To link to this article : DOI:10.1007/s10710-014-9228-y
URL : http://dx.doi.org/10.1007/s10710-014-9228-y

To cite this version : Sanchez, Stephane and Cussat-Blanc, Sylvain *Gene regulated car driving : using a gene regulatory network to drive a virtual car.* (2014) Genetic Programming and Evolvable Machines, vol. 15 (n° 4). pp. 477-511. ISSN 1389-2576

Gene regulated car driving: using a gene regulatory network to drive a virtual car

Stéphane Sanchez · Sylvain Cussat-Blanc

Abstract This paper presents a virtual racing car controller based on an artificial gene regulatory network. Usually used to control virtual cells in developmental models, recent works showed that gene regulatory networks are also capable to control various kinds of agents such as foraging agents, pole cart, swarm robots, etc. This paper details how a gene regulatory network is evolved to drive on any track through a three-stages incremental evolution. To do so, the inputs and outputs of the network are directly mapped to the car sensors and actuators. To make this controller a competitive racer, we have distorted its inputs online to make it drive faster and to avoid opponents. Another interesting property emerges from this approach: the regulatory network is naturally resistant to noise. To evaluate this approach, we participated in the 2013 simulated racing car competition against eight other evolutionary and scripted approaches. After its first participation, this approach finished in third place in the competition.

Keywords Gene regulatory network · Virtual car racing · Machine learning · Incremental evolution

1 Introduction

The simulated racing car competition (SRC) aims to design a controller in order to race competitors on various unknown tracks. This competition is based on the open-

S. Sanchez (✉) · S. Cussat-Blanc
IRIT - CNRS UMR 5505, University of Toulouse, Toulouse, France
e-mail: stephane.sanchez@irit.fr

S. Cussat-Blanc
e-mail: sylvain.cussat-blanc@irit.fr

source racing car simulator (TORCS[1]). Both scripted and evolutionary approaches can be used to control the virtual car. Within the frame of this competition, we have proposed a new approach based on gene regulation evolved with a genetic algorithm to produce a virtual car controller. Gene regulatory networks are bio-inspired approaches usually used to control virtual cells in developmental models. However, over the past few years, they have been found to be competitive approaches to control agents that have to deal with uncertainty. To our knowledge, this work is the first attempt to use such a controller to drive a virtual car. The competition offers a very unique frame to compare this approach to other ones within a common benchmark. The remainder of this section presents the existing approaches based on an evolutionary process. Further details about them, and about other fully hand scripted, supervised learning or imitation based controllers can be found in [23, 24]

Amongst the existing controllers for virtual car racing games that use an evolutionary process to optimize the driver behavior, we can broadly consider two kind of controllers. The first kind are indirect controllers where the inputs are not directly linked to the outputs of the virtual racing car. The inputs, such as track sensors, angle sensors, and speed sensors, are computed and transmitted to driving policies based on hand-coded rules or heuristics that manage steering and throttle controls. Usually, these controllers already know how to drive but an evolutionary algorithm is used to improve and tune the driving policies in order to obtain a competitive driver. They learn how, or are evolved, to drive better than their initial design. The driver presented in [5], COBOSTAR, is such a controller. It is based on two hand coded driving strategies, one for on-track driving and the other for off-track driving. The parameters of these strategies are then optimized with CMA-ES. The controllers Autopia in [26] and Mr. Driver in [29, 30] are both based on a modular architecture. The modules are hand-coded heuristics or sets of rules in charge of the basic control of the car (gear, steering wheel, opponents management, etc.). A learning module then detects the segments of the track and adapts the target speed of the car to drive as fast as possible. These kind of controllers are efficient and they have been at the top of the SRC competition since 2009.

The second kind of controllers are direct controllers where sensors are directly mapped to the car effectors. These direct controllers actually learn how to drive the car using its sensors and actuators. These direct controllers can be based on evolved artificial neural networks [2, 34], genetic programming [1] or the NEAT algorithm [6, 8, 32, 33]. These methods have produced controllers that are specialized and efficient on one specific track. The one in [32] approaches curves from the outside and then cuts inside to reach the curve apex and maximize its speed. However, they can also produce more generalized controllers that drive on most of the tracks but on a safer race line rather than on the optimal race line of the tracks [6]. Because evolving a direct controller from scratch that can drive on-track and manage all car controls and race events is difficult, these controllers are sometimes mixed with hand-coded policies that modify the controller outputs to handle crash recovery or opponents, or that manage specific controls such as gear handling.

[1] http://torcs.sourceforge.net/.

The controller we present in this paper is a direct controller. Instead of designing complex hand-coded heuristics, we prefer to evolve the controller to drive the car by using a standard genetic algorithm. In this work, the controller is based on a Gene Regulatory Network (GRN). In order to optimize this GRN, we have used an incremental evolution (as in [5, 34]) based on different fitnesses that gradually refine the controller's behavior. The experiments presented in this paper show that this controller is able to drive on every kind of track. The performance of the GRN has also been improved by the means of hand-coded contextual modifications of its inputs to make it drive faster and overtake opponents.

The paper is organized as follows. Section 2 presents gene regulatory networks in general, describing the existing computational models and the problems they are currently handling. This section also introduces the computational model we have used in this work. Section 3 describes how the GRN is connected to the car sensors and actuators and how the GRN is incrementally trained with a genetic algorithm to produce a basic driver. This section also demonstrates the capacity of the GRN to naturally handle noisy sensors without any noise filter in-between the GRN and the sensors. Then, Sect. 4 shows how we biased the GRN's inputs to improve its capacity to drive fast and avoid opponents. A comparitive study follows, showing how the GRN performs in comparison to other approaches involved in the past years competition. A discussion about how the obtained GRN works and about the advantages and the weaknesses of our approach is provided in Sect. 6. Finally, the paper concludes how this approach could be improved to produce a more efficient car controller and how the GRNs are becoming a competitive alternative to other common evolutionary approaches.

2 Gene regulatory network

Gene regulatory networks (GRNs) are biological structures that control the internal behavior of living cells. They regulate gene expression by enhancing and inhibiting the transcription of certain parts of the DNA. For this purpose, the cells use protein sensors dispatched on their membranes; these provide crucial information to guide the cells through their cycle. Figure 1 presents the functioning of the network. Many modern computational models of these networks exist. They are used both to simulate real gene regulatory networks [3, 14, 31] and to control agents [10, 12, 13, 16, 19, 25].

When used for simulation purpose, a GRN is usually encoded within a bit string, as DNA is encoded within a nucleotid string. As in real DNA, a gene sequence starts with a particular sequence, called the *promoter* in biology [21]. In the real DNA, this sequence is represented with a set of four protein: *TATA* where T represents the thymine and A the Adenine. In [31], Torsten Reil is one of the first to propose a biologically plausible model of gene regulatory networks. The model is based on a sequence of bits in which the promoter is composed of the four bits *1010*. The gene is coded directly after this promoter whereas the regulatory elements are coded before the promoter. To visualize the properties of these networks, he uses graph visualization to observe the concentration variation of the different proteins of the

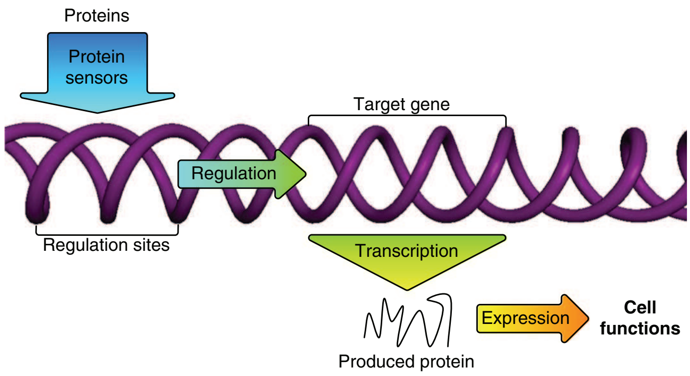

Fig. 1 In real cells, the gene regulatory network uses external signals to enhance or inhibit the transcription of target genes. The expression of these genes will determine the final behavior of the cell

system. He points out three different kinds of behavior from randomly generated gene regulatory networks: stable, chaotic and cyclic. He also observes that these networks are capable of recovering from random alterations of the genome, producing the same pattern when they are randomly mutated. In 2003, Wolfgang Banzhaf formulates a new gene regulatory network heavily inspired from biology [3]. He uses a genome composed of multiple 32-bit integers encoded as a bit string. Each gene starts with a promoter coded by any integer ending with the sequence "XYZ01010101". This sequence occurs with a 2^{-8} probability (0.39 %). The gene following this promoter is then coded in five 32-bits integers (160 bit) and the regulatory elements are coded upstream to the promotor by two integers, one for the enhancing and one for the inhibiting kinetics. Banzhaf's model confirms the hypothesis pointed out by Reil's one; the same properties emerges from his model.

From these seminal models, many computational models have been initially used to control the cells of artificial developmental models [13, 14, 19]. They simulate the very first stage of the embryogenesis of living organisms and more particularly the cell differentiation mechanisms. One of the initial problem of this field of research is the French Flag problem [36] in which a virtual organism has to produce a rectangle that contains three strips of different colors (blue, white and red). This simulates the capacity of differentiation in a spatial environment of the cells. Many models addressed this benchmark with cells controlled by a gene regulatory network [9, 19, 20]. More recently, gene regulatory networks have proven their capacity to regulate complex behaviors in various situations: they have been used to control virtual agents [12, 18, 25] or real swarm or modular robots [10, 16].

2.1 Our model

The gene regulatory network used to control a virtual car in this paper is a simplified model based on Banzhaf's model. It has already been successfully used in other applications. It is capable of developing modular robot morphologies [10], producing 2-D images [11], controlling cells designed to optimize a wind farm layout [35] and controlling reinforcement learning parameters in [17]. This model has been designed for computational purpose only and not to simulate a biological network.

This model is composed of a set of abstract proteins. A protein a is composed of three tags:

- the *protein tag* id_a that identifies the protein,
- the *enhancer tag* enh_a that defines the enhancing matching factor between two proteins, and
- the *inhibitor tag* inh_a that defines the inhibiting matching factor between two proteins.

These tags are coded with an integer in $[0, p]$ where the upper bound p can be tuned to control the precision of the network. In addition to these tags, a protein is also defined by its concentration that will vary over time with particular dynamics described later. A protein can be of three different types:

- *input*, a protein whose concentration is provided by the environment, which regulates other proteins but is not regulated,
- *output*, a protein with a concentration used as output of the network, which is regulated but does not regulate other proteins, and
- *regulatory*, an internal protein that regulates and is regulated by others proteins.

With this structure, the dynamics of the GRN are computed by using the protein tags. They determine the productivity rate of pairwise interaction between two proteins. For this, the affinity of a protein a for another protein b is given by the enhancing factor u^+_{ab} and the inhibiting factor u^-_{ab} calculated as follows:

$$u^+_{ab} = p - |enh_a - id_b|; \quad u^-_{ab} = p - |inh_a - id_b| \tag{1}$$

The proteins are then compared pairwise according to their enhancing and inhibiting factors. For a protein a, the total enhancement g_a and inhibition h_a are given by:

$$g_a = \frac{1}{N} \sum_b^N c_b e^{\beta u^+_{ab} - u^+_{max}}; \quad h_i = \frac{1}{N} \sum_b^N c_b e^{\beta u^-_{ab} - u^-_{max}} \tag{2}$$

where N is the number of proteins in the network, c_b is the concentration of the protein b, u^+_{max} is the maximum observed enhancing factor, u^-_{max} is the maximum observed inhibiting factor and β is a control parameter which will be detailed hereafter. At each timestep, the concentration of a protein a changes with the following differential equation:

$$\frac{dc_a}{dt} = \frac{\delta(g_a - h_a)}{\Phi}$$

where Φ is a normalization factor to ensure that the total sum of the output and regulatory protein concentrations is equal to 1. β and δ are two constants that influence the reaction rates of the network. β affects the importance of the matching factors and δ is used to modify the production level of the proteins in the differential equation. In summary, the lower both values are, the smoother the regulation is; the higher the values are, the more sudden the regulation is.

