Automatic Track Generation for High-End Racing Games Using Evolutionary Computation

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

Abstract— **We investigate the application of evolutionary computation to the automatic generation of tracks for high-end racing games. The idea underlying our approach is that *diversity* is a major source of challenge/interest for racing tracks and, eventually, might play a key role in contributing to the player's fun. In particular, we focus on the diversity of a track in terms of its shape (i.e., the number and the assortments of turns and straights it contains), and in terms of driving experience it provides (i.e., the range of speeds achievable while driving on the track). We define two fitness functions that capture our idea of diversity as the entropy of the track's curvature and speed profiles. We apply both a single-objective and a multi-objective real-coded genetic algorithm to evolve tracks involving both a wide variety of turns and straights and also a large range of driving speeds. The results we report show that both single-objective and multi-objective approaches can successfully evolve tracks with a high degree of diversity both in terms of shape and achievable speeds.**

I. Introduction

The automatic generation of game content has been a central issue for the game industry since the early 1980s, when the limitation of existing platforms did not allow the distribution of large amounts of pre-designed content (typically game levels). Accordingly, ad-hoc algorithmic procedures were widely applied to generate game content on-the-fly, so as to provide infinite levels of fun [1], and the field of procedural content generation (or PCG) was born. Nowadays, procedural content generation is widely applied by the game industry and its importance for the development has dramatically increased (i) to control the design costs as games get bigger and bigger, (ii) to retain the crucial ability to make changes throughout the production process [2], and also (iii) to allow the emergence of new game types centered around content generation (e.g., Maxim's Spore) [3].

Procedural content generation is usually based on a constructive approach [3] in which human designers develop and tune ad-hoc algorithmic procedures to generate huge amount of interesting and diverse game content (e.g., *SpeedTree*[1], *Subversion*[2]). In the recent years however, researchers have been investigating generative-and-test approaches [3], in which methods of evolutionary computation are exploited to automatically discover innovative and interesting content. This research area has been recently dubbed Search-Based Procedural Content Generation [3].

Racing games, a rather popular game genre, can be classified into two main categories. Some racing games reproduce an actual event (for instance, EA F1-2010 reproduces the 2010 F1 championship) and therefore involve predefined game content such as specific car models (e.g., the Ferrari F2010), drivers (e.g., Fernando Alonso) and tracks (e.g., Monza). Others do not reproduce a specific event but take place in a fictional game universe and thus focus on the driving experience itself. These games are not bounded to a predefined reality (e.g., the F1 2010 championship) and therefore provide a rich set of car models, tracks, and scenarios. Accordingly, in these games, the content plays a key role for the commercial success of the title (see for instance, Trackmania by Nadeo[3] and rFactor by Image Space Inc.[4]). In this paper, we focus on the latter type of racing games, involving a fictional game universe, and investigate the application of search-based procedural content generation to evolve tracks for a high-end car racing simulator, The Open Source Racing Car Simulator (TORCS) [4].

Our approach is inspired by the seminal work of Togelius et al. [5], [6], [7], who applied evolutionary computation to evolve tracks for a simple two-dimensional car racing game. However, while Togelius et al. [5], [6], [7] address the evolution of personalized content for a target player, in this work, we focus on the evolution of tracks that can provide a *large degree of diversity*, in a rather broad sense, while also being adequately challenging. For this purpose, we extend the *radial representation* of Togelius et al. [6] and adapt it to a high-end simulator like TORCS. We define (i) the curvature profile of a track, as the distribution of curvature values of all the track segments, and the (ii) speed profile of a track, as the distribution of the speed values that a competitive driver achieves along the track. We finally define the fitness of a track either as (i) the entropy of its curvature profile or (ii) the entropy of its speed profile. We apply both single-objective and multi-objective genetic algorithms to evolve tracks that maximize these fitness functions. Our results show that the proposed approach can successfully evolve a wide variety of feasible TORCS tracks with a large degree of diversity both in terms of shape and in terms of achievable speeds. In particular, our results suggest that multi-objective genetic algorithms are well-suited for this application and can generate tracks providing the best trade-

Daniele Loiacono (loiacono@elet.polimi.it), Luigi Cardamone (cardamone@elet.polimi.it), and Pier Luca Lanzi (lanzi@elet.polimi.it) are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy. Pier Luca Lanzi (lanzi@illigal.ge.uiuc.edu) is also member of the Illinois Genetic Algorithm Laboratory (IlliGAL), University of Illinois at Urbana Champaign, Urbana, IL 61801, USA.

[1]http://www.speedtree.com/

[2]http://www.introversion.co.uk/subversion/ (Introversion Software)

[3]http://www.trackmania.com

[4]http://www.rfactor.net

off between the two objectives. At the end, we perform a preliminary validation with human subjects to test whether there is an agreement between our fitness definitions and the users' preferences. Our results suggest that there is a statistically significant correlation between our metrics and the preferences expressed by the human subjects.

The paper is organized as follows. At first, in Section II, we briefly overview the field of Procedural Content Generation. Then, in Section III, we discuss the evolution of racing tracks, overview the few published works on this topic, and we present our approach. Next, in Section IV, we briefly overview the car racing game we used (TORCS). In Section V and Section VI we discuss our approach in details describing our encoding of tracks and the metrics we used for their evaluation. In Section VII, we present the results of our experiments while in Section VIII we report on a preliminary validation we performed with human subjects. Finally, in Section IX, we draw some conclusions and discuss future research directions.

II. Related Work

In this section, we provide a brief overview of the two major areas that deal with the automatic generation of game content: Procedural Content Generation and Search-Based Procedural Content Generation [3].

A. Procedural Content Generation

Procedural content generation dates back to the early 1980s when simple algorithmic procedures were applied to generate huge, possibly infinite, amount of game content with very limited resources. One of the early example in this area is *Rogue*[5], an ASCII role-playing game by Michael Toy and Glenn Wichman in which an unlimited number of dungeon maps, filled with monsters and treasures, were procedurally generated on-the-fly [1]. In 1984, the space trading game *Elite*[6] by Ian Bell and David Braben (Acornsoft 1984) provided an expansive environment with eight galaxies, each containing 256 stars. At each location, the player could fight pirates, dock at a space-station, and trade goods carried aboard their ship. Apart from providing (for that time) stunning 3D graphics, *Elite* also used a procedural approach to generate the content of each galaxy and needed only few kilobytes to store the entire universe. In 1989, *MidWinter*[7], a first-person action role-playing game by Microplay, required only less than half a megabyte to store a huge post-apocalyptic scenario spanning for 410000km^2. Nowadays, although the memory limitations are long forgotten, procedural content generation is still widely used both to reduce the design costs and to generate those immense scenarios which would be infeasible for human designers. Recent examples include, *Diablo*, *Diablo II*[8], and *Sid Meyer's Civilization*[9] which apply level-generation methods similar (in principle) to the one used in *Rogue* [1]. The massively multi-player on-line space-game *Eve Online*[10] involves a universe generated using fractal methods, conceptually similar to what was done in *Elite*. In the game *Spore*[11], players can create their own "creature" and procedural methods are applied to animate the huge variations of possible creature creations. The game *.kkrieger*[12], developed by the *.theprodukkt* team, is an extreme example of procedural content generation as the entire demo game and all its game content fits into only 97,280 bytes. In *Left4Dead*[13], the enemies and the objects are generated based on the players movements so that they will appear outside its view so as to scare the player. In the sequel, *Left4Dead 2*, the geometry and the content of the levels evolve according to the game-play. *Borderlands*[14] provides a huge arsenal of three millions weapons generated using a single parameter vector [8]; *Far Cry 2*[15] applies procedural content generation for the weather, the day cycle, and the fire propagation. *Subversion* exploits a procedural city generator to create the game world.

