Evolving the Optimal Racing Line in a High-End Racing Game

Matteo Botta, Vincenzo Gautieri, Daniele Loiacono, and Pier Luca Lanzi

Abstract— **Finding a racing line that allows to achieve a competitive lap-time is a key problem in real-world car racing as well as in the development of non-player characters for a commercial racing game. Unfortunately, solving this problem generally requires a domain expert and a trial-and-error process. In this work, we show how evolutionary computation can be successfully applied to solve this task in a high-end racing game. To this purpose, we introduce a novel encoding for the racing lines based on a set of connected Bézier curves. In addition, we compare two different methods to evaluate the evolved racing lines: a simulation-based fitness and an estimation-based fitness; the former does not require any previous knowledge but is rather expensive; the latter is much less expensive but requires few domain knowledge and is not completely accurate. Finally, we test our approach using The Open Racing Car Simulator (TORCS), a state-of-the-art open source simulator, as a testbed.**

I. INTRODUCTION

The optimal racing line is defined as the line to follow to achieve the best lap-time possible on a given track with a given car. In general, finding the optimal racing line is a common problem in real-world car racing [6] as well as in the development of commercial racing games [12]. Unfortunately, finding an optimal racing line is not an easy task as it depends on several factors [6], [3] ranging from the shape of the track, the grip, the car aerodynamics, the power of the car engine, etc. Thus, racing lines are usually drawn by domain experts [12] and then tested and tuned by game developers through actual game-play. Accordingly, evolutionary computation might be a promising technique to support the design of optimal racing lines and to speed-up the game development process.

In a recent work [4], we applied genetic algorithms to search for the best trade-off between two given racing lines. In this paper, we extend our previous work by applying genetic algorithms to evolve the optimal racing lines from scratch. To this purpose, we introduced a new encoding for a racing lines consisting of a set of connected Bézier curves, such that each gene defines a small portion of the racing line. Therefore, while in [4] we *simply* evolve the mixing between two racing lines, here the evolution is responsible of the entire design of the racing line. In addition, in this paper we compare two different methods to evaluate the evolved racing line; the first one is based on testing the evolved racing lines in a racing simulator as done in [4]; the second one consists of estimating the performance of a racing line through a computational model.

Matteo Botta (matteo.botta@mail.polimi.it), Vincenzo Gautieri (vincenzo.gautieri@mail.polimi.it), Daniele Loiacono (loiacono@elet.polimi.it), and Pier Luca Lanzi are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy.

In this work, we used The Open Racing Car Simulator (TORCS) as a testbed and applied our approach to four different tracks. Our results show that our approach is able to evolve competitive racing lines using both the evaluation methods, although the simulation-based evaluation provide significantly better results.

II. BACKGROUND

In this section we provide some useful background for the remainder of the paper. First, we introduce the problem of finding an optimal racing line; then, we give some useful definitions about the Bézier curves used in this work; finally, we briefly describe The Open Racing Car Simulator (TORCS)

A. Optimal Racing Line

The optimal racing line is the path that a driver should follow to complete a lap on a given track in the smallest amount of time possible. As the lap-time depends both on the distance raced and on the average racing speed, finding the optimal racing involves two different sub-problems [3]: (i) racing the shortest distance possible and (ii) racing as fast as possible along the track.

Racing the shortest distance. It requires to find the shortest path along the track, i.e., the racing line within the track borders with the shortest length. In practice, racing line might be represented with different methods in a racing games (e.g., a set of waypoints [3], a sequence of parametrized curves [12], [16], etc.). Accordingly, the search space of the shortest path along the track is typically *limited* by the method used to represent the racing lines. Depending on the size of such a search space and on the representation constraints, computing *analytically* the shortest path might be either computationally unfeasible or too expensive for the development process needs.

Racing fast. The highest speed a driver can keep along a given racing line without loosing the control of the car depends on several factors and can be computed as follows [3]:

$$v_{\max} = \sqrt{\frac{\mu}{\kappa}\left(g + \frac{F_a}{m}\right)}, \qquad (1)$$

where $v_{\max}$ is the highest speed allowed, m is the mass of the car, μ is the tire-road friction coefficient, F_a the aerodynamics downforce, κ is the curvature of the given racing line, i.e., the inverse of the curvature radius. Accordingly, the smallest is the curvature of the racing line followed by the car, the highest would be the allowed speed (disregarding the mass of the car, the friction parameters, and the car aerodynamics). As in the previous problem, the method used to represent the

racing line defines the search space and, thus, the complexity of this optimization problem.

Both the problems discussed so far, i.e., racing the shortest distance and racing fast, only depends on the track shape and, thus, can be solved without any knowledge of the racing car dynamics (e.g., the acceleration profile, the braking capabilities, the aerodynamics model, etc.). Unfortunately, these problems cannot be solved independently as they have conflicting objectives: the shortest path is not usually the fastest one and vice versa. Accordingly, finding the optimal racing line requires to search for the optimal *trade-off* between racing the shortest distance and racing fast [3]. Of course, such an optimal trade-off also depends on the racing car dynamics, i.e., it is different for different car models, and on the *sequence* of the track segments (e.g., the ideal line through the same turn might change if it is followed by a straight or a tight turn).