Figure 2 summarizes how the model functions. The edges represent the enhancing (in green) and inhibiting (in red) matching factors between two proteins. Their thickness represents the distance value: the thicker the line, the closer the proteins.

3 Using a GRN to drive a virtual car

3.1 Linking the GRN to the car sensors and actuators

The GRN can be seen as any kind of computational controller: it computes inputs provided by the problem it is applied to and it returns values to solve the problem. To use the gene regulatory network to control a virtual car, our main wish is to keep the connection between the GRN and the car sensors and actuators as simple as possible. In our opinion, the approach should be able to handle the reactivity necessary to drive a car, the possible noise of the sensors and unexpected situations. The car simulator provides 18 track sensors spaced 10° apart and many other sensors such as car fuel, race position, motor speed, distance to opponents, etc. However, in our opinion, all of the sensors are not required to drive the car. Reducing the number of inputs directly reduces the complexity of the GRN optimization. Therefore, we have selected the following subset of sensors provided by the TORCS simulator:

- 9 track sensors that provide the distance to the track border in 9 different directions,
- longitudinal speed and transversal speed of the car.

Figure 3 represents the sensors used by the GRN to drive the car. Before being computed by the GRN, each sensor value is normalized to $[0, 1]$ with the following formula:

$$norm(v(s)) = \frac{v(s) - min_s}{max_s - min_s} \tag{3}$$

where $v(s)$ is the value of sensor s to normalize, min_s is the minimum value of the sensor and max_s is the maximum value of the sensor.

Once the GRN input protein concentrations are updated, the GRN's dynamics are run one time in order to propagate the concentration modification to the whole network. The concentrations of the output proteins are then used to regulate the car actuators. Four output proteins are necessary: two proteins o_l and o_r for steering (left and right), one protein o_a for the accelerator and one o_b for the brake. The final values provided to the car simulator are computed as follow:

$$steer = \frac{c(o_l) - c(o_r)}{c(o_l) + c(o_r)} \tag{4}$$

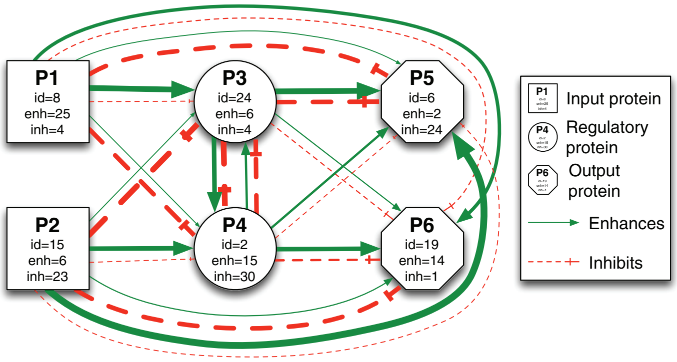

Fig. 2 Graphical representation of a GRN: the nodes are the proteins and the edges represents the enhancing and inhibiting affinity between two proteins. The bigger the edges, the closer the proteins (Color figure online)

$$accel = \begin{cases} 0 & \text{if ab} <= 0 \\ ab & \text{otherwise} \end{cases} \tag{5}$$

$$\begin{gathered} brake = \begin{cases} -ab & \text{if ab} <= 0 \\ 0 & \text{otherwise} \end{cases} \\ \text{with ab} = \frac{\mathrm{c(o_a) - c(o_b)}}{\mathrm{c(o_a) + c(o_b)}} \end{gathered} \tag{6}$$

where *steer* is the final steering value of the car in $[-1, 1]$, *accel* is the final acceleration value in $[0, 1]$, *brake* is the final brake value in $[0, 1]$, $c(o_*)$ is the concentration of the output protein o_*. Figure 4 shows the connection of the GRN to the virtual car.

Finally, the gear value is hand-written as it is a very simple script to develop; when the motor is over a given threshold that depends of the current gear, the driver shifts up. Under another threshold, the driver shifts down. The thresholds are detailed in Table 1.

Whereas other approaches use a noise reduction filter in addition to the standard anti-locking braking system (ABS) and the traction control systems (TCS), the GRN approach does not need any noise filter: it is naturally noise-resistant. The ABS and TCS are switched on because they provide a large support in the braking and acceleration zones. The impact of noise on the GRN reaction is detailed in Sect. 3.4. The code of the GRNDriver is avalailable online on the SRC competition http://scr.geccocompetitions.com. However, some improvements (minor bug corrections) have been made for this particular paper.

3.2 GRN genome

Before it can drive, the regulatory network needs to be optimized. In this work, we use a standard genetic algorithm to optimize the GRN's protein tags, enhancing tags and inhibiting tags. The GRN can be easily encoded in a genome. The genome contains two independent chromosomes. The first one is defined as a variable length

Fig. 3 Sensors of the car connected to the GRN. The *red plain arrows* are used track sensors whereas the *gray dashed* ones are the track sensors also available in the simulator but not used by the GRN. The *plain arrows Speed X* and *Speed Y* are respectively the longitudinal and the transversal car speeds (Color figure online)

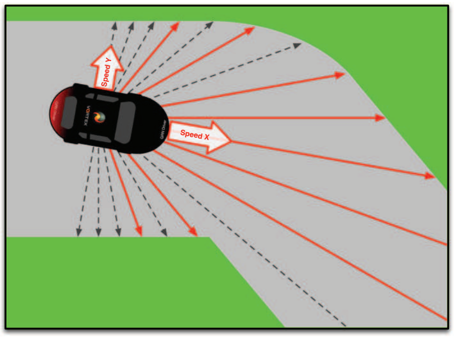

Fig. 4 The GRN uses 9 track sensors and the longitudinal and transversal speeds to compute the steering, the acceleration and the brake of the car

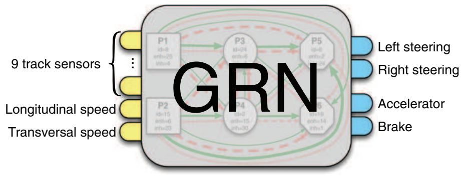

Table 1 Motor speed thresholds to shift down and up a gear

Current gear	Shift down threshold (rpm)	Shift up threshold (rpm)
1	-	9,500
2	4,000	9,500
3	6,300	9,500
4	7,000	9,500
5	7,300	9,000
6	7,300	-

chromosome of indivisible proteins. Each protein is encoded with three integers between 0 and p that correspond to the three tags. In this particular work, p is set at 32 and the genome proteins are organized with the input proteins first, followed by the output proteins and then regulatory proteins. The inputs and outputs presented in the previous section will be always be linked to the same protein, as represented in Fig. 5.

This chromosome requires particular crossover and mutation operators (represented in Fig. 6):

- a *crossover* can only occur between two proteins and never between two tags of the same protein. This ensures the integrity of both subnetworks when the GRN is subdivided into two networks. When assembling another GRN, local connections are kept with this operator and only new connections between the two networks are created.
- three *mutations* can be equiprobably used: add a new random regulatory protein, remove one protein randomly selected in the set of regulatory proteins, or mutate a tag within a randomly selected protein.

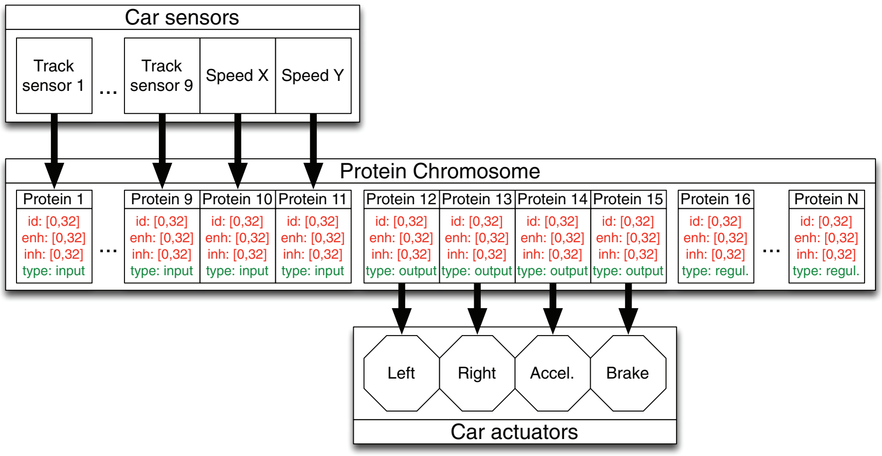

Fig. 5 Organization of the protein chromosome and link to the car sensors and actuators: the tags (in *red*) are evolved by the genetic algorithm whereas the types (in *green*) are fixed and always plugged to the same car sensors (for input proteins) and the same car actuators (for output proteins) (Color figure online)

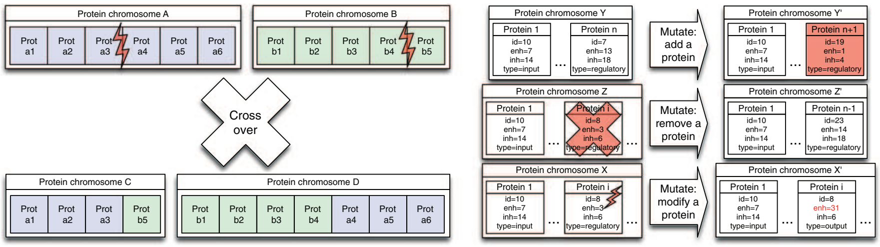

Fig. 6 Crossover and mutation operators applied to the protein chromosome. A crossover (on the *left-hand side*) can only occur between two proteins and a mutation (on the *right-hand side*) consists of adding, removing or changing a protein

A second chromosome is used to evolve the dynamics variables β and δ. This chromosome consists of two double-precision floating point values and uses the standard mutation and crossover methods. These variables are evolved in the interval $[0.5, 2]$. Values under 0.5 produce unreactive networks whereas values over 2 produce very unstable networks. These values are chosen empirically through a series of test cases.

3.3 Incremental evolution

In order to optimize the GRN to drive a car, we use an incremental evolution in three stages.[2] During these stages, the same parameters have been used to tune the

[2] Videos of this evolution are available online: http://www.irit.fr/~Sylvain.Cussat-Blanc/GRNDriver/index_en.php.

genetic algorithm. Only the fitness function is modified. The genetic algorithm parameters are:

- Population size: 500,
- Mutation rate: 15 %,
- Crossover rate: 75 %,
- GRN Size: [4, 20] regulatory proteins plus inputs and outputs.

3.3.1 Stage 1: learning to drive on one simple track

The first stage consists of training the GRN to drive as far as possible, with a minimum speed, on one track. We use CGSpeedway, the left-hand side track of Fig. 7, which is simple with long turns and straight lines. In our opinion, this track is interesting for learning to drive. It is a relatively easy track with long fast turns and with fast straight lines to learn how to steer and to accelerate, and with more difficult short turns to learn how to slow down and to brake. Each GRN is tested on this track for 31 km (about 10 laps) maximum. The simulation is stopped as soon as the car leaves the track or gets damaged (by hitting a rail for example). To ensure the car is driving fast enough, we use a ticket system in which the GRN must cover $500 * nLap$ meters per 1000 simulation steps, where $nLap$ is the current lap number. This pushes the GRNs that go far to accelerate. If a GRN cannot reach this objective, the simulation is stopped. When the simulation ends, the fitness function is given by the distance covered by the GRN along the central line of the track. If a GRN has traveled all 31 km, a bonus is added. The bonus is inversely proportional to the number of simulation steps needed to completed the race.

The top curve in Fig. 8 presents convergence of the genetic algorithm with this fitness. In order to avoid plateauing, we have implemented a restart function which renews the entire population with the best individual, 25 individuals mutated from the best one and 474 new genomes. The effects of the restart function can clearly be seen on the convergence curve with a drastic drop of the fitness average, pointed by the symbol (a).