More information about procedural content generation is available at several websites[16]. Andrew Doull recently wrote a survey available on line [9].

Procedural content generation methods are also available as development tools for professionals. For instance, *SpeedTree* is a powerful toolkit that can generate whole areas of 3D vegetation for games, films, and animations, based on just few parameters.

B. Search-Based Procedural Content Generation

In the recent years, researchers from academia have investigated generative-and-test approaches for procedural content generation in which methods of evolutionary computation are exploited to automatically discover innovative and interesting content. This research area has been recently named Search-Based Procedural Content Generation [3].

Hastings et al. [10] introduced NEAT Particles, a modified version of NEAT [11] which could be applied to interactively evolve complex and interesting graphical effects to be embedded in computed games so as to enrich their content. The work was extended in [12] where the authors introduced the game *Galactic Arms Race* and provided the first demonstration of evolutionary content generation in an on-line multi-player game. In particular, the authors applied another extension of NEAT, called context-generating NEAT, to evolve new weapons on-line based on what players used more often.

Marks and Hom [13] were the first to evolve a set of game rules to obtain a balanced board game which could be equally

[5]http://en.wikipedia.org/wiki/Rogue_(computer_game)
[6]http://en.wikipedia.org/wiki/Elite_(video_game)
[7]http://en.wikipedia.org/wiki/Midwinter_(video_game)
[8]http://www.blizzard.com (Blizzard)
[9]http://www.civilization.com/

[10]http://play.eveonline.com/ (CCP)
[11]http://www.spore.com (Maxis 2008)
[12]http://www.theprodukkt.com/
[13]http://www.l4d.com (Valve Corporation)
[14]http://www.borderlandsthegame.com (Gearbox Software, 2009)
[15]http://www.fracry2.com (Ubisoft)
[16]http://groups.google.com/group/proceduralcontent
http://game.itu.dk/pcg/
http://pcgames.fdg2010.org/
http://pcg.wikidot.com

hard to win from either side and rarely ended with a draw.

Togelius et al. [6] combined procedural content generation principles with an evolutionary algorithm to evolve racing tracks for a simple 2D game which could fit a target player profile (see Section III for details). Later, Togelius and Schmidhuber [14] evolved complete rule sets for games. The game engine was capable of representing simple Pacman-style games, and the rule sets described what objects were in the game, how they moved, interacted (with each other or with the agent), scoring and timing. The fitness function measured how fast another evolutionary algorithm could learn to play the game. The idea behind this fitness function is that fun is dependent on learning, and that a good game is not winnable without learning it, but should be learned quickly.

Frade et al. [15] applied Genetic Programming [16] to the evolution of terrains for games which would attain the aesthetic feelings and desired features of the designer. The approach is currently employed in the game *Chapas*[17].

El-Nasr et al. [17] presented an adaptive lighting design system, called Adaptive Lighting for Visual Attention (ALVA), that dynamically adjusts the lighting color and brightness to enhance visual attention within game environments using features identified by neuroscience, psychophysics, and visual design literature.

Zheng and Kudenko [18] presented an approach to generate commentary to football games in *Championship Manager 2008 (CM2008)*[18] by learning to map game states to high-level commentary concepts.

In [19], Pedersen and colleagues present novel methods for player modeling and suggest that the same methods might be very useful for the automatic generation of game content

More recently, Togelius et al. [20] applied multi-objective evolutionary computation to evolve maps for *StarCraft* using a set of fitness functions evaluating the player's entertainment. Sorenson and Pasquier [21] presented a generative approach for level creation following a top-down approach and validated it using *Super Mario Bros.* and a 2D adventure game similar to the *Legend of Zelda*[19].

III. Evolving Tracks for Racing Games

Racing games are a popular game genre with a rather rich catalog of titles which ranges from games that reproduce an actual event to games that take place in a fictional game universe. The former ones base their commercial success on the accurate reproduction of reality and therefore their game-play is restricted to a well-defined set of tracks such as the ones involved in the F1 championship. In contrast, the latter ones are not bound to a specific event and therefore their game-play is not restricted to a specific set of tracks. Accordingly, they base their commercial success on their capability of providing a rich set of high quality tracks (e.g., Trackmania by Nadeo) and provide the ideal application domain for the methods of search-based procedural content generation. In fact, the possibility of generating a virtually infinite number of high-quality tracks may represent an attractive feature for the players, who would get infinite fun, but also for the developers and the publishers, who can inexpensively extend the game's shelf life.

A. Previous Approaches

To the best of our knowledge, Togelius et al. [6] published the most relevant work about the evolution of tracks for racing games (see also the first early [5] and the later survey [7]). In [6], the authors combined procedural content generation principles and an evolutionary algorithm to evolve racing tracks that would fit a target player profile. As the very first step, they designed a test track featuring two long straights, a long smooth curve, and a sequence of sharp turns. They uniformly distributed a set of control points along the track (see Figure 1) and profiled a human player by recording (i) the number of control points it passed in a fixed amount of time, (ii) its driving speed and (iii) its orthogonal deviation on each control point. Then, they applied an evolutionary algorithm to evolve a controller that could mimic the player's driving profile. Finally, they used the same controller to evolve tracks which would deliver the right amount of varying challenge to the human player with fast driving sections.

In [6], Togelius et al. experimented with two main encodings. The first encoding represents a track as a set of points in the Cartesian space while the initial population is either seeded with random tracks or with just one simple track resembling a rectangular shape; Gaussian mutation is applied to perturb the points coordinates. The second *radial* encoding represents a track as a set of points with a radial configuration; the initial population is seeded with circular tracks in which all the points are at the same distance from the track center; mutation randomly changes the distance between the points and the center of the track while the angular position stays the same. These sets of points are used to generate the actual track as a sequence of Bezier curves joined together to form a b-spline.

B. Our Approach

We propose an evolutionary approach for the off-line generation of TORCS tracks. Our approach differs from the work of Togelius et al. [5], [6], [7] in several respects.