B. Bézier Curves

Bézier curves are a family of parametrized curves widely used in computer graphics and in related fields. They have been first introduced and studied in 1959 by the mathematician Paul de Casteljau and became popular when, in 1962, the engineer Pierre Bézier employed them to design cars. Nowadays, Bézier curves are frequently used in vector graphics to model smooth *paths*, as they offer a compact and convenient representation.

A Bézier curve is defined by a set of *control points*, $\{P_0, P_1, \cdots, P_n\}$, where n is the *order* of the curve. The first and the last control points, i.e., P_0 and P_n, are respectively the beginning and the end of the curve, while the intermediate control points do not usually lie on the curve.

Figure 1 shows an example of Bézier curve defined by a set of 9 control points. Given a set of control points $\{P_0, P_1, \cdots, P_n\}$, the resulting Bézier curve is defined as follows [8]:

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i, \qquad (2)$$

where $t \in [0, 1]$ is a control variable to move along the line, such that $B(0) = P_0$ and $B(1) = P_n$.

C. The Open Racing Car Simulator

The Open Racing Car Simulator (TORCS) [7] is a state-of-the-art open source car racing simulator which provides a sophisticated physics engine, full 3D visualization, several tracks, several models of cars, and various 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 (game tick), a bot can access the current game state, which includes several information about the car and the track, as well as the information about the other cars on the track; a bot can control the car using the
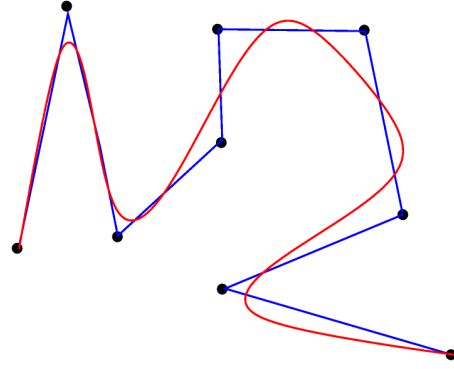

Fig. 1. An example of Bézier curve: black points are the *control point* of the curve; blue lines simply connect with straight segments the control points; red line is the resulting Bézier curve.

gas/brake pedals, the gear stick, and steering wheel. The game distribution includes many programmed bots which can be easily customized or extended to build new bots.

III. RELATED WORK

In general, commercial racing games typically rely on human-designed racing lines designed by domain experts [12] with few notable exceptions as Colin McRae Rally[1] (Codemasters), where the racing line is learned by imitation using a neural network [11], and as Forza Motorsport[2] series (Microsoft), where evolutionary computation is applied to optimize their racing lines [16].

Instead, open-source racing games typically combines good practices with heuristics to generate a racing line for any given track automatically. The most successful examples of such approaches include the K1999 algorithm [5], which was developed by Remi Coulom and exploits gradient descent; Simplix [2], developed by Wolf-Dieter Beelitz for The Open Car Racing Simulator (TORCS)[7], based on a simple heuristic; the bot by Jussi Pajala for the Robot Auto Racing Simulator (RARS) [1] applies A*; the DougE1 bot for RARS by Doug Elenveld applies a genetic algorithm.

Then, in the context of real-world car racing, Casanova [6] showed that to find the best racing line several aspects of the car dynamics must be taken into account, e.g., the braking points over the track, the acceleration capabilities of the car, the changes of direction, etc. Although theoretically well grounded, the approach in [6] requires a complete and very detailed formal model of the vehicles and of its interactions with the racing environment, which is typically unavailable in racing games.

More recently, Braghin et al. [3] suggested that racing lines can be computed by solving the trade-off between minimizing the distance raced on the track and minimizing the curvature of the racing line followed. In particular, Braghin et al. [3] defined the problem of finding the best optimal racing line as a quadratic programming problem.

Finally, in a recent work [4], inspired by the work of Braghin et al., we applied genetic algorithms to evolve the optimal racing line by searching for the best trade-off between the shortest path along the track and the racing line with the minimum curvature possible. This work differs from our previous one in several respects: (i) here we applied genetic algorithm to evolve a racing line from scratch, while in [4] we applied it only to evolve the optimal mixing between two given racing lines; (ii) in this work we proposed a novel method to encode a racing line based on a connected set of Bézier curves; (iii) here we investigated the application of an approximated fitness function that does not involve any simulation, while in [4] fitness function is computed only through simulation.

IV. EVOLVING RACING LINES

In this section we briefly describe how the racing lines are encoded in order to be evolved using a genetic algorithm and how they are evaluated.