In this convergence curve, five stages clearly appear. The first stage, denoted (b), represents the time to learn to accelerate and to steer to avoid the track border of the very first turn (turn 1 on Fig. 7). The second stage, denoted (c), represents the time needed to learn to steer in order to go through turn 2. Once this is done, the GRN can go through the complex series of turns 3, 4 and 5. At the third stage, denoted (d), the best GRN can finish one lap, but the GRN stops in the second lap between turn 4 and turn 8. The GRN is too slow and is eliminated from the race by the ticket system. The GRN then learns to drive faster until it can finish the second lap. At this point, the ticket system increases the speed pressure on the GRN and the evolution reaches a new stage (e). The best GRNs are once again stuck in turns 3–5 part of the circuit. A smooth optimization of the GRN is observable in stage (f): the GRN optimizes the trajectory in order to increase the car speed and go further. However, it is not sufficient to finish the third lap.

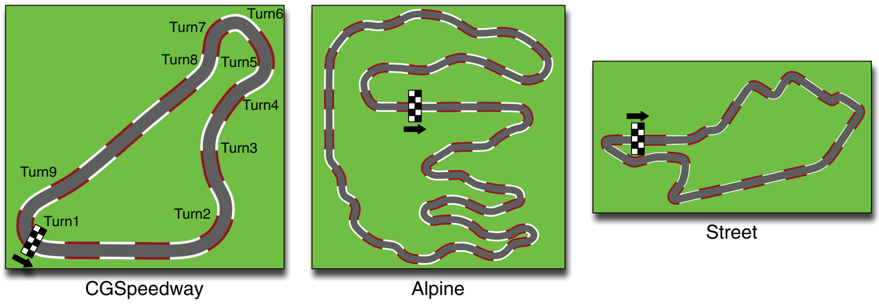

Fig. 7 Tracks used to train the GRN. All of them are provided by TORCS

At this point of the evolution, two GRNs are remarkable:

- the best GRN of stage (e) is able to drive endlessly on this track, without the speed pressure. It is a safe driver that regulates its speed so that it can go through all the turns of this track.
- the best GRN of stage (f) is able to drive faster than the previous one but takes more risks. It optimizes the trajectories specifically to this track. In our opinion, this controller is overspecialized: whereas the first one can cover some other easy tracks, this one cannot.

Moreover, as presented on Fig. 9, the car is slightly shifted to the right side of the track. That might explains why the GRN cannot generalize its driving to other tracks: most of the turns on the training track are to the left. Thus, staying on the right side is better. However, on tracks with hard right turns, this position can be dangerous, the angle for right turns being closed. Moreover, some significant oscillations on the steering can be noticed. Even if they do not imply oscillations on the car track position, this behavior is unwanted and can be harmfull in a car race. The aim of the next evolution stages is to correct these defects.

3.3.2 Stage 2: generalization on three tracks

From the previous observation, we want a GRN able to safely cover all possible tracks, with all possible kinds of turns. With this aim in mind, we evolved the two previous GRNs a second time with the same evolutionary process but on three different tracks. The tracks used are CGSpeedway (in order not to lose the driving capacity of the previous GRN), Alpine and Street, whose layouts are presented in Fig. 7. The fitness function consists of summing the fitnesses of the first evolution stage successively applied to the three tracks.

The middle curve of Fig. 8 plots the evolution of the population's best, worst and average fitnesses. The restart mechanisms has also been applied: this explains the average fitness drops on the blue curve. Plateauing can be noticed during this evolution. It also corresponds to the successive difficulties of the tracks:

- the beginning hair pins of Alpine,
- the three turns at the top of Street,
- the very slow hair pin at the end of the long straight line of Street.

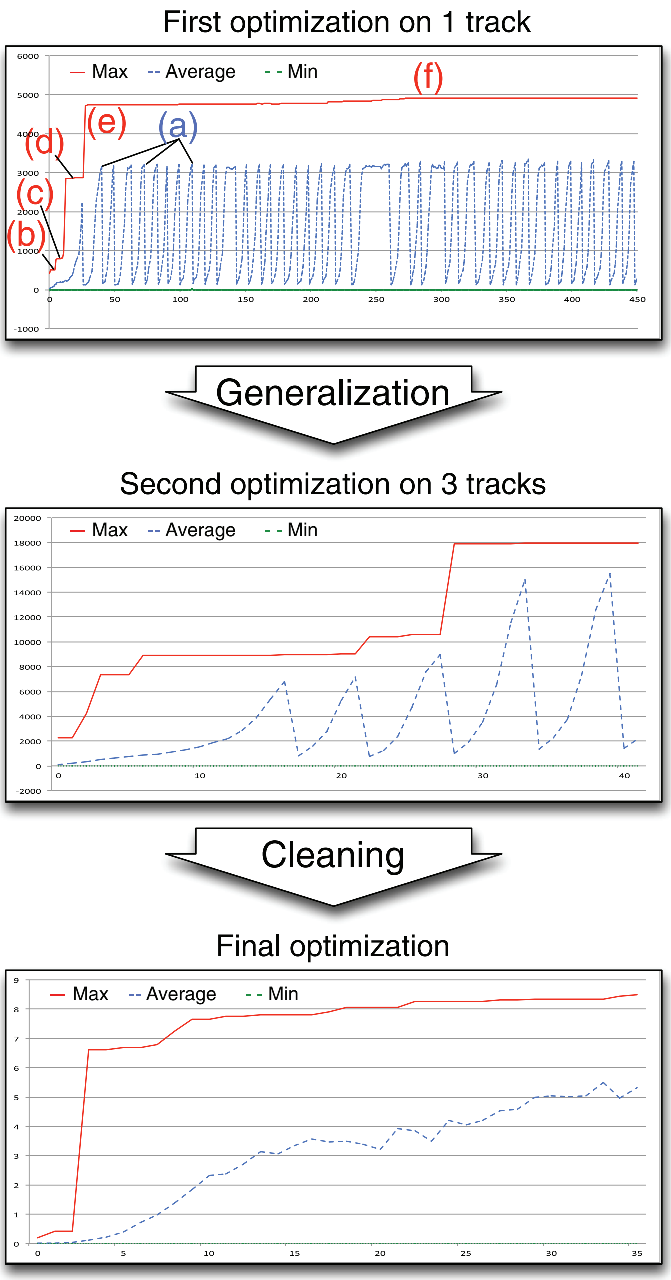

Fig. 8 Evolution of the fitnesses over the three evolution stages of the GRN. First, the GRN is evolved on one track to learn to drive. Then, the GRN is generalized on three different tracks. Then, the GRN behavior is cleaned up in order to reduce oscillatory issues

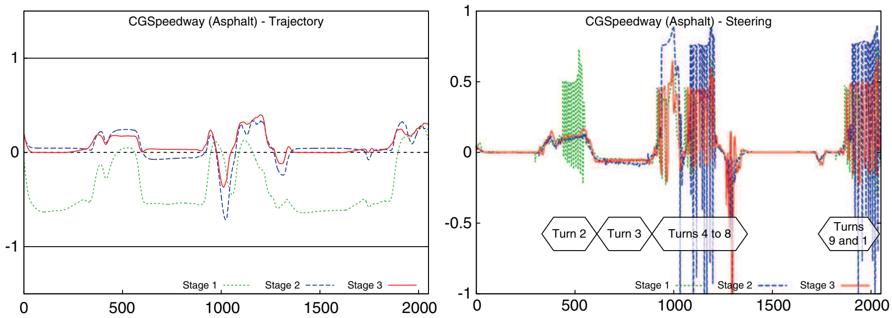

Fig. 9 Evolution of the car behavior during the three different stages of evolution. The *left-hand side* plot represents the track position of the car along the distance from the start line: 0 means the car is on the track centerline, -1 means the car is on the right edge of the track and 1 means the car is on the left one. The *right-hand side* plot is the steering output value along the distance from start. -1 means the steer is fully rotated to the right and 1 means fully rotated to the left (Color figure online)

At the end of this evolution, the best GRN is able to drive on every possible track. It drives very safely, going at a suitable speed to go through every kind of turn and braking when it detects a turn. However, the best GRN has an oscillatory behavior and is slightly shifted to the right hand side of the track. Whereas oscillatory behaviors are common in gene regulatory networks, both these issues could be harmful during a car race. This oscillation can be observed in Fig. 9 where the trajectories and the steering of the car are plotted during the second lap on the learning track (CGSpeedway). This oscillatory behavior can still be noticed on the steering plot in which the blue curve, which represents the second stage of evolution, strongly oscillates. The result is some parasitic behavior of the car on the trajectory, especially at the end of the turns. The final cleaning stage aims to reduce these parasitic behaviors.

3.3.3 Stage 3: cleaning the GRN's imperfections

To minimize the oscillatory behavior, we evolve the best GRN one last time. This time we add to the fitness function another test case that penalizes the continuous oscillations of the car on straight lines and long turns or fast multiple steering changes from full right to full left. As with the ticket system or the damage control used in the previous fitness functions, we simply stop the evaluation if we detect oscillatory behavior.

The detection routine proceeds as follows. A potential oscillatory behavior is detected when the steering wheel crosses its neutral position (i.e. if goes from left to right or from right to left). This initiates a countdown of 50 simulation steps. Within this 50 simulation steps, if the steering wheel crosses the neutral position more than three times and the sum of the steering variations is greater than a specified threshold (here empirically set up to 2.0, which corresponds to one steering switch from full right to full left), the oscillatory behavior is confirmed and the evaluation stops.

The green line on Fig. 9 shows the steering values of the best GRN on the CGSpeedway track at the end of this evolution stage. The steering spikiness of the previous evolution (blue curve) that is visible in the first two fast curves is smoothed and the steering does not oscillate anymore from full right to full left in the track section from turn 4 to turn 8 and from turn 9 to turn 1 (see Fig. 7).

It can be noted that this last evolution stage reinforces the generalization stage by improving the central position of the car. The GRN is also faster than before because the oscillations reduce the car speed in general. These multiple evolution stages were then strongly efficient to produce a GRN able to drive the car efficiently on most of the tracks. Table 2 shows the time performed by the best GRN on the learning tracks and on the 2012 SRC Competition tracks without further learning. The time represents a 10-laps race without opponents, fuel management, or damages. The GRN can also adapt to various kinds of track surfaces such as rock and sand. Here again, no re-optimization is necessary; the GRN naturally handles these new conditions. The next section shows how this GRN is able to naturally handle noisy sensors.

3.4 Noise resistance

All of the evolutions presented above have been performed without noisy sensors. The aim was to reduce the computational effort: noise implies multiple evaluations of the same individual in order to lower the effects of randomness. Moreover, we were expecting the GRN to be particularly resistant to noise. To verify this hypothesis, we have compared the time performed by the best GRN previously evolved during two 10-laps races on multiple tracks: one without noisy sensors and one with noisy sensors. According to SRC client and server manual [22], when noisy option is enabled, sensors are affected by independent and identically distributed normal noises with a standard deviation equal to 10 % of sensors range (track sensors) or to 2 % of sensors range (opponents sensors). We never use the focus sensors that are only affected by a 1 % standard deviation. When the GRN is used in a noisy environment, no filter is used between the noisy sensors and the GRN inputs: the noisy values are directly provided as non-noisy ones.

Table 2 compares the results obtained without noisy sensors and with noisy sensors.[3] In a 10-lap race, the time loss due to the noise management is not substantial. In some cases, on Kerameikos-mountain for example, the noise is even beneficial to the GRN: this track, a slippery stony road with harsh hairpin turns, is particularly difficult. The noise helps the GRN by creating micro oscillations that allow the GRN to escape from difficult situations. More generally, Fig. 10 presents the trajectories of the GRN without (gray dashed line) and with (red plain line) noise. The trajectories are represented by the distance to the track centerline: 0 means the car is on the centerline, -1 means the car is on the right edge of the track and $+1$ means the car is on the left edge of the track. The trajectories of the driver without and with noise are very similar on the four tracks tested. Some minor micro

[3] A video of the capacity of the GRN to handle the noise is available on-line: http://www.irit.fr/~Sylvain.Cussat-Blanc/GRNDriver/index_en.php.