Our representation extends the *radial encoding* of [6] by including two additional parameters to each control point, namely, the angular coordinate and the slope of the track tangent line.

Our mapping between the encoding (the genotype) and the actual track (the phenotype) has to deal with the several constraints that a high-end simulator like TORCS introduces. For instance, in the 2D arcade racing game considered in [6] there is only one constraint on the first and second derivative of subsequent Bezier curves which must be equal. However, in a racing game like TORCS, tracks need to be

[17]https://forja.unex.es/projects/chapas
[18]http://www.championshipmanager.co.uk
[19]Miyamoto, S., Nakago, T., Tezuka, T.: The Legend of Zelda. Nintendo (1986)

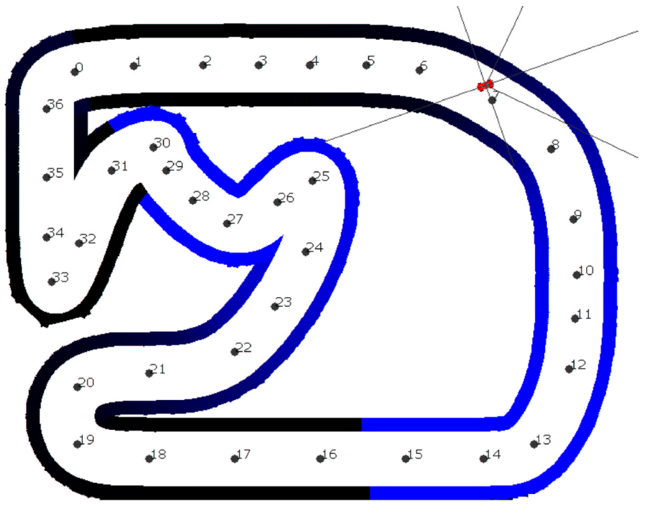


Fig. 1. The test track used by Togelius et al. [6] to generate the player profile. The image appears as Figure 2 in the original paper.

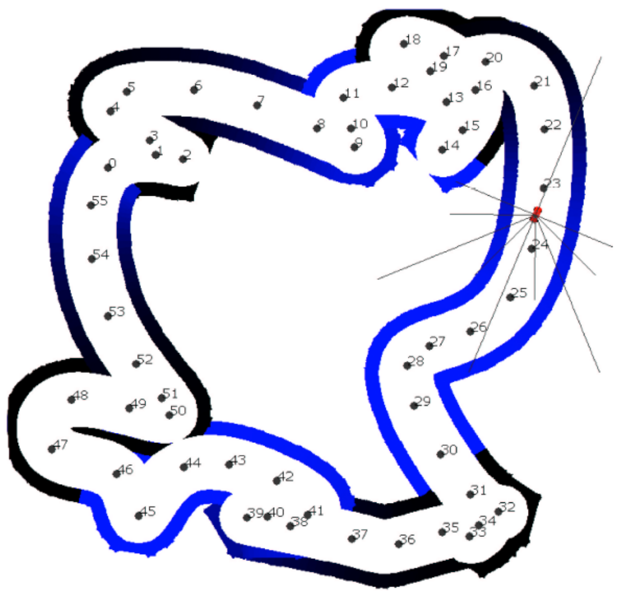


Fig. 2. An example of track evolved by Togelius et al. [6]. The image appears as Figure 5 in the original paper.

believable when placed in the realistic environment of the simulator. Thus, tracks must be (obviously) closed and their curvature radius is typically constrained both in range and in the way it can change. Finally, TORCS introduces an additional representation constraint since the curvature radius of turns is piece-wise constant in that curves are defined as sequences of subsections with a constant curvature radius. As a consequence, not all the closed b-splines employed in [6] represent feasible TORCS tracks and therefore the radial representation of Togelius et al [6] cannot directly applied in our case.

Our track evaluation focuses on the evolution of tracks with a large degree of diversity rather than trying to fit a target player profile as done in [6]. More precisely, we view *track diversity* as the main source of challenge and interest for a racing games and eventually as one of the several features enabling players' fun. We focus on two metrics to capture diversity in terms of (i) the variety of turns and straights in the track and (ii) the range of driving speed achievable along the track.

IV. TORCS

The Open Racing Car Simulator (TORCS) [4] is a state-of-the-art open source car racing simulator which provides a sophisticated physics engine, full 3D visualization, several tracks, car models, and game modes (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 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 gas/brake pedals, the gear stick, and steering wheel.

All the experiments reported in this paper have been carried out with TORCS 1.3.1.

Fig. 3. A screenshot of the game TORCS.

V. Track Representation

The representation of game content is a central issue in Search-Based Procedural Content Generation [3]. In this section, we briefly describe the TORCS representation of tracks, i.e., the phenotype, the indirect encoding we designed, i.e., the genotype, the genetic operators, and the mapping between genotypes and phenotypes.

A. *Track Representation in TORCS (the phenotype)*

In TORCS, a track is represented as an ordered list of segments. Each segment is either a straight or a turn. A straight is defined by just one parameter, its length. A turn is defined by (i) the direction (i.e., left or right); (ii) the arc it covers measured in radians; (iii) its start radius and (iv) its end radius. In addition, the track must be feasible, i.e., it must be *closed*, and therefore the last segment must overlap the first segment.

B. Track Encoding (the genotype)

The direct encoding of the track representation in TORCS into a genotype is infeasible since it produces a huge search space (thus leading to the curse of dimensionality) in which the feasible solutions (i.e., the closed tracks) are only a tiny proportion. Accordingly, we employed an *indirect encoding* inspired by the work of Togelius et al. [6] on a simple 2D car racing simulator (see Figure 1). In [6], a track is represented as a set of *control points* that the track has to cover; the track is generated as a sequence of Bezier curves connecting three control points and, to guarantee smoothness, have the same first and second derivatives at the point they join.

In this work, we encoded a track as a sequence of control points $\vec{p} = \{p_1, \dots p_n\}$, where p_i consists of three parameters r_i, θ_i, and s_i; the parameters r_i and θ_i identify the position of the control point p_i in a polar coordinate system (r_i is the distance from the origin or *radial coordinate*, θ_i is the *angular coordinate*); s_i controls the slope of the track tangent line in p_i. Figure 4 shows an example of our encoding: control points are depicted in red; the blue dot represents the origin of the polar coordinate system; the curves represent what generated by the genotype to phenotype mapping process, discussed in the next section.