A. Racing Line Encoding

While a *single* Bézier curve could be in principle used to represent a racing line along the whole track, this might easily result in a racing lines which lies outside the track borders; moreover, small changes to a control point of such a Bézier curve might have a large impact on the whole racing line, making difficult to design effective genetic operators. Accordingly, we represent racing lines using a sequence of connected Bézier curves. Each Bézier curve is defined only by a limited number of control points (in all the experiments reported in this paper, we used 20 as the limit for the number of control points used to define Bézier curves) and covers only a segment of the track. In addition, to have an efficient and compact representation of the racing lines, every control point is allowed to move only along an orthogonal section of the track. As illustrated in Figure 2, control points are distributed uniformly with respect to the curvature of the track: the higher the curvature the more frequent will be the control points, i.e., a racing line through a tight turn would require much more control points than a racing line which cover a straight. As a result, the encoding of the racing lines is an array of real values, such that each value defines the position of a control point along a corresponding orthogonal section of the track. The i-th gene is encoded with a real value $\alpha_i \in [-0.1; 1.1]$ which defines the position of a control point as follows:

$$P_i = (1 - \alpha_i)Q_i + \alpha_i R_i,$$

where P_i is the i-th control point, Q_i and R_i are respectively the left and the right border of the track section the control point belongs to; please notice that when α_i is greater than 1 or smaller than 0, the control point would fall slightly outside the track border.

B. Racing Line Evaluation

In this paper, we study two different methods to evaluate the evolved racing lines, i.e., two different methods to
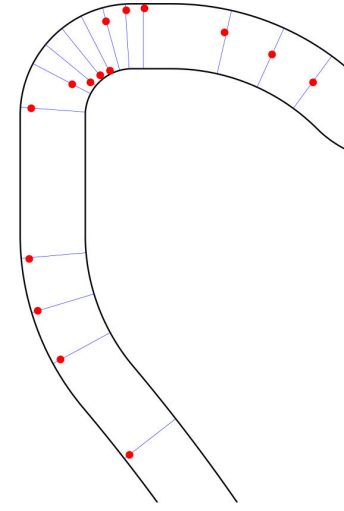

Fig. 2. Control points used to represent a racing line in a segment of track with a Bézier curve; black lines are the border of the track; blue dashed lines are the orthogonal sections of the track where control points are allowed to move; red points are the *control point* of the curve.

compute the fitness function of the candidate solutions. The first method involves the full simulation of a complete lap on the track, while the second is based on an estimate of the performance of the racing line using a computational model. While the first approach is very accurate approach and does not require any previous knowledge, the second one might be computationally less expensive.

Simulation-based evaluation. Computing the fitness function of the racing lines with this approach is straightforward. It involves a simulation of a complete lap following the racing line to evaluate. To this purpose, we modified *Simplix* [2], one of the best performing controller available for TORCS, to follow a given racing line. Therefore, the evaluation process consists of two simple steps: (i) the racing line to evaluate is loaded by the controller and (ii) the controller completes two laps following the target racing line. Accordingly, the fitness function is computed as the lap-time achieved during the second lap with a negative sign, i.e., the smallest is the lap-time the largest is the value of the fitness function. While this approach provide an exact evaluation of the racing lines (i.e., the fitness function is actually the real performance[3] that can be achieved following the racing line in the actual game), it is also rather expensive in computational terms as it requires a full simulation. In addition, this approach does not require any previous domain knowledge as the fitness is the result of a simulation.

Estimation-based evaluation. This method relies on a computational model which provides an estimate of the lap-time that could be achieved following the racing line to evaluate. Such a model typically involves also the knowledge of the

---

[3]Please, notice that the achieved performance and the optimal racing lines evolved depend on the controller used in the simulator to evaluate the solutions. Accordingly, using a different controller for the fitness evaluations might lead to different evolved racing lines as well to different performances.

car dynamics and of the track grip in order to estimate the speed that can be reached in each point of the racing line. As soon as an estimate of the lap-time is available, the fitness function is computed as in the simulation-based method, i.e., the fitness function of a racing line is the estimated lap-time following it with a negative sign. Although this method requires previous domain knowledge and the accuracy of the fitness function computed depends on the model itself, it is generally significantly less expensive than the simulation-based evaluation. In fact, with this method, the evaluation of the racing line does not require any simulation but involves only some computations. In particular, in this work we exploited the lap-time estimator built in *Simplix* [2]: using our modified version of *Simplix* we load the target racing line to evaluate and let the controller analyze it and estimate the associated lap-time.

V. EXPERIMENTAL RESULTS

In this section, we first describe the experimental setup and then report the results obtained with our approach using simulation-based and the estimation-based evaluations.