Table 2 Time of the GRNDriver on various tracks with and without noisy sensors (elapsed time of a 10-lap race without opponents, fuel management, or damages)

Track name	Track type	Time without noisy sensors	Time with noisy sensors	Difference
Alpine	Asphalt	28:54.98	29:02.55	+00:07.57 (+0.4 %)
CGSpeedway	Asphalt	07:34.09	07:35.23	+00:01.14 (+0.2 %)
Street	Asphalt	16:00.25	16:21.68	+00:21.43 (+2.2 %)
Emero-city	Asphalt	12:31.35	12:37.35	+00:06.00 (+0.8 %)
Illschwang-desert	Sand	15:40.51	14:44.48	−00:56.03 (−6.0 %)
Kerameikos-mountain	Rocks	20:36.40	18:43.22	−01:53.18 (−9.1 %)
Kerang-desert	Sand	14:03.42	14:08.65	+00:05.23 (+0.6 %)
Mikegrady-hill	Asphalt	14:53.56	14:59.47	+00:05.91 (+0.7 %)
Mueda-city	Asphalt	12:56.05	13:04.54	+00:08.49 (+1.1 %)
Noceda-city	Asphalt	12:02.79	12:05.80	+00:03.01 (+0.4 %)
Senhor-hill	Asphalt	17:46.20	18:32.03	+00:45.83 (+4.3 %)
Zvolenovice-mountain	Rocks	13:14.08	13:20.66	+00:06.58 (+0.8 %)
Average				−00:04.84 (−0.3 %)

Time format: mm:ss.ms

oscillations appear with noisy sensors but they are not sufficient to destabilize the car. Some larger oscillations appear in a particular section: on Kerang-desert, at the position 2,750, the car oscillates more than usual but the GRN is able to stabilize quickly after three periods of oscillations. The same phenomenon, less pronounced, appears on Noceda-city at position 1,750 and on Mikegrady-hill at position 2,250. The same recovery behavior can be noticed: the GRN stabilizes the car once again in two oscillation periods. These results are very satisfactory, keeping in mind that the GRN is used without a filter on the inputs.

We have compared the effect of noise on our driver and on six other approaches. These are Mr Racer's CMA-ES based approach [29, 30], Autopia's fuzzy controller [26, 27], Cobostar CMA-ES optimized hand-coded strategies [5], Cardamone's NEAT driver [6–8], Ready2Win's modular architecture [4] and Mariscal's expert system [15]. All these drivers have competed either in the 2013 competition or in older editions. They are six successful approaches used in the SRC competition (winner to third position). These drivers have been downloaded from the competition website. Table 3 presents the gain percentage (a positive percentage means the driver drives slower with noise than without and vice-versa) of the drivers on the different tracks. The results are obtained by running each driver on the tracks for a 10-laps race with damages, without opponents and fuel management. At the end of the 10-laps with noise and then without noise, the percentages are computed with the global elapsed time of the races. It has to be notice that the GRNDriver is directly connected to the simulator inputs without any filter nor the use of the focus sensors (that reduce the noise in one chosen direction). Actually, few drivers use a noise reduction system: only Mr Racer uses a quadratic regression to handle the noise [28] and Ready2Win use a simple noise remover method based on averaging

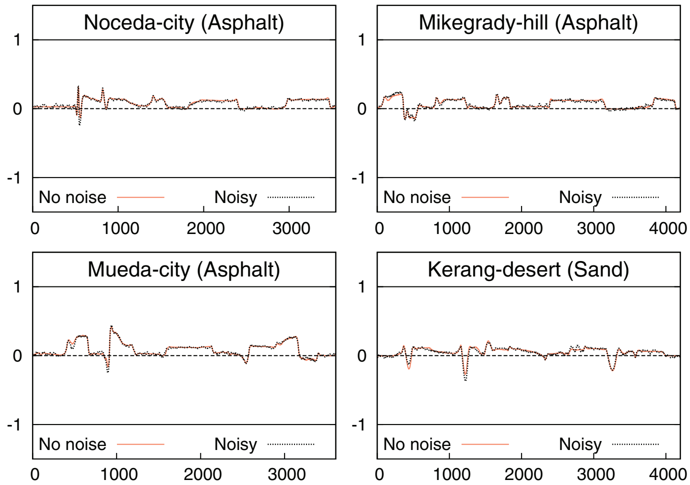

Fig. 10 Trajectories of the GRNDriver on four different tracks, during one lap, without (*gray dashed line*) and with (*red plain line*) noise. Abscissa represents the distance from the start line and ordinate represents the position of the car on the track (Color figure online)

Table 3 Comparison of the resistance to noise of the GRNDriver and 6 other approaches

	GRN driver (%)	Mr Racer (%)	Autopia (%)	Cobostar (%)	Cardamone (%)	Ready2Win (%)	Mariscal (%)
Alpine	+0.4	DNF	+0.4	+3.5	+0.9	DNF	−13.2
CGSpeedway	+0.2	+0.2	+0.7	+4.0	+1.6	−0.9	+0.5
Street	+2.2	−0.3	+1.6	+2.6	+3.6	+7.1	+11.6
Emero-city	+0.8	+0.2	−0.4	+2.8	+0.5	−0.3	+2.8
Illschwang-desert	−6.0	−0.1	0.0	+12.2	+1.5	−0.8	DNF
Kerameikos-mountain	−9.1	DNF	+0.1	+8.4	+1.4	DNF	+0.8
Kerang-desert	+0.6	−7.5	+0.4	−8.1	+3.4	−4.4	DNF
Mikegrady-hill	+0.7	+0.1	+1.6	+4.9	+1.5	−0.9	+4.8
Mueda-city	+1.1	−4.9	+0.6	+3.8	+2.1	+1.4	+4.6
Noceda-city	+0.4	−0.5	+1.5	+9.6	+1.7	DNF	+3.9
Senhor-hill	+4.3	+2.1	−0.2	−9.8	+1.3	DNF	+0.6
Zvolenovice-mountain	+0.8	−0.2	−0.1	+6.0	+1.0	+3.1	−12.3
Average	−0.3	−1.1	+0.5	+3.3	+1.7	+0.5	+0.4

Each value of the table is a percentage that represents how much time the driver is taking with noisy sensors to finish a 10-laps race with damages but without opponents, fuel management than without noisy sensors. DNF means that the driver did not finish the race due to damages. Averages are thus computed without DNF races values that are 5 % higher or lower than the 5 past averaged values. As the GRNDriver, all other drivers have a direct connection of the inputs to the control systems. Comparing the average values, the GRNDriver and Mr Racer are the only two drivers to gain time with the noise. Whereas Mr Racer's quadratic noise reduction system provides it a strong advandage, the GRNDriver is not really affected by the noise and even gain few seconds avoiding crashes as stated previously. In comparison, other approaches loose few seconds handling noise. Three other approaches handle the noise quite well: Autopia, Ready2Win and Mariscal are almost unsensitive to the noise with an average gain percentage lower than 0.5 %. In conclusion, GRNDriver handles the noise well in comparison to other approaches without noise resistance systems.

4 Optimizing the GRN for racing

4.1 Learning to drive fast

The gene regulatory network optimized with the previous method is a safe driver, able to finish almost any kind of tracks, with or without noisy sensors. However, this GRN is not fast enough to compete with opponents. In order to make it drive the car faster, we have distorted the GRN longitudinal speed. The idea is to trick the GRN about its speed to make it accelerate and brake in particular areas of the track. To do so, the speed sensor value is multiplied by a coefficient. The bigger the coefficient is, the faster the GRN thinks the car is going and the most it tries to slow down by braking. Actually, this proves the GRN perfectly correlates its speed and the dangerousness of the current car state (track sensors, lateral speed, etc.). This coefficient is calculated according to a target speed learned during the warm-up stage of the race.[4] The GRN speed perception is distorted in order to reach the target speed as follows:

- If the car speed is 10 km/h under the target speed, the GRN's speed sensor is set to zero in order to make it accelerate as much as it can handle it.
- If the car speed is 10km/h over the target speed, the speed sensor provided to the GRN is multiplied by $1 + d_s/50$ where d_s is the difference between the current speed and the target speed. The GRN is then pushed to reduce its speed but not too drastically. For example, if the car is in a turn, it would be counter-productive to brake (the car will spin).

To learn the target speeds, we use a scripted approach. This approach is comparable to Butz et al. method [5] in which crash points are detected during the race and the car is slowed down in these areas in order to secure the car behavior. In our approach, this optimization is made during the warm-up stage. To do so, the

[4] The warm-up stage consists of 100,000 timesteps that can be used by the competitors in order to collect data about an unknown track.

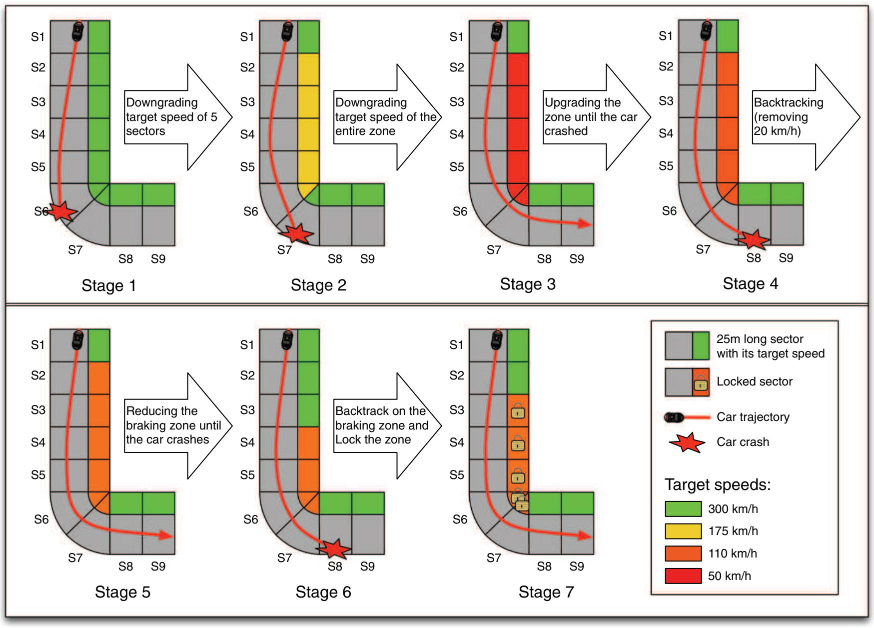

Fig. 11 Example of target speed optimization on one turn: the target speed is first initialized to maximum speed (300 km/h). If the GRN cannot go through a turn, the speed is gradually reduced by 125 km/h until reaching 50 km/h (Stages 1 and 3). When the car can manage the turn, the target speed is gradually increased by 20 km/h until the car crashes again (Stage 3 and 4, here only represented on one picture). Then, the braking zone is reduced as much as possible (Stages 5 and 6). When the car crashes again, the braking zone is backtracked to the previous size and all sectors are locked (Stage 7)

track is first divided into 25 m long sectors. The target speeds are all initialized to 300 km/h in order to push the GRN to drive as fast as possible. When the GRNDriver spins or leaves the track, the sectors of a zone which contain the current sector and the 5 sectors upstream are marked as "reducing". When marked, the target speeds of these sectors are reduced by 125 km/h. When the target speeds of this zone reach a minimal value of 50 km/h, one sector is added to the zone and its target speed is decreased. With this method, we can guarantee that the GRN can handle every kind of turns, even when it approaches very fast and has to brake quickly. Once the zone is passed, all sectors of the zone are marked as "increasing" and the target speeds of all sectors are gradually increased by 20 km/h until the GRNDriver crashes again in this zone. When this happens, the previously added 20 km/h is subtracted and the sectors are marked as "braking". The final step consists of reducing the possibly too long braking zone. To do so, the target speed of the zone's first sector is set to 300 km/h: the braking zone will be reduced each time the GRNDriver is able to go through the modified zone. When the GRNDriver crashes once again in this zone, the previous target speed is restored and the zone is marked as "locked". When locked, the GRNDriver can still crash because of the noise. If this happens, the target speeds are reduced by 5 km/h to secure the zone. Figure 11 presents an example of the optimization mechanism.