C. Genotype to Phenotype Mapping

Algorithm 1 reports the pseudo code of the GENERATE-TRACK procedure that maps our encoding into the track representation used in TORCS. The procedure takes as input the n control points $\vec{p}$ and returns a list $\vec{t}$ of track segments in TORCS format. Initially, the polar coordinates of the n control points (i.e., the r_i and θ_i values) are used to compute the ranges of feasible slope values for each control point (line 2). If there exist a control point for which there are no feasible slope values (thus it is not possible to join the incoming and the outgoing segments in that point), no track can be derived from $\vec{p}$ and therefore no track (a NULL track) is returned. Otherwise, given the ranges of feasible slope values, the actual slope values are computed on the basis of the s_i values of the control points $\vec{p}$ (line 6). Next, for each pair of control points, p_i and p_{i+1}, the corresponding TORCS segment is generated using both the position and the slope values as constraints. Then, the procedure tries to close the track by generating one or more segments to connect t_{n-1} to t_1 (line 9). If the process succeeded (line 10), the resulting track $\vec{t}$ is returned otherwise a NULL track is returned.

D. Closing the Track

In general, it is not always possible to connect the last and first control points with just one single segment and a sequence of straights and turns might be required. Accordingly, the procedure CLOSETRACK (line 10) considers the parameters defining the end and starting control points p_n and p_1 and try different heuristics to generate a feasible closed track.

Algorithm 1 Generate the track from the genotype

```
1:  procedure GENERATETRACK($\vec{p}$)
                              ▷ $\vec{p}$: array of $n$ control points $\langle r_i, \theta_i, s_i \rangle$
                              ▷ $\vec{t}$: array of TORCS segments $\langle t_1, \dots t_n \rangle$
2:      $\{I_i\}$ = GENERATESLOPERANGES($\vec{p}$)
                                                      ▷ No feasible slope range
3:      if $\exists i | I_i = \emptyset$ then
4:          return NULL
5:      end if
                                                      ▷ Select slope values
6:      $\vec{\alpha}$ = SELECTSLOPES($\vec{p}$, $\{I_i\}$)
                                          ▷ Generate the first $n-1$ segments
7:      for $i = 1$ to $n-1$ do
              $t_i$ = GENERATESEGMENT($p_i$, $p_{i+1}$, $\alpha_i$)
8:      end for
                                                      ▷ Close the track
9:      $t_n$ = CLOSETRACK($t_{n-1}$, $t_1$)
                                  ▷ Check whether the procedure failed
10:     if CLOSEDTRACK($\vec{t}$) then
11:         return $\vec{t}$
12:     else
13:         return NULL
14:     end if
15: end procedure
```

```
1:  procedure GENERATESEGMENT($p_i$, $p_j$, $\alpha$)
                                          ▷ Compute the slope of $\overline{p_i p_j}$
2:      $\alpha^*$ = SLOPE($p_i$,$p_j$)
3:      if $|\alpha - \alpha^*| < \epsilon$ then
              return GenerateStraightSegment($p_i$, $p_j$)
4:      else if $\alpha < \alpha^*$ then
              return GenerateLeftTurn($p_i$, $p_j$, $\alpha$)
5:      else if $\alpha > \alpha^*$ then
6:          return GenerateRightTurn($p_i$, $p_j$, $\alpha$)
7:      end if
8:  end procedure
```

E. Genetic Operators

We applied the standard *one-point crossover* and the rather typical *polynomial mutation* operator used in real-valued genetic algorithms. It is worth noting that, while in the *radial representation* in [6] only the radius was mutated in our case all both the radial and angular position of the gene can be recombined and mutated.

VI. TRACK EVALUATION

Our approach focuses on the evolution of racing tracks that can provide both an adequate amount of challenge (as in [6]), and can also provide a large degree of diversity, in a rather broad sense. Accordingly, while [6] focuses on the

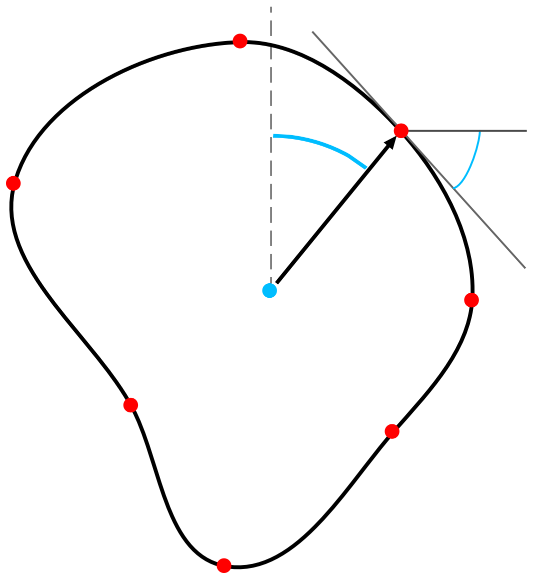


Fig. 4. Track encoding used by the evolutionary process: the red dots identify the control points; the blue dot is the origin of the coordinate system; an example of a feasible slope is also shown. The curves are the result of the genotype to phenotype mapping process.

fitting of a target player profile, in our case, we focused on the maximization of *the track diversity*. In particular, we can identify four ways of defining diversity in a racing game. The number, the distribution, and the types of turns in the track, i.e., the *curvature profile*, is one of the most evident source of diversity. The range and the distribution of the achievable driving speeds over the track, i.e., the *speed profile*, is another major source of diversity; in this respect, it is worth noting that, two tracks with the same curvature profile (i.e. containing the same number and types of turns and straights) can have very different shapes and therefore very different achievable speeds. Also the variety and the details of the roadbed (e.g., the presence of bumps, the percentage of gravel and asphalt) might be considered as a source of diversity. Finally, the surrounding scenery (e.g., landscape, trees, etc.) is an additional source of diversity.

In this work, we focused on diversity in terms of shape and in terms of achievable driving speed and present an approach to evolve racing tracks that maximize the diversity in terms of curvature and speed profiles. For this purpose, we evaluate racing tracks based on the entropy of their curvature and speed distributions and apply single-objective and multi-objective real-coded genetic algorithms to evolve tracks which maximize either one of the two criteria or both at the same time.

A. Evaluation Based on Curvature Profiles

As the first measure of track diversity, we focused on the shape and more precisely on the number and types of track segments. Accordingly, we defined the diversity of a track as the entropy of its *curvature profile* C, that is, the distribution of curvature values of all its track segments. For this purpose, we initially estimated the range of feasible curvature values by analyzing all the segments of all the human-designed tracks available on-line (i.e., those available in the TORCS distribution and the ones created by the users). Then, we partitioned the curvature range into sixteen bins $(b_1, \ldots, b_{16})$.