A. Experimental Design

In this work we used TORCS, an open-source racing simulator (see Section II-C), as testbed. In particular, we used a slightly modified implementation of TORCS (ver. 1.3.1) similar to the one used for the Simulated Car Racing Competitions [14], [13]. Such a modified implementation, offers some additional features like the possibility of running batch simulation without graphical output.
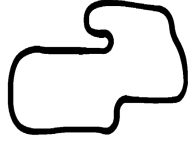
Table I shows the four tracks used for the experimental analysis in this work: A-Speedway, CG Speedway 1, Ruudskogen, and Alpine 2. They are all provided with the standard distribution of TORCS and offers different degree of challenges: A-Speedway is a fast oval track, which requires few control points; CG Speedway 1 is a short and rather fast track, which requires a quite small number of control points; Ruudskogen alternates quite fast sections to challenging turns and require an even higher number of control points; finally, Alpine 2, is the most challenging tracks and requires many more control points.

B. Experiments with Simulation-Based Evaluation

In the first set of experiments, we applied genetic algorithms to search for the optimal racing line on a target track using the simulation-based evaluation. To this purpose we used the C++ Genetic Algorithm Toolbox [15] with the following parameters setting: the population size was set to 300; tournament selection without replacement was used and the tournament size was set to 2; one point crossover is applied with probability $p_\chi = 0.9$ and each gene is mutated using a Gaussian mutation with probability $p_m = 0.1$. For each one of the four tracks considered in this paper (Table I), we performed 5 runs and each run was stopped as soon as 300 generations were reached.

Figure 3 shows the performance of the evolved racing lines on the four target tracks and compares it to the performance

Alpine 2	**A-Speedway**
Length: 3775.57 m	Length: 1908.32 m
Control Points: 234	Control Points: 75

CG Speedway	**Ruudskogen**
Length: 2057.56 m	Length: 3274.20 m
Control Points: 100	Control Points: 159

achieved by the *Simplix* controller [2] on the same track. For the evolved racing line, we report both the average performance of the population (line with empty boxes in Figure 3) and the performance of the best individual evolved over the generations (line with filled boxes in Figure 3). The results show that on all the four tracks, the best individual evolved is finally able to outperform the *Simplix* controller. As expected the results also show that the most complex is the track, the highest is the number of generations necessary to reach the performance of *Simplix*.

C. Experiments with Estimation-Based Evaluation

In the second set of experiments, we investigate the estimation-based approach to evaluate the evolved racing line. Accordingly, we repeated exactly the same set of experiments described previously with the only exception that the estimation-based evaluation was used instead of the simulation-based one. As in the previous set of experiments, we performed 5 evolutionary runs for each track.

Figure 4 compares, for each target track, the *estimated* performance of the evolved racing lines to the performance of the *Simplix* controller. That is, both the average performance of the population (lines with empty boxes in Figure 4) and the performance of the best individual (lines with empty boxes in Figure 4) are *estimates* provided by the evaluation model used to compute the fitness of the individuals. While the results show that the genetic algorithms are still able to improve the performances over the the generations, in this second set of experiments the final performance achieved on the more complex tracks (i.e., Ruudskogen and Alpine 2) appears to be rather distant from the performance of *Simplix*.

Finally, we validated the performance achieved using the estimation-based evaluation methods as follows. For each run, we re-evaluated the best 30 racing lines evolved through simulation in TORCS. Table II shows the results of such val-
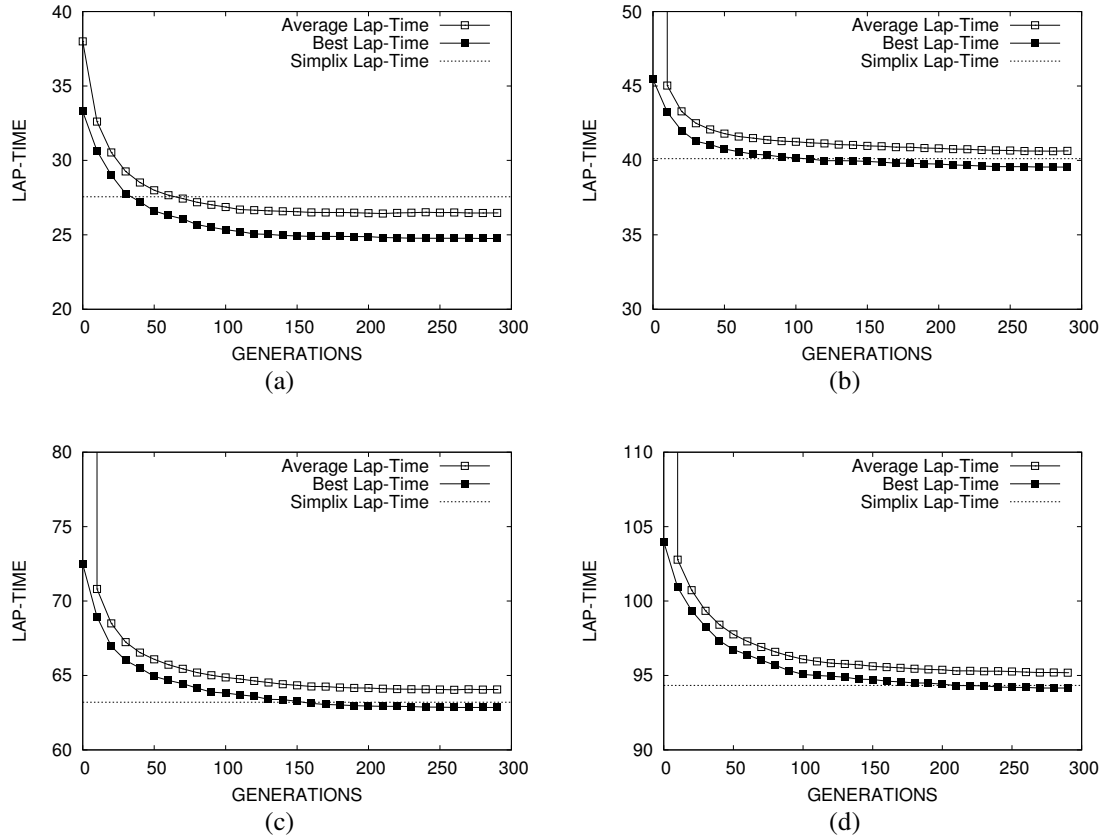
Fig. 3. Evolution of the optimal racing line on four tracks of TORCS, using simulation-based evaluation: (a) A-Speedway, (b) CG Speedway 1, (c) Ruudskogen, and (d) Alpine 2. The figure shows the average lap-time of the evolved population (empty boxes) and the best lap-time (filled boxes). Curves are averages of 5 runs.