Table 4 Comparison of the best lap out of three with and without target speed optimization

Track name	Track type	Time without target speeds	Time with target speed	Difference
Emero-city	Asphalt	12:37.35	10:51.85	−01:45.50 (−13.9 %)
Illschwang-desert	Sand	14:44.48	11:08.05	−03:36.43 (−24.5 %)
Kerameikos-mountain	Rocks	18:43.22	14:43.77	−03:59.45 (−21.3 %)
Kerang-desert	Sand	14:08.65	12:32.10	−01:34.55 (−11.1 %)
Mikegrady-hill	Asphalt	14:59.47	13:00.87	−01:58.60 (−13.2 %)
Mueda-city	Asphalt	13:04.54	10:46.48	−02:18.06 (−17.6 %)
Noceda-city	Asphalt	12:05.80	09:35.98	−02:29.82 (−20.6 %)
Senhor-hill	Asphalt	18:32.18	13:26.18	−05:05.85 (−27.5 %)
Average				−02:51.03 (− 18.7 %)

Format of the lap time: mm:ss.ms

The process can be pipelined along multiple runs: if the car crashes on the third zone, it means that the third zone must be modified but it also implies that the GRN was able to handle the first two zones. Thus, they can be optimized by potentially going to the next step. The marking process is linear per zone (a mark of a zone can only be increased, never downgraded to a previous stage) expect for the first zone which can be marked as "reducing" when the car crashes in this zone after it finishes a lap. This ensures that the first turn is perfectly covered, even if it is after a long straight line.

Table 4 presents the time performed by the best GRN on various tracks taken from the 2012 competition with and without this speed optimization. To do so, the GRN is tested without target speed and with target speed on all tracks with noisy sensors. For the target speed optimization, we use the standard warm-up procedure that consists of running the optimization for 100,000 simulation steps; then the GRN with the optimized target speed vector is run in race mode without opponents for 10 laps. Damages and fuel are disabled but the GRN has to handle the noise, as in the competition. By learning the target speed for the different sectors of the track, the GRN runs on average 2 min 51 s or 18.7 % faster than the default GRN. The gain is significant in all tested tracks.

Figure 12 presents the track position and the longitudinal speed of the car on 4 different tracks, with and without target speed optimization. The speed gain of this approach is undeniable in all the sectors of the tracks. The GRN is accelerating earlier and stronger and brakes later. Moreover, it may be noted that the increase of the speed drives the GRN to use the full width of the track, the speed dragging the car to the outside edge of the track.

4.2 Avoiding opponents

In order to compete efficiently, the driver must avoid or overtake opponents. The GRN controller has learned to drive without such considerations and, even if it can be fast using the "target speeds" method, it needs to be able to change trajectories in order to avoid or to overtake opponents. The GRN controller we chose to compete

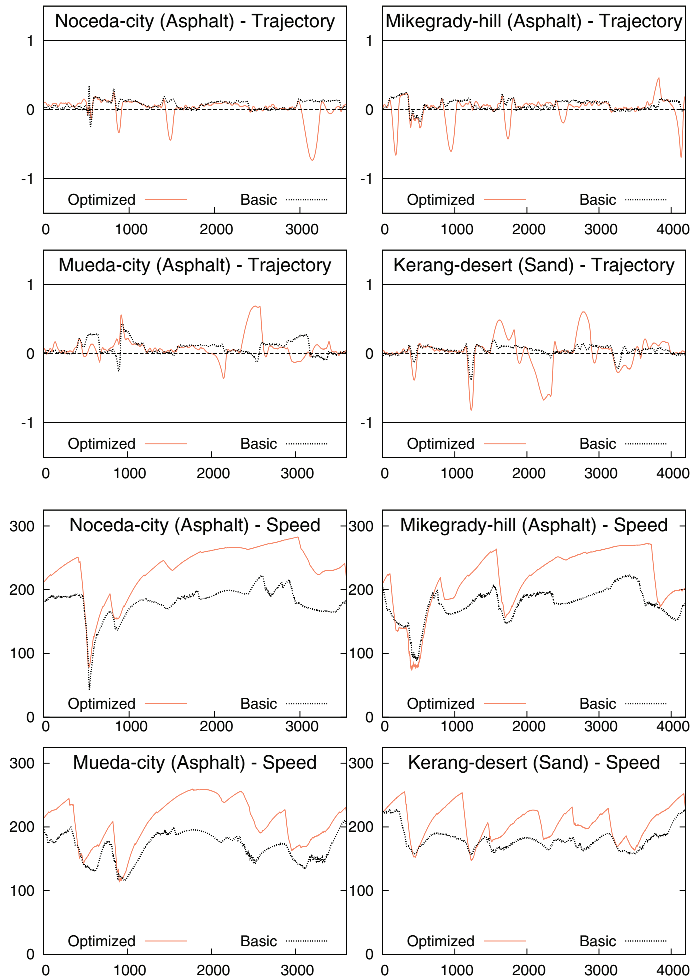

Fig. 12 Trajectories (*top*) and longitudinal speed (*bottom*) of the car on 4 different tracks with and without target speed optimization

presents an interesting and usable feature: two track sensors, one on each side of the car, are linked to the steering actuators in order to keep the car on the centerline of the track. This characteristic is common to almost every GRN that was evolved through the process previously described, although the track sensors used to center the car on the track might differ and are not necessarily symmetric. In order to modify the car trajectory toward the left or the right hand side of the track, we can alter both sensors as we altered the longitudinal speed sensor to make the GRN drive faster in the previous section. For example, we can trick the GRNDriver into thinking that the track is much larger on the left by making the input from left track sensor greater, but keeping the one from the right sensor unchanged. The GRNDriver will then believe that it is driving on the right side of the track instead of in the middle of the track and it will turn left to reach the new imaginary centerline. Moreover, the others sensors being untouched, the controller still automatically compensates steering and throttle output levels to keep the car inside the track limits.

The two chosen sensors for this purpose are presented by bold arrows in Fig. 13. By multiplying the left sensor value by 1.65 (empirically chosen through several tests), the car shifts its trajectory by the width of a car to the left. Multiplying the right sensor value by 2.65 (empirically chosen) shifts the car trajectory by the width of a car to the right. Figure 14 presents the trajectory of the car on Noceda without modification to the track sensors (plain red line), the one produced by applying the 1.65 factor to the left sensor (dashed blue line) and the one produced by applying the 2.65 factor to the right sensor (dashed green line). The car shifts its trajectory as expected but is still able to drive through all the track. The GRN adapts its behavior in order to keep the car on track even with the modified sensor inputs.

Based on this observation, this method is used to modify the car trajectory according to the opponents detected in the car neighborhood. If an opponent is detected within a 25 m range, the car deviates in the opposite direction to the one the opponent is detected in. This procedure only applies if the front track sensor value is >50 m. If it's not, overtaking is detected as unsafe and the car will stay behind the opponent car with a procedure described hereafter. If overtaking occurs, the opponent car is tracked during the whole operation and the track sensors return to their real value when the front and side opponent sensors do not detect any opponent nearby. Figure 15 shows this procedure and Fig. 16 shows the trajectory taken by the car in a real situation extracted from TORCS.

If opponents are detected on both sides of the car or if the distance ahead is not sufficient, the GRNDriver will stay behind the closest opponent ahead (Fig. 17). To keep the GRNDriver behind an opponent, we adjust the speed of the car using the same method we used in Sect. 4.1. In this case, the longitudinal speed sensor is adjusted according to the speed of the closest opponent as follows:

$$speedX = \begin{cases} 5 * speedX & \text{if } d < 5 \\ speedX * max\left(tr, 4 - \frac{4d}{25}\right) & \text{if } 5 \le d < 25 \\ speedX * tr & \text{if } d \ge 25 \end{cases} \tag{7}$$

where *speedX* is the value of the longitudinal speed, *tr* is the value of speed factor in the current sector (see Sect. 4.1), d is the distance between the car and the closest opponent in front of the car. The factor applied to the longitudinal speed when the opponent is closer than 5 m forces emergency braking to avoid a collision. It might be that the GRN is stuck behind an opponent while overtaking another one as represented in Fig. 17b. In this case, the three concerned inputs (left track sensor, right track sensor and longitudinal speed) are simultaneously modified according to the overtaking and staying behind rules.

4.3 Recovering from a crash

During a race with opponents, the car can go off the track for different reasons (collision, braking or steering errors due to the noise, etc.). In this case, a track

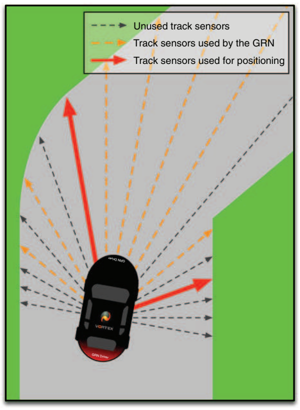

Fig. 13 Two track sensors (*in plain red*) can be modified in order to change the car trajectories. For example, increasing the left sensors value will automatically modify the GRN behavior, tricking it about its position on the track. Thus, it will change its trajectory while maintaining a global coherent behavior (stay on the track) (Color figure online)

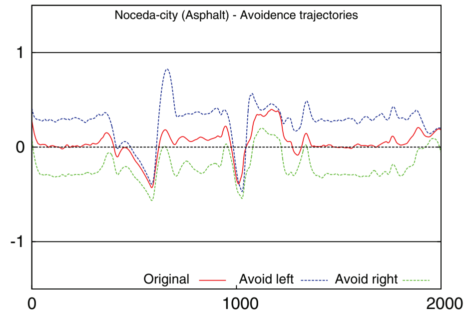

Fig. 14 The *plain red line* represents the normal racing line computed by the GRNDriver on Noceda Track. The *dashed blue line* is the altered racing line using a fixed 1.65 factor on the left track sensor. The *dashed green line* is the altered racing line using a fixed 2.65 factor on the right track sensor (Color figure online)

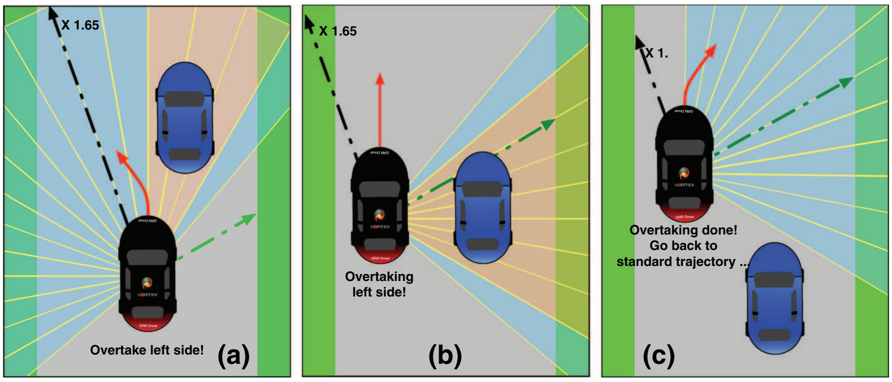

Fig. 15 Three phases in avoidance routine: **a** The GRNDriver detects a car on the right and begins to overtake on the *left side*. **b** The GRNDriver overtakes, keeping the car on the left of the track. **c** Overtaking is done, the GRNDriver resumes its normal racing behavior. The *blue* and *red* angular sectors represent the opponent sensors as described in the SRC competition client. Only the useful sensors are represented for clarity matters (Color figure online)

recovery routine is applied to get the car back on track. The track sensors provided by TORCS when the car is off the track are all equal to -1. Because of that, the GRN cannot learn to go back on the track. Instead, we have implemented a simple

Fig. 16 The GRNDriver avoids the red car. The plain line is the normal trajectory, the *dotted line* is the altered trajectory using a 2.65 factor on the right track sensor used to position the car in the middle of the track (Color figure online)

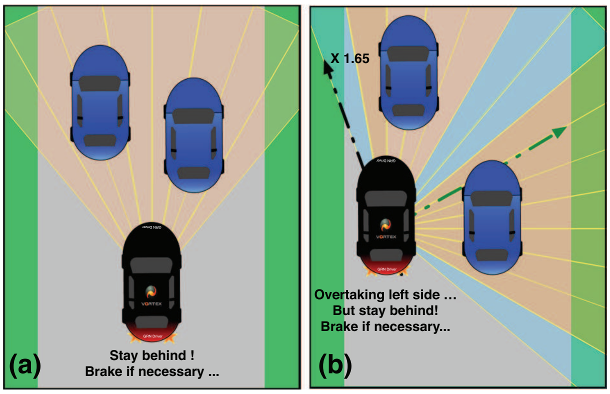

Fig. 17 **a** The GRN detects opponents slower than it but no sides are free to overtake, it then slows to stay behind. **b** The GRN is overtaking the blue car on the right, but another slower opponent blocks the line, it then slows to stay behind the car on the left, keeping its altered trajectory to still overtaking the car on the right (Color figure online)

script that turns the car in the direction of the middle of the track and drives forward. Once the track sensor values are coherent again, the GRN takes control of the car back and resumes race.