Given a new track $\vec{t}$, we define its curvature diversity as follows. Firstly, for each track segment t_i we compute its length and its curvature as $1/\hat{r}_i$, where $\hat{r}_i$ is defined as (i) the radius for a right turn, (ii) the opposite of the radius for a left turn and (iii) it is considered infinite for a straight. Next, we generate the track curvature profile $C = \{c_1, \ldots, c_{16}\}$, using the sixteen bins computed from the human-designed tracks; c_i is the percentage of the track with a curvature that belongs to bin i. Finally, we compute the entropy of its curvature profile $H(C)$ as,

$$H(C) = -\sum_{i=1}^{16} c_i \log_2 c_i \qquad (1)$$

so that $H(C) \geq 0$ and $H(C) \leq \log_2 16$. The entropy $H(C)$ measures the diversity in the distribution of curvature values in the track: it is maximum when all the curvature values occupy the same percentage of the track that is $c_i = 1/16$ for all the i; it is minimum when all the curvature values belong to the same bin, that is, there is an i such that c_i is one. Note that, we determined the number of bins (16) empirically as the best trade-off. In fact, an analysis we performed showed that a small number of bins (e.g., 2 or 4) would tend to produce too simple profiles which could not capture many of the difference between similar tracks. On the other hand, a larger number of bins (e.g., 32 or 64) would tend to produce too fine-grained profiles with several empty.

As an example, Table I reports the curvature profiles of four tracks available in the TORCS distribution, listed in increasing entropy value. The first track, *D-Speedway*, has an oval shape with a long straight and two long turns with large curvature radius r; accordingly, all the track segments belong to the central bin corresponding to near zero curvatures. In fact, since the curvature of a turn is computed as $1/r$ and the curvature of a straight is zero, both straights and large-radius bends have near zero values. As the number and model of different track segments increases, the track diversity increases too so that the curvature profile tends to include other bins corresponding to values quite far from zero (see for instance, the track *Aalborg*). The last track, *Spring*, provides the greatest diversity with a good variety of bends; as a consequence, the majority of the sixteen bins are covered by some track segments.

B. Evaluation Based on Speed Profiles

The second measure of track diversity we considered is based on the distribution of the achievable speeds. The underlying idea is that tracks are different not only because they look different (they have different types and distribution of turns and straights) but also because they allow for a wide variety of speeds. Therefore, we defined the track diversity in terms of achievable of speed as the entropy of its *speed profile* which we computed as follows.

TABLE I

Curvature profiles in four of the tracks available with TORCS.

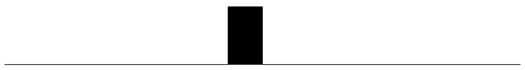
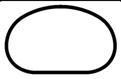
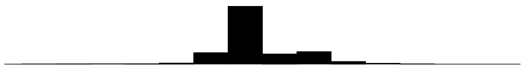
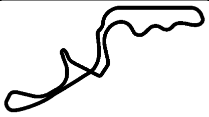
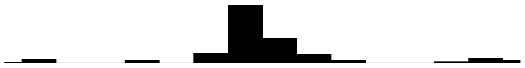
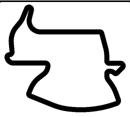
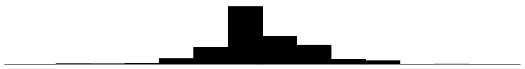
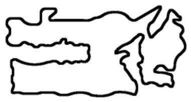
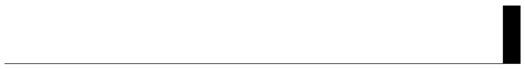
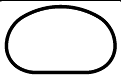
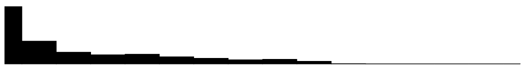
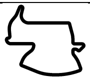
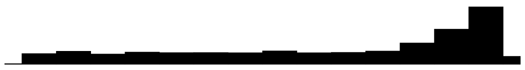
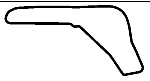
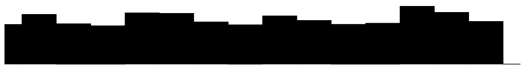
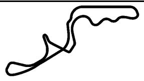
Track Name	**Profile**	Shape
D-Speedway		
Wheel-2		
Aalborg		
Spring		

TABLE II

Speed Profiles in some tracks of the TORCS package

Track Name	**Profile**	Shape
D-Speedway		
Aalborg		
Forza		
Wheel-2		

As the very first step, we let several TORCS bots drive in all the available (human-designed) tracks for several laps and collected the car speed during each game tick; this produced the distribution of the feasible speed values over all the human-designed tracks. Then, we partitioned the range of feasible speeds into sixteen bins ($b_1, \ldots, b_{16}$), as we did for curvature profiles.

Given a new track $\vec{t}$, we measure its speed diversity as the entropy of its speed profile, which we compute as follows. Firstly, we have a bot completing three laps on the track $\vec{t}$ and, during the second lap, we measure the car speed for each game tick. Next, we generate the speed profile $S = \{s_1, \ldots, s_{16}\}$, using the previously generated bins, where s_i is the percentage of game ticks, in the second lap, that the car raced with a speed corresponding to the bin i. Finally, we compute the entropy of the track speed profile as,

$$H(S) = -\sum_{i=1}^{16} s_i \log_2 s_i \qquad (2)$$

As before, $H(S) \geq 0$ and $H(S) \leq \log_2 16$. The entropy $H(S)$ measures the diversity in the distribution of speed values over the track: it is maximum when the car spent the same amount of time in all the speed ranges $s_i = 1/16$ for all the i; it is minimum when the car spent most of the time in the same speed range, that is, there is an i such that s_i is one.

Table II reports the speed profiles of four tracks available in TORCS, listed in increasing entropy value. The oval track *D-Speedway*, which did not provide much diversity in terms of curvature profile, does not provide much diversity also in terms of speed. Its long bends and long straights allows the driver to keep basically the same high speed range all the time. In fact, all the recorded speed values fall into the last bin. The second track, *Aalborg* has several narrow bends that result in a speed profile skewed toward the lower speeds. In contrast, *Forza* has several fast stretches and few bends of rather different curvature accordingly its speed profile is skewed towards higher speeds; on the other hand, the rather different curvatures of the few bends in *Forza* allows the car to cover almost all the possible speed ranges. The fourth track, *Wheel-2*, is one the track providing the highest degree of diversity among all the one available in TORCS as demonstrated by the well-balanced and almost uniformly distributed speed profile.

C. Discussion

Wheel-2 and *Aalborg* are good examples of how the two diversity measures we introduced actually estimates two very different ideas of diversity. When the topology is concerned, although *Wheel-2* looks more articulated than *Aalborg*, it turns out that *Wheel-2* provides *less* diversity than *Aalborg* since most of its turns and bends have similar curvatures. In contrast, *Aalborg* has several long stretches but also many turns of very different radius which provide more diversity than *Wheel-2*. On the other hand, when speed is concerned, *Wheel-2* provides *more* diversity than *Aalborg* because its balanced combination of many soft turns and few tight turns, which allows the drivers to cover basically all the possible speed ranges. In contrast, the many different tight turns of *Aalborg* restrict the driver towards the lower scale of the speed range. Even when the car is driving along the stretches, the car speed cannot increase too much because of the incoming very tight turns. In fact, the two measures we introduced are only slightly overlapping since there are very few tracks with qualitatively similar curvature and speed profiles (e.g., *D-Speedway* or other ovals). Accordingly, in this work we investigated both the application of (i) single-objective evolution to maximize the diversity for each one of the two criteria (Section VII-A and Section VII-B) and (ii) multi-objective evolution to search for a good trade-off between the two (Section VII-C).