idation process; the column labeled as `Estimated` reports the computed by the estimation-based evaluation model; instead, the column labeled as `Real`, reports the actual performance of the evolved racing lines when used in the simulator; finally, as comparison, the table reports also the performance of the *Simplix* controller on the same tracks. Data show that the *real* performance achieved using the estimation-based evaluation is better than the performance of *Simplix* only in the A-Speedway track. On the other hand, we wish to stress that the performances using an *approximated* fitness function are very close to the performance of *Simplix* on all the tracks except for the Alpine 2 track. It is also interesting to note that the accuracy of the estimates seem rather uncorrelated both with the complexity of the track; in fact, estimated performance appears to be almost equally accurate on the more complex track considered, i.e., Alpine 2, as well as on the most simple one, i.e., A-Speedway.

Concerning the computational cost, evolutionary runs using simulation-based evaluations required 90000 simulations (i.e., 300 individuals for 300 generations) which take approximately from 12 to 36 hours depending on the track. In contrast, using estimation-based evaluation, each run required approximately between slightly more than 3 hours and 5 hours. As expected, the estimation-based evaluation is much

TABLE II

PERFORMANCE OF THE BEST RACING LINES EVOLVED FOR EACH TRACK USING THE ESTIMATION-BASED EVALUATION. STATISTICS ARE COMPUTED OVER 5 RUNS.

Track	Lap-Time (s)		
	Estimated	**Real**	**Simplix**
A-Speedway	24.75 ± 0.04	24.89 ± 0.00	27.56
CG Speedway 1	39.07 ± 0.18	40.38± 0.08	40.12
Ruudskogen	65.12 ± 0.13	63.52 ± 0.01	63.20
Alpine 2	96.38 ± 0.12	96.56 ± 0.10	94.33

more cheap than the simulation-based one. In addition, the implementation of the estimation-based evaluation might be still improved as it is currently based on a module of *Simplix* and currently involves expensive and useless operations that could be removed such as loading the 3D model of the track and the controller module.