5 Comparative study

To evaluate the model presented in this paper, we have compared the GRNDriver with other approaches that have competed during the past Simulated Car Racing competitions. As in Sect. 3.4, the selected drivers for this comparative study are Mr Racer, Autopia, Cobostar, Cardamone, Ready2Win and Mariscal. The comparative study is based upon:

- the best lap times in qualification mode,
- the elapsed times on a 10-laps race with noise, damages and without fuel management nor opponents,
- the final positions on 10-laps races with all the opponents, with noise, damages and without fuel management.

All these comparison have been made after independent warm-up stages for each driver on each track. These results might differ from the results obtained during the competition due to minor bug resolution in our code for this paper. The three next sections present these comparisons.

5.1 Qualifications: best lap comparison

In this first comparison, the drivers are alone on a track and are running for 10-laps. Fifteen tracks have been selected to have various types of coatings (asphalt, rock and sand) and different profiles (city, mountain, etc.). Table 5 presents the best lap of each driver at the end of these 10 laps.

In this table, the GRNDriver is compared before and after the target speed optimization presented in Sect. 4.1. Without target speeds optimization, the GRNDriver competes with the slowest drivers with an average final position of 6.40 over 8 participants. The main observation that can be made is that the GRNDriver can drive on any kind of tracks, without prior learning on these tracks. However, it has a safe driving behavior that does not allow it to compete with the best approaches.

With target speed optimization, the GRNDriver is still not the fastest driver but it is usually well ranked: it finishes amongst the first three fastest drivers 13 times out of 15. The GRNDriver competes particularly well on slippery tracks such as mountain (rocks) and desert (sand) tracks. We can note that the GRNDriver has only been trained on asphalt tracks (CGSpeedway, Alpine and Street) and never on slippery tracks. Moreover, there is no specific parameters or sub-routines to handle slippery tracks. The GRN controller is used as is. This emphasizes the capacity of the GRN to adapt its behavior under unknown conditions.

5.2 10-laps races without opponents

In this second comparison, the drivers are running for a 10-laps race with noise and damages and without fuel management or opponents. Prior to each race, a warm-up has been run so that each driver starts on a fresh learning basis. Table 6 presents the results of each driver on 15 different tracks.

The GRNDriver without target speed is evaluated first. Whereas some drivers cannot reach the finish line of the race (see DNF signs in the table) even after a warm-up session, the GRNDriver without target speed optimization, and thus without any *a-priori* knowledge of the tracks, finishes all tracks. Moreover, GRNDriver is faster than one of the opponent (Cardamone): its average final position is 5.8 in comparison to 6.87 for Cardamone.

With the target speed optimization, the GRNDriver still finishes all the races and it is very competitive: its average final position is 1.80 and it finishes 14 races out of 15 in the top three pilots. Whereas other approaches defeats the GRNDriver on a one-lap race, the GRNDriver is more competitive on longer runs. This shows the capacity of the GRN to keep a stable behavior on long noisy runs. Once again, we can notice that the GRNDriver beats all other drivers when the track conditions

Table 5 Comparison of the best laps (in seconds)

	GRND. w/o TS	GRND. w. TS	Mr Racer	Autopia	Cobo star	Carda-mone	Ready2Win	Maris-cal
Alpine 1	173.49	156.67	147.93○	**142.3** ★	199.37	182.02	147.34●	153.69
CGSpeedway	44.97	41.02○	49.39	**40.54** ★	40.99●	51.18	42.94	43.03
Street 1	94.92	91.46	86.85○	86.3●	95.49	106.97	**84.72** ★	87.05
Emero-city	75.17	66.87	**64.11** ★	66.53○	65.47●	85.92	66.67	68.93
Mueda-city	89.55	64.59●	**64.55** ★	64.78○	65.54	86.77	68.76	70.38
Noceda-city	71.73	57.35●	**56.95** ★	59.77○	68.64	80.6	85.3	62.7
Sancassa-city	76.69	67.68●	69.02○	**65.85** ★	89.58	86.84	119.72	71.18
Alsoujlak-hill	90.45	75.13○	**73.5** ★	74.81●	92.79	95.26	78.84	78.65
Mikegrady-hill	104.99	74.36●	**72.66** ★	77.04	76.51○	97.82	77.31	80.55
Senhor-hill	91.21	**79.16** ★	79.19●	79.23○	80.46	100.65	256.71	83.6
Keiramekos-mountain	102.38	**85.42** ★	90.38●	90.51○	101.78	111.36	93.1	99.69
Zlovenovice-mountain	89.93	76.18●	86.11	**75.85** ★	85.39	92.59	80.52○	124.57
Arraias-desert	78.06	67.69●	**65.07** ★	68.48○	73.4	78.26	68.68	142.22
Illschwang-desert	88.23	**64.72** ★	76.46	68.38●	76.05	81.73	72.38○	98.92
Kerang-desert	83.93	**75.37** ★	82.44	77.32●	84.89	95.19	77.85○	102.48
Average position	6.40	2.47	2.67	2.27	4.93	7.20	4.47	5.67

Bold-stared values are best over all approaches, bulleted ones are seconds and circled ones are thirds. GRNDriver is tested with and without target speeds optimization (TS) become slippery (mountains and desert). That shows the capacity of the GRN to adapt to the changing track conditions without further learning.

5.3 10-laps races with opponents

In this last comparison, all the drivers compete against each other in 10-laps races on three different tracks. Each race is run nine times: 3 times with the same initial starting grid based on the 10-laps races results presented in Table 6 and 6 times with different starting positions (based on a circular rotation of all the drivers). For computational reasons, the three tracks of the 2013 SRC competition have been selected: one on asphalt (Sancassa-city), one on sand (Arraias-desert) and one in mountain (Alsoujlak-hill). Only the GRNDriver with target speed optimization and with the opponent management is evaluated in this section. Table 7 shows the starting and final position of the drivers for all the runs on these three tracks. Runs $a - c$ are runs starting with the best 10-laps solo races positions and runs $d - i$ are the one with circular starting positions.

Globally, the GRNDriver is very competitive with an average finishing position of 1.67. In comparison, the second best driver, Autopia, finishes at an average finish position of 1.89. This shows the capacity of the GRN to use modified inputs to handle opponents. Even if the GRNDriver is starting on the back of the grid, it is able to gain positions, because it is fast on long run races (see Table 6) and because the modification of the inputs is well managed by the GRN. In comparison to modifying the outputs, modifying the inputs allows the GRN to keep its regulatory ability. Thus, the GRN adapts its outputs to specific situations such as overtaking an opponent, but its driving behavior remains globally the same and the GRNDriver almost never goes out of track. For example, it can slow down if the car state becomes dangerous while overtaking an opponent in a turn to make its behavior more conservative to avoid a collision or going out of track. Locally, we can notice that in most cases (the only counter example being run d on Alsoujlak-hill), the GRNDriver always gains positions and finishes all the races. That shows its capacity to avoid opponents and dangerous situations in order to keep its damage level low and thus finish the race.

6 Discussion and analysis of the GRN

After the comparison with all the other approaches, this section discusses the GRN used in this work. Since it is obtained through evolution, we discuss the global regulation flows to explain the global functioning of the GRN. In a second part, we also discuss the advantages and weaknesses of this approach.

6.1 Analysis of the GRN

Figure 18 represents proteins and enhancement and inhibition bends of the GRN controller that competed in 2013 SRC competion.

Table 6 Comparison of the elapsed times (mm:ss) of 10-laps races

	GRND. w/o TS	GRND. w. TS	Mr Racer	Autopia	Cobo star	Carda-mone	Ready2Win	Maris-cal
Alpine 1	29:00	26:19●	DNF	**23:54** ★	34:23	30:26	DNF	27:06○
CGspeedway	07:35	06:55●	08:18	**06:51** ★	07:21	08:39	07:16	07:12○
Street 1	16:07	15:22○	**14:36** ★	14:52●	16:11	18:23	15:38	16:01
Emero-city	12:37	11:17	**11:01** ★	11:13●	11:26	14:34	11:16○	11:38
Mueda-city	15:00	**10:52** ★	11:13○	10:55●	11:23	14:34	11:49	11:52
Noceda-city	12:05	09:39●	**09:38** ★	10:05○	11:38	13:45	DNF	10:35
Sancassa-city	12:53	11:21●	11:46○	**11:08** ★	15:05	14:43	DNF	12:04
Alsoujlak-hill	15:10	12:37○	**12:32** ★	12:36●	15:44	15:59	13:23	13:20
Mikegrady-hill	17:35	12:31●	**12:11** ★	12:58○	13:08	16:25	12:59	13:49
Senhor-hill	16:00	**13:18** ★	13:33○	13:23●	14:10	16:54	DNF	14:09
Keiramekos-mountain	17:38	**14:28** ★	DNF	15:23●	17:09○	19:13	DNF	21:45
Zlovenovice-mountain	15:08	**12:52** ★	15:13	12:54●	14:21	15:34	13:56○	34:33
Arraias-desert	13:09	**11:26** ★	13:36	11:56○	12:33	14:10	11:49●	DNF
Illschwang-desert	14:49	**10:57** ★	12:57	11:36●	13:21	13:45	12:16○	18:17
Kerang-desert	14:08	**12:41** ★	16:09	13:17○	14:50	16:14	13:17●	DNF
Average position	5.80	1.80	2.07	3.80	5.13	6.87	5.00	5.53

Bold-stared values are best over all approaches, bulleted ones are seconds and circled ones are thirds. DNF means that the driver did not finish the 10 laps due to damages

The first observation is that all the evolved networks, as well as the one presented in this paper, contain few proteins. As a matter of fact, most of the GRN that are able to drive a car in TORCS present five to fifteen regulatory proteins. However, analyzing how the GRN works can be complex: a protein enhances and inhibits the linked proteins accordingly to the sum of enhancements and inhibitions it receives from other proteins in the GRN (see Sect. 2). The stronger an enhancement is (or the stronger an input is), the more it enhances and inhibits the linked proteins. A protein that is totally inhibited or that is not enhanced (an input with no signal) does not enhance nor inhibit the linked proteins. Moreover, according to the equations that compute enhancements and inhibitions in Sect. 2, and considering the control parameters $\beta = 1.07965$ and