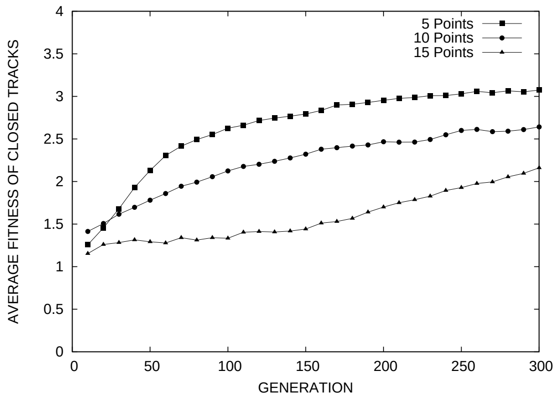
Finally, we wish to point out that, although the entropy of curvature profiles provides a more intuitive idea of track diversity (that it is related to a visual property) and it simpler to compute (since no racing is involved), it actually poses much tighter constraints than the entropy of speed profiles. In fact, the entropy of a curvature profile is maximal when the track contains *the same percentage of turns with all the feasible curvature values*. In contrast, the entropy of a track speed profile is maximal when the driver has to cover all the possible speed ranges for the same amount of racing time. Accordingly, while high-entropy curvature profiles correspond to rather complex (and difficult to evolve) track topologies, high-entropy speed profiles are possible even in simpler (and easier to evolve) topologies.

VII. Experimental Results

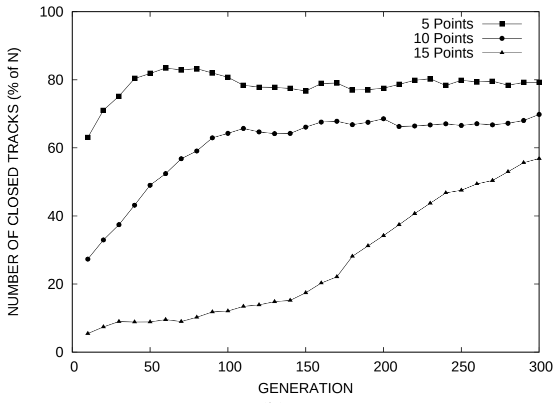
We tested our approach by applying single-objective and multi-objective real-coded genetic algorithms to evolve tracks with a large degree of diversity both in terms of curvature profile (i.e., the number and types of turns and straights) and in terms of speed profile. For this purpose, we performed three sets of experiments respectively focused on the maximization of (i) the entropy of curvature profiles (i.e., of the shape diversity), of (ii) the entropy of speed profiles (i.e., of the speed diversity) and (iii) the search of a good trade-off between these two measures. All the experiments were performed using the implementations of the single and multi-objective genetic algorithms available in Sastry's genetic algorithm toolbox [22] while for the evaluation of tracks was performed using TORCS 1.3.1.

A. Maximizing the Entropy of Curvature Profiles

In the first set of experiments, we applied a single-objective real-coded genetic algorithm in which individuals are tracks encoded using 5, 10, or 15 control points (Section V) and the fitness function is computed as the entropy of the track curvature profile (Section VI); individuals corresponding to infeasible (open) tracks receive a zero fitness; in addition, since the track generator of TORCS cannot deal with track intersections, we penalized the fitness of individuals by 0.5 for each intersection. The parameters of the genetic algorithm were set as follows [22]: the mutation



(a)



(b)

Fig. 5. Single-objective genetic algorithm using the entropy of curvature profiles as fitness function and a track encoding involving 5, 10 and 15 control points: (a) average fitness of the closed tracks in the population; (b) percentage of individuals that represent a feasible (closed) track. Curves are averages over ten runs.

probability was 0.1 while the crossover probability was 0.9; the population size was set to 50, 100, and 150 individuals for tracks represented using 5, 10, and 15 control points, respectively; selection is performed using tournament selection with a tournament size of 2; the process ended as soon as 300 generations were completed.

Figure 5 reports the average population fitness (Figure 5a) and the percentage of individuals in the population which represent feasible (closed) tracks (Figure 5b). As can be noted, the higher the number of control points used to represent the tracks is, the more complex the search space becomes. In particular, as the number of control points increases, the evolution of high entropy individuals becomes more difficult: when tracks are represented using 10 or 15 control points, the average fitness of the population grows more slowly and reaches a smaller entropy value when the process ends (Figure 5a). Similarly, as the number of control points increases, it becomes more and more difficult to generate valid tracks and the percentage of valid (closed) tracks in the population grows much slower than fewer points (Figure 5b). It is important to stress that the genetic algorithm can always increase the number of closed tracks consistently as the generations proceed. For instance, when individuals encode 15 control points, only the 5% of the initial randomly generated population consists of valid (closed) tracks; however, after 300 generation, slightly more than the 56% of the individuals in the final population represent valid tracks.

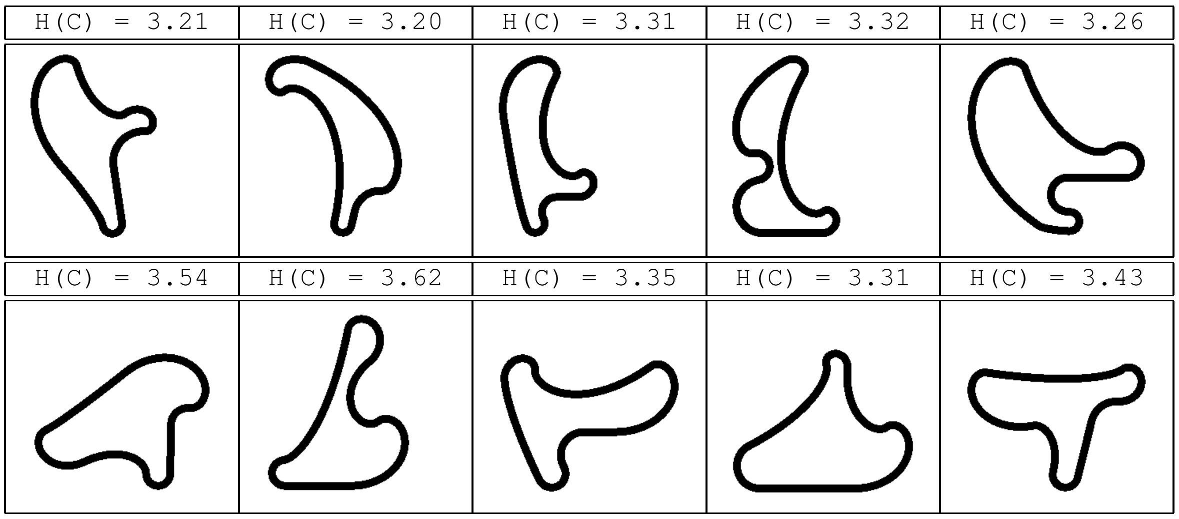
The results for the tracks encoded using 15 control points suggest that the problem of generating valid tracks is not trivial and in fact, a random population contains quite a small percentage of valid tracks. On the other hand, the evolutionary approach we propose appears to be rather successful in that it can actually produce a significant increase both in terms of diversity, as the average fitness reaches a near-optimal value (Figure 5a), and in terms of number of valid tracks evolved, as their number reaches a tenfold increase (Figure 5a).