VI. ANALYSIS OF THE RESULTS

In this section, we first provide an insight about the structure of the problem of searching the optimal racing line. Then, we briefly compares the performance of our approach to others previously introduced in the literature.
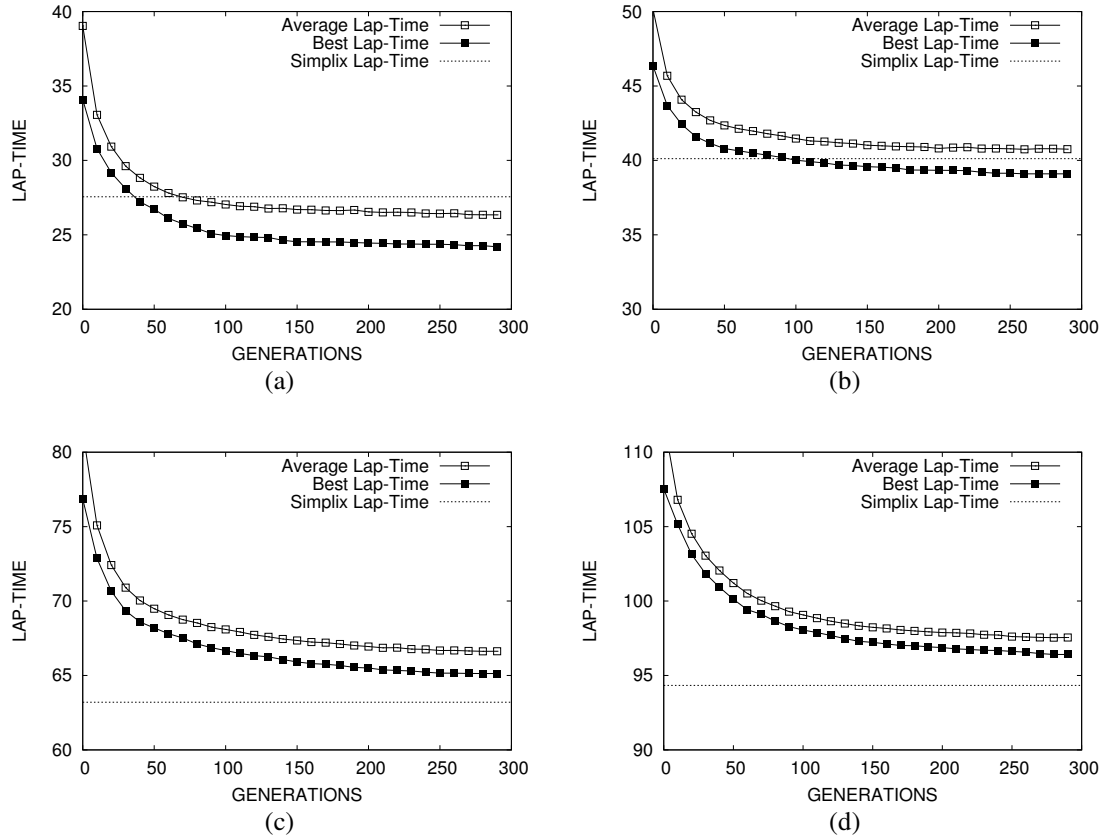
Fig. 4. Evolution of the optimal racing line on four tracks of TORCS, using estimation-based evaluation: (a) A-Speedway, (b) CG Speedway 1, (c) Ruudskogen, and (d) Alpine 2. The figure shows the average lap-time of the evolved population (empty boxes) and the best lap-time (filled boxes). Curves are averages of 5 runs.

A. Analysis of the Problem Structure

Understanding the underlying problem space is the key to design a better encoding which is compact but, at the same time, does not introduce any bias in the search process [10]. Therefore, we performed an analysis of the experiments reported in the previous section to discover the building blocks of the evolved solutions. To this purpose, we measured convergence speed of each gene during the evolutionary process as follows. For each gene, we computed the standard deviation of the values it assumed in the population. Thus, we defined the convergence speed of each gene as the number of generations taken to reach a standard deviation equal or below a given threshold (in our analysis we used as a threshold the 20% of expected standard deviation according to the prior distribution).

Figure 5 shows the convergence speed of each gene represented as a colored point on the track: the position of the point corresponds to the position of the control point associated to the gene, the color of the point represents the convergence speed; to improve the readability of the results we discretized the convergence speed in 5 intervals and labeled them as *Very Fast*, *Fast*, *Normal*, *Slow*, and *Very Slow*. Data in Figure 5 has been computed for the experiments performed using the simulation-based evaluation described

in Section V-B. However, we repeated the same analysis (not reported here) also for the experiments involving the estimation-based evaluation and obtained similar results.