Table 7 Simulated races between the opponents

		Sancassa-city										Arraias-desert										Alsoujlak-hill										Avg
		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	avg	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	avg	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	avg	
GRNDriver with TS	Grid	2	2	2	1	7	6	5	4	3		1	1	1	7	6	5	4	3	2		3	3	3	2	1	7	6	5	4		
	Finish	1	2	2	1	2	2	2	1	1	1.56	1	1	1	1	1	3	1	1	1	1.22	3	2	2	3	2	2	2	1	3	2.2	1.67
Autopia	Grid	1	1	1	7	6	5	4	3	2		3	3	3	2	1	7	6	5	4		2	2	2	1	7	6	5	4	3		
	Finish	2	1	1	2	1	1	1	2	2	1.44	2	4	2	3	3	2	4	2	2	2.67	2	3	1	1	1	1	1	2	2	1.56	1.89
Mr Racer	Grid	3	3	3	2	1	7	6	5	4		5	5	5	4	3	2	1	7	6		1	1	1	7	6	5	4	3	2		
		4	4	4	6	3	4	5	5	4	4.33	5	**6**	**6**	**6**	**7**	6	**6**	**6**	**4**	5.78	1	1	3	2	3	4	3	3	1	2.33	4.15
Cobostar	Grid	6	6	6	5	4	3	2	1	7		4	4	4	3	2	1	7	6	5		6	6	6	5	4	3	2	1	7		
	Finish	6	5	6	5	5	5	4	4	5	5	3	3	3	2	2	1	2	3	**6**	2.78	**6**	6	6	6	6	6	6	5	6	5.89	4.56
Cardamone	Grid	5	5	5	4	3	2	1	7	6		6	6	6	5	4	3	2	1	7		7	7	7	6	5	4	3	2	1		
	Finish	5	6	5	4	6	6	6	6	**7**	5.67	6	5	5	4	4	4	3	4	3	4.22	5	7	7	7	5	7	5	6	7	6.22	5.37
Ready2win	Grid	7	7	7	6	5	4	3	2	1		2	2	2	1	7	6	5	4	3		5	5	5	4	3	2	1	7	6		
	Finish	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**6**	6.89	4	2	4	5	5	5	5	5	**5**	4.44	4	5	4	4	4	3	4	**7**	4	4.33	5.22
Mariscal	Grid	4	4	4	3	2	1	7	6	5		7	7	7	6	5	4	3	2	1		4	4	4	3	2	1	7	6	5		
	Finish	3	3	3	3	4	3	3	3	3	3.11	**7**	**7**	**7**	**7**	**6**	**7**	**7**	**7**	**7**	6.89	**7**	4	5	5	**7**	5	7	4	5	5.44	5.15

Races tagged $a-c$ are races with a starting position from best lap comparison (see Table 5) and races $d-i$ are races with circular starting grids, equivalent to the one used in the competition. Grayed cells are unfinished races due to damages. Averages are the averaged final position over the 9 runs and the last column represents the global averaged final positions

$\delta = 0.712952$ of this particular network, a protein in the presented GRN can significantly enhance or inhibit neighbor proteins until ± 3 around its tag value (some of the evolved GRN present a significant influential range of ± 5 around the protein tag value with a greater β value). Thus, the dynamics of enhancement and inhibition flows can become extremely complex inside a GRN. At the time of this paper, we are still investigating how to represent and to analyze accurately how a GRN works. This study will be the subject of future scientific publications. Nevertheless, the Fig. 18 gives an insight of how the GRN presumably works.

6.1.1 General structure

The presented GRN shows several interesting structural features. Firstly, the GRN can duplicate a protein in order to amplify (by addition) its enhancing and inhibiting strengths. It is the case in this GRN of proteins $R6$ and $R7$ that are both duplicated (represented on Fig. 18 by the black background). Secondly, this network contains regulatory proteins with the same tag values but different proteins to enhance and to inhibit. The aim is to extend their enhancing and/or inhibiting influential ranges. In the presented GRN, the regulatory proteins $R6$ and $R7$ combine their actions, having the same tag value (15) but different proteins to enhance (tags 17 and 22), extending the enhancement range from 14 to 25. Thirdly, some proteins such as $R8$ or $R11$ enhance themselves and amplify enhancement and inhibition of linked proteins. In contrast, some proteins such as $R7$ inhibit themselves when enhanced. This gives them a special role: a protein needs two steps to inhibit itself because the effect on the concentration is only visible at the next regulatory step. In other words, this protein regulates on one step before inhibiting itself on the next step. That produces an oscillatory behavior. Finally, the protein $R9$ is only regulated but does not regulate any other proteins within a ± 3 identifier range. We can consider this protein as a evolutionary side effect since this protein does not participate to the regulatory process.

6.1.2 Steering regulation

An interesting spatialization of the network can also be observed with half the regulatory proteins mostly regulating left steering output protein O_L and the other half mostly regulating right steering output protein O_R. Proteins represented with a gray background activate principally left steering. White ones activate principally right steering. Gray proteins are mostly enhanced by left input proteins (named from I_{L1} to I_{L4}) and enhance output protein O_L. These proteins also inhibit white regulatory proteins that activate right steer. Symmetrically, right input proteins (named from I_{R1} to I_{R4}) enhance mostly regulatory proteins $R5$, $R6$, $R7$ and $R8$ that directly enhance right steer output protein O_R, or that enhance protein $R10$ that enhances output protein O_R. They also directly inhibit gray regulatory proteins that activate left steer or enhance white regulatory protein $R11$ that inhibit gray ones. This means that if left track sensors indicate a farther distance than right side sensors, the GRN enhances steering to the left and inhibits the regulatory proteins that enhance steering to the right. If right track sensors indicate a farther distance than left track sensors, the opposite effect occurs. If sensors from both sides sense

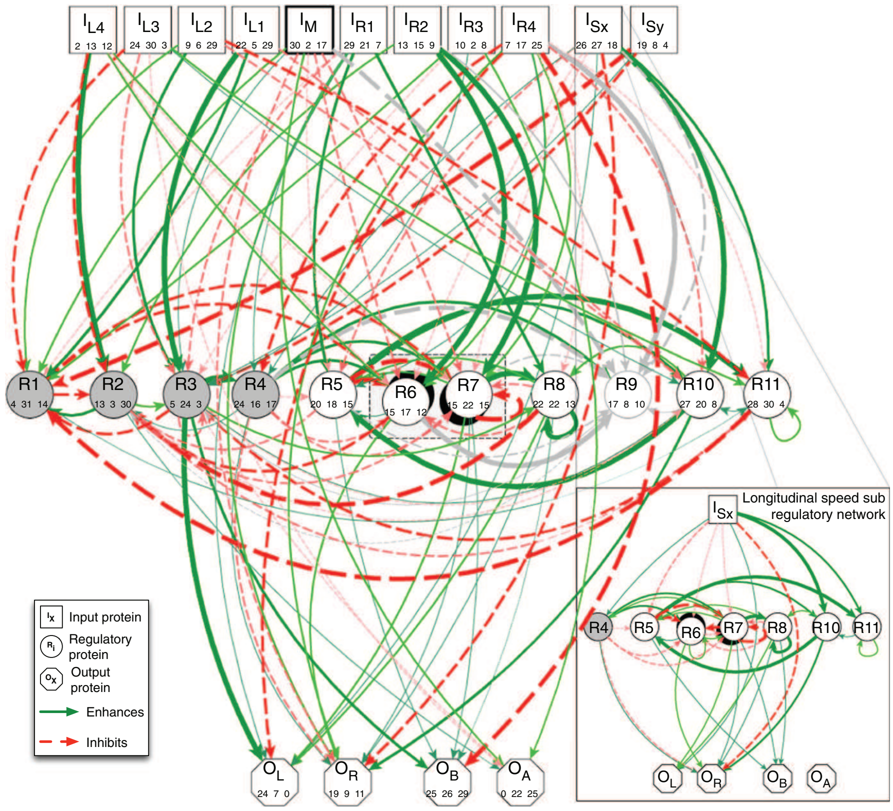

Fig. 18 Graphical representation of the GRN obtained through evolution. Nodes are the proteins (renamed I_x for input, R_i for regulatory and O_x for output proteins) and edges represent the affinity between the two proteins for enhancement (in *plain green*) and for inhibition (in *dashed red*) (Color figure online)

close or equal distances, left and right outputs are equals, steering wheel is in middle position.

The role of the lateral speed input protein I_{Sy} is also important in the steering regulation. When the car slides, I_{Sy} concentration increases. The interesting fact is that this protein inhibits proteins $R1$ and $R4$, which are proteins involved in the left steering behavior. The effect is to reduce the left steering, and consequently reduces the global use of right steering as well by propagation in the whole network in the very few next regulatory steps. This might explain the capacity of the GRN to properly drive the car in various track coating such as sand or rock.

6.1.3 Thrust regulation

The braking output protein O_B is linked to regulatory proteins $R3$, $R5$ and $R8$ and thus to track sensors from left and right sides of the car. This provides a constant, seemingly weak, enhancement. However, right input protein I_{R4} directly provides a strong inhibition to the brake output O_B. The farther the distance sensed by input protein I_{R4} (rightmost track sensor) is, the less the GRNDriver brakes (and vice versa).

The acceleration output protein O_A is directly enhanced by the middle track sensor (input protein I_M) and the second rightmost sensor (input protein I_{R3}): the farther the sensed distance is, the more GRNDriver accelerates. Left side sensors seem to have a lesser influence: second leftmost sensor (input protein I_{R3}) and regulatory proteins $R2$ and $R3$ (linked to left side sensors) inhibit and enhance the accelerate output protein, possibly canceling their mutual actions.

Another important protein for thrust regulation is the longitudinal speed input protein, named I_{Sx}. For better understanding, we have zoomed the graph of Fig. 18 with the only significant regulation proteins involved directly or indirectly with this protein (see bottom-right box). Firstly, the steering is not affected by input I_{Sx}, since I_{Sx} inhibits O_R and enhances O_R through $R10$. Therefore, the regulation flows negate each other. All other enhancing flows to both steering output proteins are inhibited by the inputs. However, concerning the car thrust, this input protein enhances $R10$ (both directly and through $R11$) that enhances $R8$ and that finally enhances slightly O_B. However, since the $R8$ is self-catalyzed, the final reaction can be substantial. In summary, the longitudinal speed protein I_{Sx} enhances the brake so that when the car goes too fast, the driver slows down.

6.2 Advantages and weaknesses of the approach

In this paper, we showed that the GRN is suitable to drive efficiently a simulated racing car. We proved the GRN naturally handles noisy sensors as well on-the-fly modification of its inputs with the aim to improve its behavior. Since the inner dynamics of the GRN can be compared to neural network with inter-connected neurons activated and inhibited by a given function, the main advantage of this approach is the compact structure of genomes. Whereas each neuron and each connection between two neurons have to be encoded in a neural network, the encoding of the GRN builds all protein interactions with only three numbers. Modifying the architecture of the protein network is therefore easier: a simple mutation on any tag value in a protein globally modify the structure of the GRN. Moreover, when subject to evolution, crossing two networks is extremely simple and efficient since all connections are coded within the proteins. However, this advantage has a side effect: some regulatory flows are hard to dissociate, due to the low number of available protein tags. For example, in Fig. 18, left steering output protein O_L and braking out protein O_B have respectively 24 and 25 tag values. This means that every protein that influences O_L influences O_B as well (and vice versa). In order to avoid that the GRNDriver brakes each time it turns left (or turns left each time it brakes), the evolutionary process has produced complex enhancement and inhibition flows to compensate this default. Unfortunately, our GA-based evolution is not always so effective and dissociation artifacts can remain. This is particularly the case with more complex problems that involves large networks. We are currently working on modifying the protein affinity formula (distance between the protein tag values) so that more regulation channels can be added by only modifying a variable (that could be subject to evolution too).