Figure 6 reports the best track evolved for each of the ten runs of the genetic algorithm when 5 control points (Figure 6a), 10 control points (Figure 6b), and 15 control points (Figure 6c) are used. As should be expected, as the number of control points increases, also the complexity of the evolved tracks increases, making the maximization of the entropy of curvature profiles more difficult. In fact, the best tracks evolved using 5 control points (Figure 6a) have a slightly higher fitness than the best tracks evolved using more control points. From the one hand, this result suggests that 5 control points are enough to evolve tracks which (i) include a quite large range of different turns (in terms of radius) and (ii) are also well-balanced, in that they provide a good mix of different types of turns and straights. On the other hand, with more than 5 control points, it is still possible to evolve tracks featuring a wide range of different turns and straights, but it might be difficult to obtain well-balanced tracks.

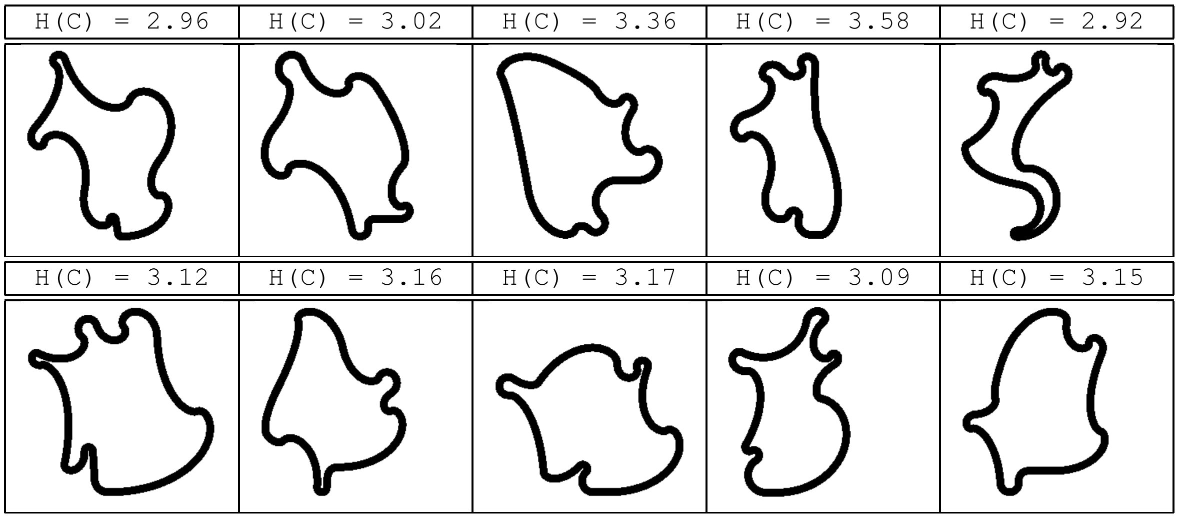
Finally, it is worth noticing that not all the tracks include straights. This is not surprising and depends on the way we compute the curvature profile for a track. In fact, according to our definition of curvature profile, both straights and turns with very large radius of curvature are basically identical in terms of profile since they belong to the same discretization bin. As a consequence, evolution will tend to evolve tracks featuring turns with with small curvature turns (or with very large radius of curvature) instead of simple straights.

B. Maximizing the Entropy of Speed Profiles

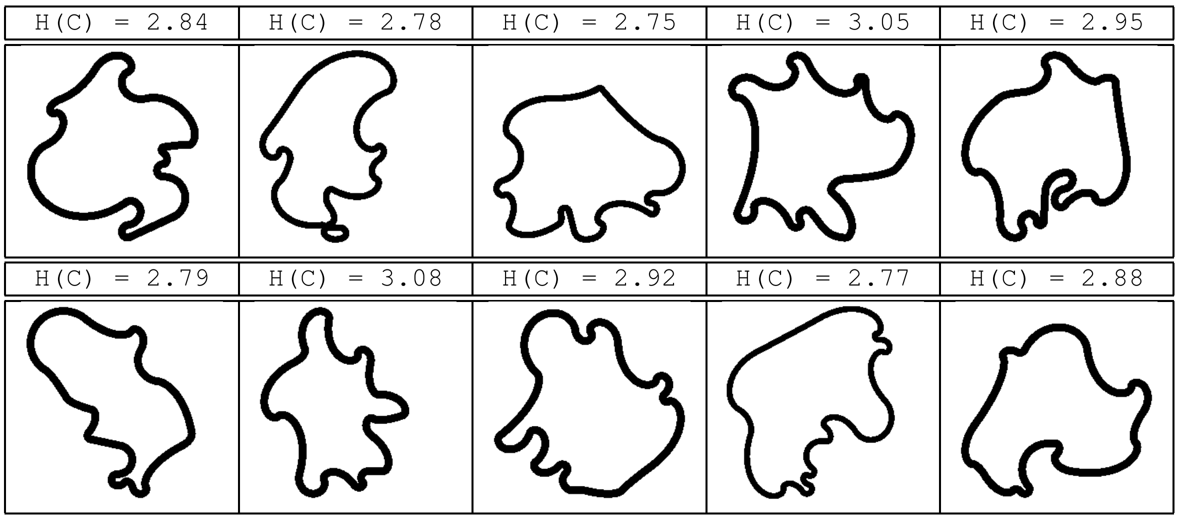
We repeated the same set of experiments, using the same parameter settings, while computing the fitness of a track as the entropy of its speed profile. Also in this case, we penalized the fitness of individuals by 0.5 for each intersection. Figure 7 reports the average population fitness (Figure 7a) and the percentage of feasible (closed) tracks in the population (Figure 7b) as a function of the number of generations. As in the previous set of experiments involving curvature profiles, a higher number of control points results in a slower convergence to high entropy individuals (Figure 7a). In contrast to what happened with the curvature-based fitness, in this case, all the average population fitnesses converge almost to the same near-optimal value. This result can be easily explained by considering that, as we previously noted



(a)

