, 2010.
- [17] "Simplic website." [Online]. Available: <http://www.wdbec.gotdns.org:8086/SIMPLIX/SimplicDefault.aspx>
- [18] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization." Springer-Verlag, 1994, pp. 249–257.
- [19] D. Goldberg, *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.
- [20] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *J. Mach. Learn. Res.*, vol. 9, pp. 937–965, 2008.
- [21] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation," in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, May 1996, pp. 312–317.
- [22] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer, 2008.
- [23] "The cma evolution strategy website." [Online]. Available: <http://www.lri.fr/~hansen/cmaesintro.html>
- [24] "the torcs racing board forum." [Online]. Available: <http://www.berniw.org/trb/forum/showthread.php?topicid=3039>