This analysis provide several interesting insights about the problem. First of all, most of the points converges very slowly or does not converge at all (dark blue points in Figure 5), suggesting that relatively a small number of points have a major impact on the final outcome. Second, the most important control points, i.e., the ones associated to genes which converge fast or very fast (dark red points in Figure 5), are typically distributed either in the middle or at the end of a turn. This can be easily explained by noting that these control points define the *apex* of the line and affects the speed of the car when it exit from the turn. In contrast, the points which lie on a straight appears much less important as the position of the car on a straight does not affect too much the performance; an interesting exception to this consideration is represented by the control points within the short straights of A-Speedway (see Figure 5); however, those points should be actually considered within a curve as they are exactly between two very fast turns and heavily affect the racing line through them.
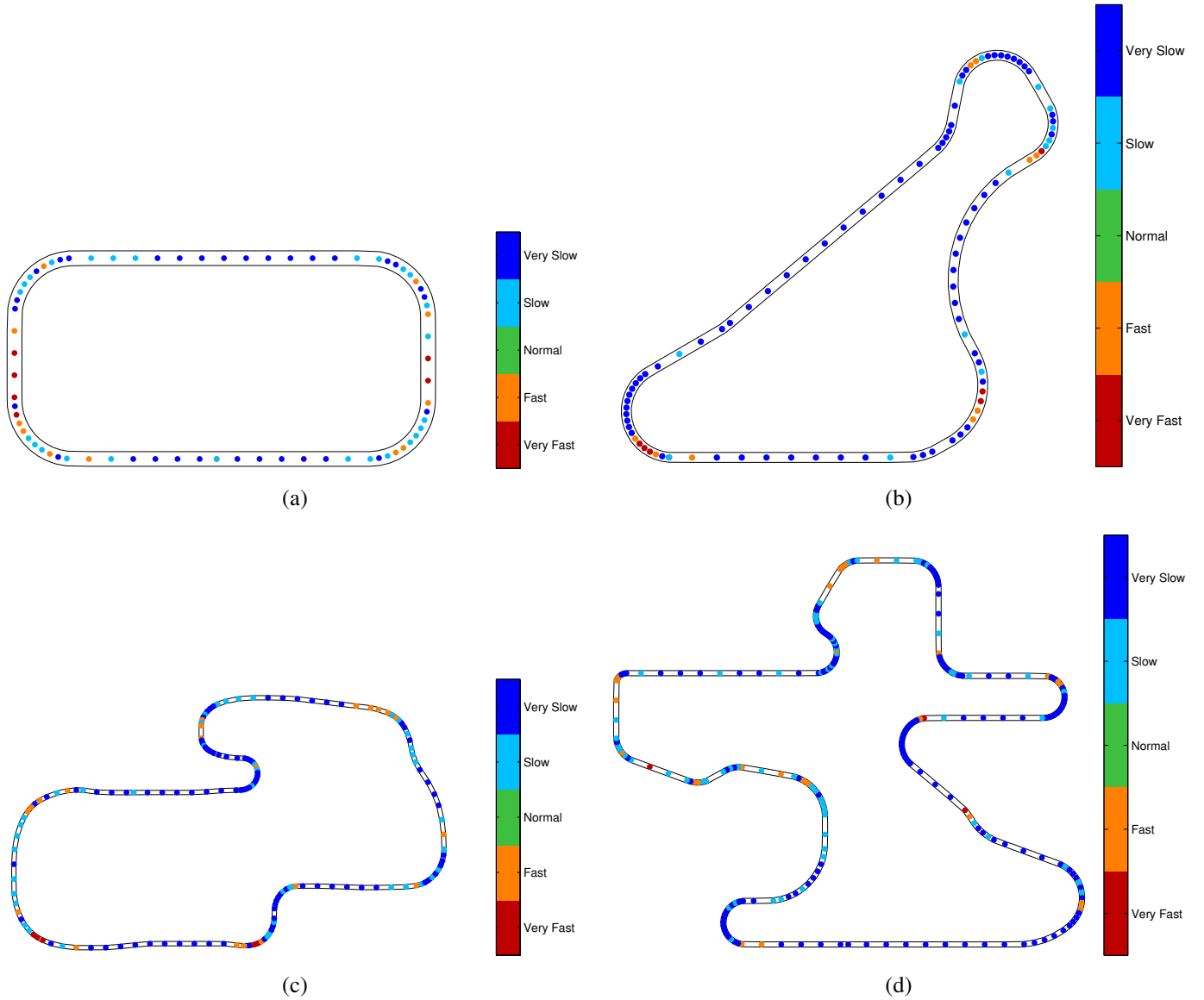
Fig. 5. Convergence speed of all the genes associated to the control points of the racing line on the four test tracks. Statics are computed as averages of the 5 runs described in Section V-B.

TABLE III

COMPARISON OF THE BEST PERFORMANCE ACHIEVED WITH (I) *Simplix* CONTROLLER, (II) APPROACH PRESENTED IN [4], (III) OUR APPROACH WITH SIMULATION-BASED EVALUATION, AND (IV) OUR APPROACH WITH ESTIMATION-BASED EVALUATION.

Track	Simplix	Cardamone et al. [4]	Simulation-Based Evaluation	Estimation-Based Evaluation
A-Speedway	27.56	**24.70 $\pm$ 0.00**	**24.76 $\pm$ 0.04**	24.89 $\pm$ 0.00
CG Speedway 1	40.12	**39.37 $\pm$ 0.00**	**39.55 $\pm$ 0.13**	40.38 $\pm$ 0.08
Ruudskogen	63.20	**62.73 $\pm$ 0.00**	**62.85 $\pm$ 0.10**	63.52 $\pm$ 0.01
Alpine 2	94.33	**92.53 $\pm$ 0.01**	94.16 $\pm$ 0.09	96.56 $\pm$ 0.10

B. Performance Analysis

Table III compares the best performances achieved with the approach presented in this paper, both using the simulation-based and the estimation-based evaluation, to the performance achieved with the *Simplix* controller and to the approach presented by Cardamone et al. in [4]. The data shows that the results obtained in [4] are slightly better than the ones achieved by the approach presented in this paper when the simulation-based evaluation is used. However, both these evolutionary approaches are able to outperform *Simplix* in all the four tracks. In contrast, using the estimation-based evaluation leads to performances notably worse than the ones achieved using the simulation-based evaluation and slightly worse of the performance achieved by *Simplix* controller.