Another evolutionary side effect is the global imperfection of produced solutions: most evolved GRN present contradictory regulation flows. For example, in the GRN presented in Fig. 18, I_{L4} is enhancing and inhibiting R_2 in the same time and we can notice that the enhancer identifier of I_{L4} (which is 13) is almost equal to its inhibiting identifier (equal to 12). This means that almost all proteins enhanced by this protein will be also inhibited. This behavior is not efficient and could be improved by defining new mutation operator that would check this kind of inconsistency and solve them adequately (by generating an identifier out of the range for example).

One more advantage of the GRN is that all the GRN's variables are subject to evolution and thus do not have to be set up. The only parameters that need a set up are the ones involved in the evolutionary algorithm used to optimize the network, such as the crossover and mutation rates, the population size, the selection algorithm, etc. Using a GRN is then very easy for people with experience in evolutionary algorithms: the main difficult aspect of using a evolutionary algorithm to evolve a GRN is the formulation of the adequate fitness. But this is a usual difficulty with problems that involve an evolutionary algorithm.

However, the GRN still has weaknesses that should be addressed in order to make them more efficient or easier to use. The main difficulty about using a GRN is the connection of the input and output proteins to the problem it has to solve. As presented in this paper, since the sum of the regulatory and output protein concentrations is always equal to 1, it is usually necessary to have two outputs to obtain a continuous values: one is used as a self-adjusted threshold and the second is used to evaluate the final value according to the threshold. We are currently working on this negative aspect of the regulation by modifying the network dynamics.

7 Conclusion

In this paper, we have showed how to used a gene regulatory network to drive a virtual car. The connections between the car and the GRN have been kept as simple as possible. The GRN has been naturally resistant to sensor noise: the impact of noise on the GRN's capacity to drive the car is very low. Moreover, the GRN is able to generalize a behavior learned on asphalt tracks to other types of surfaces such as sandy and rocky tracks. The GRN evolved through a 3-steps evolutionary process has been found to be a safe driver. To make it become a real racer, we have distorded the GRN inputs to make it more aggressive with the break and accelerator and to create multiple trajectories in order to make it overtake or avoid other cars. A recovery procedure has also been implemented in order to put the car back on track when the GRN fails to handle a complex situation.

To improve this work, multiple options have to been investigated. Our goal is to design a driver with as much automatic learning as possible. First, the use of the GRN as a racing driver requires the design of a track learning method to speed up the wise GRNs we generally obtain by evolution. We would like to teach the GRN to go faster by the use of a hierarchical architecture: a second GRN, pre-optimized on multiple tracks and reoptimized during the warm-up stage, could modify the inputs and/or the outputs of the driving GRN according to the current car state. The specialization capacity of the GRN observed in the first evolutionary step could be helpful during this warm-up stage.

This GRN must also be improved in order to correctly handle opponents. For now, the perception of the GRN is modified by a hand-written script in order to overtake or avoid an opponent detected too close to the car. This approach is innovative in comparison to most other approaches because they usually directly impact the car actuators. Modifying the inputs instead of the output keeps the controller as the center piece of the algorithm. However, we want the GRN to learn to handle this move by itself because most overruns are currently due to this script. Having all the information the car can detect and letting the GRN decide the best move could reduce this issue.

A full detailed study on how the GRN actually handles noise could be interesting to conduct. Our first hypothesis is that the granularity between the protein affinities helps the network to compensate for input distortions. Studying this phenomenon precisely could help us to better understand the dynamics of the evolved GRNs and possibly to prove their capacity to handle noisy inputs.

According to past experiences using the GRN as an agent controller (in developmental models, foraging agents, pole carts, etc.), we believe this approach is now ready to be used in a wide range of agent-based problems. This method can handle uncertainty because of the kinetics of the network. The GRN is thus easy to plug to any kind of agent; the only requirement is being able to convert the input and output signals into normalized concentration values. The strength of the GRN is also in handling cooperative and conflicting behavior within the same network. In our opinion, this method can compete with neural networks, genetic programming, and other evolutionary approaches on multiple domains.

References

1. A. Agapitos, J. Togelius, S.M. Lucas, Evolving controllers for simulated car racing using object oriented genetic programming. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. ACM (2007), pp. 1543–1550
2. C. Athanasiadis, D. Galanopoulos, A. Tefas, Progressive neural network training for the open racing car simulator. In *IEEE Conference on Computational Intelligence and Games (CIG), 2012*. IEEE (2012), pp. 116–123
3. W. Banzhaf, in *Artificial regulatory networks and genetic programming*, eds. by R.L. Riolo, B. Worzel. Genetic Programming Theory and Practice, chap 4 (2003), pp. 43–62
4. M. Bednár, A. Brček, B. Marek, M. Florek, V. Juhász', J. Kosmel', I. Valenčík, The modular architecture of an autonomous vehicle controller.
5. M.V. Butz, T.D. Lönneker, Optimized sensory-motor couplings plus strategy extensions for the torcs car racing challenge. In *Proceedings of the 5th International Conference on Computational Intelligence and Games, CIG'09*.IEEE Press, Piscataway, NJ, USA (2009), pp. 317–324
6. L. Cardamone, D. Loiacono, P.L. Lanzi. Evolving competitive car controllers for racing games with neuroevolution. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, GECCO '09 pp. 1179–1186. ACM, New York, NY, USA (2009)

7. L. Cardamone, D. Loiacono, P.L. Lanzi, On-line neuroevolution applied to the open racing car simulator. In *Proceedings of the Eleventh conference on Congress on Evolutionary Computation, CEC'09*. IEEE Press, Piscataway, NJ, USA (2009), pp. 2622–2629
8. L. Cardamone, D. Loiacono, P.L. Lanzi, Learning to drive in the open racing car simulator using online neuroevolution. IEEE Trans. Comput. Intell. AI in Games **2**(3), 176–190 (2010)
9. S. Cussat-Blanc, N. Bredeche, H. Luga, Y. Duthen, M. Schoenauer, Artificial gene regulatory networks and spatial computation: a case study. In *Proceedings of the European Conference on Artificial Life (ECAL'11)*. MIT Press, Cambridge, MA (2011)
10. S. Cussat-Blanc, J. Pollack, A cell-based developmental model to generate robot morphologies. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*. ACM New York, NY, USA (2012)
11. S. Cussat-Blanc, J. Pollack, Using pictures to visualize the complexity of gene regulatory networks. Artif. Life **13**, 491–498 (2012)
12. S. Cussat-Blanc, S. Sanchez, Y. Duthen, Simultaneous cooperative and conflicting behaviors handled by a gene regulatory network. In *IEEE Congress on Evolutionary Computation (CEC), 2012*, pp. 1–8. IEEE (2012)
13. R. Doursat, Organically grown architectures: creating decentralized, autonomous systems by embryomorphic engineering. In *Organic Computing, IX* (Springer, 2008), pp. 167–200
14. P. Eggenberger Hotz, Combining developmental processes and their physics in an artificial evolutionary system to evolve shapes. In *On Growth Form and Computers* (Elsevier 2003), pp. 302–318
15. D.M. Fernández, A.J. .Fernández-Leiva, Una experiencia de diseño de controladores en juegos de carreras de coche mediante algoritmos evolutivos multiobjetivos y sistemas expertos. In *VIII Congreso Español sobre Metaheurística, Algoritmos Evolutivos y Bioinspirados*, ed. by J.A. Gámez et al. (UCLM, Albacete, 2012), pp. 683–690
16. H. Guo, Y. Meng, Y. Jin, A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. BioSystems **98**(3), 193–203 (2009)
17. K.I. Harrington, E. Awa. S. Cussat-Blanc, J. Pollack, Robot coverage control by Evolved Neuromodulation. In *IJCNN 2013* (2013)
18. M. Joachimczak, B. Wróbel, Evolving gene regulatory networks for real time control of foraging behaviours. In *Proceedings of the 12th International Conference on Artificial Life* (2010)
19. M. Joachimczak, B. Wróbel, Evolution of the morphology and patterning of artificial embryos: scaling the tricolour problem to the third dimension. In *Advances in Artificial Life. Darwin Meets von Neumann. ECAL 2009, Budapest, Hungary, Revised Selected Papers, Part II*, ed. by G. Kampis, I. Karsai, E. Szathmary (Springer, 2011), pp. 35–43
20. J. Knabe, M. Schilstra, C. Nehaniv, Evolution and morphogenesis of differentiated multicellular organisms: autonomously generated diffusion gradients for positional information. Artif. Life XI **11**, 321 (2008)
21. R. Lifton, M. Goldberg, R. Karp, D. Hogness, The organization of the histone genes in drosophila melanogaster: functional and evolutionary implications. In *Cold Spring Harbor Symposia on Quantitative Biology*, Cold Spring Harbor, NY, vol 42 (1978), pp. 1047–1051
22. D. Loiacono, L. Cardamone, P.L. Lanzi, Simulated car racing championship: competition software manual. CoRR (2013)
23. D. Loiacono, P.L. Lanzi, J. Togelius, E. Onieva, D.A. Pelta, M.V. Butz, T.D. Lönneker, L. Cardamone, D. Perez, Y. Sáez et al., The 2009 simulated car racing championship. IEEE Trans. Comput. Intell. AI in Games**2**(2), 131–147 (2010)
24. D. Loiacono, J. Togelius, P.L. Lanzi, L. Kinnaird-Heether, S.M. Lucas, M. Simmerson, D. Perez, R.G. Reynolds, Y. Saez. The wcci 2008 simulated car racing competition. In: *IEEE Symposium on Computational Intelligence and Games, 2008. CIG'08*. IEEE (2008), pp. 119–126
25. M. Nicolau, M. Schoenauer, W. Banzhaf, Evolving genes to balance a pole. In A.I. Esparcia-Alcazar, A. Ekart, S. Silva, S. Dignum, A.S. Uyar (eds.) *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010* ,vol 6021. LNCS, (2010), pp. 196–207
26. E. Onieva, D.A. Pelta, J. Alonso, V. Milanés, J. Pérez, A modular parametric architecture for the torcs racing engine. In *Proceedings of the 5th International Conference on Computational Intelligence and Games, CIG'09*. IEEE Press, Piscataway, NJ, USA (2009), pp. 256–262
27. E. Onieva, D.A. Pelta, J. Godoy, V. Milanés, J. Pérez, An evolutionary tuned driving system for virtual car racing games: the autopia driver. Int. J.Intelli. Syst. **27**(3), 217–241 (2012)

28. M. Preuss. J. Quadflieg. G. Rudolph. Torcs sensor noise removal and multi-objective track selection for driving style adaptation. In: *IEEE Conference on Computational Intelligence and Games (CIG), 2011*. IEEE (2011), pp. 337–344
29. J. Quadflieg. M. Preuss, O. Kramer, G. Rudolph. Learning the track and planning ahead in a car racing controller. In: *2010 IEEE Symposium on Computational Intelligence and Games (CIG)*, IEEE (2010), pp. 395–402
30. J. Quadflieg, M. Preuss, G. Rudolph, Driving faster than a human player. In *Proceedings of the 2011 International Conference on Applications of Evolutionary Computation-Volume Part I*. Springer (2011), pp. 143–152
31. T. Reil, Dynamics of gene expression in an artificial genome-implications for biological and artificial ontogeny. Lecture notes in computer science (1999), pp. 457–466
32. K. Stanley, R. Sherony, N. Kohl, R. Miikkulainen, Neuroevolution of an automobile crash warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (2005)
33. K.O. Stanley, R. Miikkulainen. Evolving neural networks through augmenting topologies. Evol. Comput. **10**, 99–127 (2002)
34. J. Togelius, S.M. Lucas, Evolving robust and specialized car racing skills. In *IEEE Congress on Evolutionary Computation. CEC 2006*. IEEE (2006), pp. 1187–1194
35. D. Wilson, E. Awa, S. Cussat-Blanc, K. Veeramachaneni, U.M. O'Reilly. On learning to generate wind farm layouts. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*. ACM (2013), pp. 767–774
36. L. Wolpert, Positional information and the spatial pattern of cellular differentiation. J. Theor. Biol. **25**(1), 1 (1969)