To investigate whether the differences reported in Table III are statistical significant, we performed a statistical analysis using the *Wilcoxon Signed Rank* non parametric est [9]. Our analysis suggests that the differences between the approach presented here (with the simulation-based evaluation) and the approach introduced by Cardamone et al. in [4] are not statistically significant except for the Alpine 2 track where our approach performs significantly worse (with a confidence level of 99%). When comparing our approach (with the simulation-based evaluation) to *Simplix*, the statistical analysis shows that the differences are statistically significant in all the tracks (with a confidence level of 99%) except for the Alpine 2 track. Finally, the differences the simulation-based and the estimation-based evaluation resulted statistically significant on all the tracks considered.

Overall, our results suggest that, despite involving a rather larger search space, evolutionary computation can be applied to evolve from scratch an optimal racing line optimizing the position of its control points. However, the performance achieved on the most complex track, i.e., Alpine 2, also suggests that our approach might not scale up as well as the the one previously introduced in [4]. Finally, the results also suggest that evaluating the performance of the evolved racing lines without using a simulation-based approach might easily lead to significantly worse performances.

VII. Conclusions

In this paper, we proposed an approach to evolve an optimal racing line to support the development of a modern racing games. To this purpose, we proposed to encode the racing lines using a set of connected Bézier curves and compared two different evaluation methods to compute the fitness function: a simulation-based evaluation and an estimation-based evaluation. The former relies on a simulation of the evolved racing lines to compute its fitness, while the latter computes the fitness on the basis of an analysis of the racing line with a computational model. Then, we tested our approach applying it to four tracks of The Open Car Racing Simulator (TORCS), an open source racing simulator.

The results obtained using the simulation-based evaluation showed that our approach is able to outperform *Simplix* one of the best controller available for TORCS and to achieve performances similar to the ones achieved with a different evolutionary approach introduced in [4]. However, both the performance achieved on the most complex track, i.e., Alpine 2 track, and the analysis of the evolutionary runs suggested that our approach might benefit from a more compact encoding.

When the estimation-based evaluation was used, our approach was not able to achieve the same performances of the *Simplix* controller. Such a result suggests that the computational model used to estimate the fitness of the racing line is not able to capture completely the underlying problem. Accordingly, despite being much less expensive in terms of computational resource, the application of the estimation-based evaluation requires further investigation in order to be successfully applied to evolve optimal racing lines.

References

[1] Robot auto racing simulator. http://rars.sourceforge.net/.

[2] Wolf-Dieter Beelitz. The SIMPLy mIXed best practice TORCS robot. http://www.wdbee.gotdns.org:8086/SIMPLIX/SimplixDefault.aspx.

[3] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. Race driver model. *Comput. Struct.*, 86(13-14):1503–1516, 2008.

[4] L. Cardamone, D. Loiacono, P.L. Lanzi, and A.P. Bardelli. Searching for the optimal racing line using genetic algorithms. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 388 –394, aug. 2010.

[5] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control.* PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[6] R.S. Sharp D. Casanova. *On minimum time vehicle manoeuvring: the theoretical optimal time.* PhD thesis, Cranfield University, 2000.

[7] Eric Espié, Christophe Guionneau, Bernhard Wymann, Christos Dimitrakakis, Rémi Coulom, and Andrew Sumner. TORCS, the open racing car simulator. http://www.torcs.org, 2005.

[8] G.E. Farin. *Curves and surfaces for computer-aided geometric design: a practical guide.* Number v. 1 in Computer science and scientific computing. Academic Press, 1997.

[9] J. D. Gibbons. *Nonparametric Statistical Inference.* Marcel Dekker, 1985.

[10] David E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[11] Jeff Hannan. Interview to jeff hannan, 2001. http://www.generation5.org/content/2001/hannan.asp.

[12] Stefano Lecchi. Artificial intelligence in racing games. In *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*, pages 1–1, Piscataway, NJ, USA, 2009. IEEE Press.

[13] D. Loiacono, P.L. Lanzi, J. Togelius, E. Onieva, D.A. Pelta, M.V. Butz, T.D. Lonneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg. The 2009 simulated car racing championship. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(2):131 –147, jun. 2010.

[14] D. Loiacono, J. Togelius, P.L. Lanzi, L. Kinnaird-Heether, S.M. Lucas, M. Simmerson, D. Perez, R.G. Reynolds, and Y. Saez. The wcci 2008 simulated car racing competition. In *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, pages 119–126, Dec. 2008.

[15] K. Sastry. Single and multiobjective genetic algorithm toolbox for matlab in c++. Technical Report 2007017, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, 2007.

[16] David Stern and Joaquin Qui nonero Candela. Playing machines: Machine learning applications in computer games. In *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*, pages 1–1, Piscataway, NJ, USA, 2009. IEEE Press.