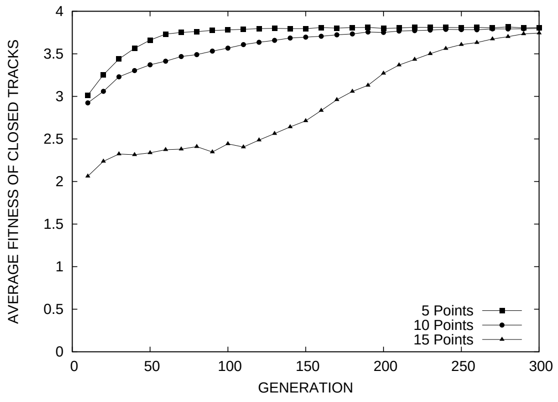


(b)

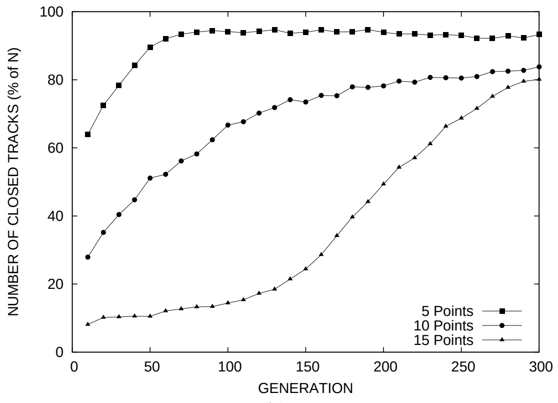


(c)

Fig. 6. The best tracks evolved for each run using the entropy of curvature profiles when the tracks are encoded using (a) 5 control points, (b) 10 control points, and (c) 15 control points; for each track is also reported `H(C)`, the entropy of the curvature profiles.



(a)



(b)

Fig. 7. Single-objective genetic algorithm using the entropy of speed profiles as fitness function and a track encoding involving 5, 10 and 15 control points: (a) average fitness of the closed tracks in the population; (b) percentage of individuals that represent a feasible (closed) track. Curves are averages over ten runs.

in Section VI, it is generally easier to evolve tracks with high entropy speed profiles than it is for curvature profiles because of the tighter constraints they pose.

The plot for the percentage of closed tracks in the population (Figure 7b) also confirms our previous findings: as the number of control points used to represent the track increases, the evolution of valid closed tracks becomes more difficult. In fact, a higher number of control points corresponds to an initial random population mainly made of infeasible (open) tracks: with 10 control points only around the 30% of the random tracks are closed, with 15 control points only around the 5% of the random tracks are closed. Also in this case however, the evolutionary process can successfully get rid of infeasible individuals so as to converge to a population containing mainly valid tracks with high entropy values. In particular, it is worth noticing that the percentage of valid tracks in the final population evolved using speed profiles (Figure 7b) is much higher than the one obtained using curvature profiles (Figure 5b).

Figure 8 reports the best tracks evolved during each one of the ten runs involving 5 control points (Figure 8a), 10 control points (Figure 8b), and 15 control points (Figure 8c). Figure 8 confirms our previous findings in that 5 control points seem enough to evolve tracks with a large degree of diversity which, in this case, means tracks involving a wide range of turns and straights and uniformly distributed variety of speed values over the lap. In this case however, in contrast to what happens for curvature profiles, the tracks evolved using 10 and 15 control points achieve almost the same high entropy values as the tracks evolved using only 5 control points.

C. Searching for a Trade-off Using Multi-Objective GAs

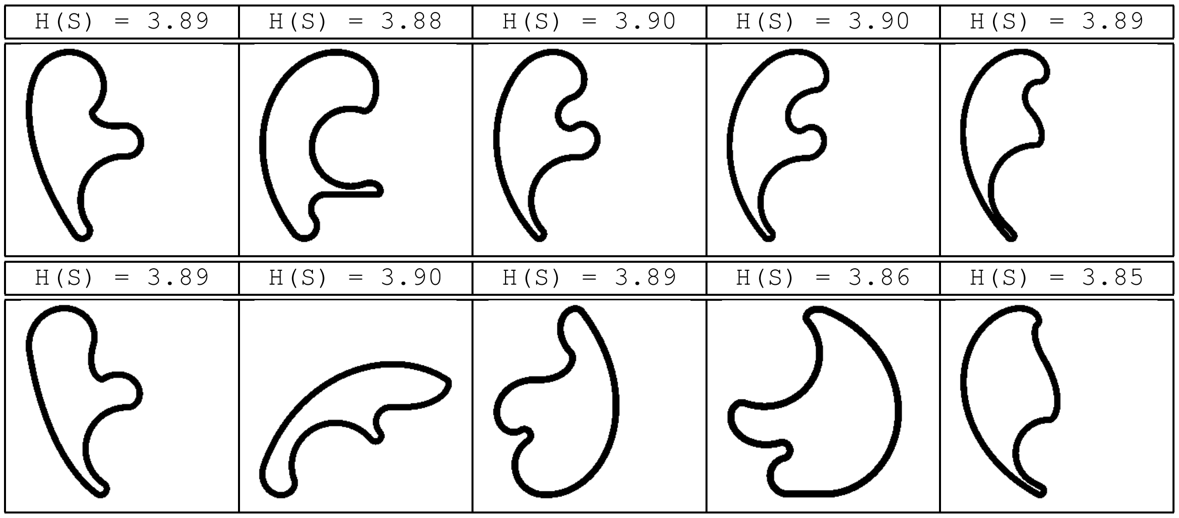
The results for the single-objective genetic algorithm show that the fitness based on curvature profiles produces tracks with a rather broad range of turns, from hairpin to very large and fast bends, while the fitness based on speed profiles produces tracks with a good balance of fast and slow stretches, leading to a rich and challenging driving experience. Early on, in Section VI, we argued that these two fitness definitions capture rather different ideas of diversity. We tested our hypothesis empirically by analyzing the correlation between the entropy of curvature profiles and the entropy of speed profiles using a set of 1000 tracks evolved using the same single-objective genetic algorithms used in the previous experiments. Our analysis returned a correlation coefficient of 0.56 suggesting that the two fitness definitions *are not highly correlated* and that a track with a high entropy curvature profile does not necessarily have a high entropy speed profile. Accordingly, we applied a multi-objective genetic algorithm to evolve tracks, encoded with 5, 10, and 15 control points, which could maximize the two objective at the same time. For this purpose, we used the same parameters settings used in the previous experiments; offspring selection was performed using the Non-Dominated Sorting Genetic Algorithm (NSGA-II) implementation available in Sastry's genetic algorithm toolbox [23], [22].

Figure 9 reports, for each encoding, three representatives of the Pareto front evolved after 300 generations and the overall Pareto front. All the results agree in that the three Pareto fronts (Figure 9d-5, d-10 and d-15) clearly show that there is a conflict between the two objectives, i.e., the maximization of diversity in terms of curvature and in terms of speed profiles. Nevertheless, the evolved Pareto fronts contains very different tracks models providing both a very good trade-off between the two competing objectives with a nice balance between the variety of turns and a rich driving experience.

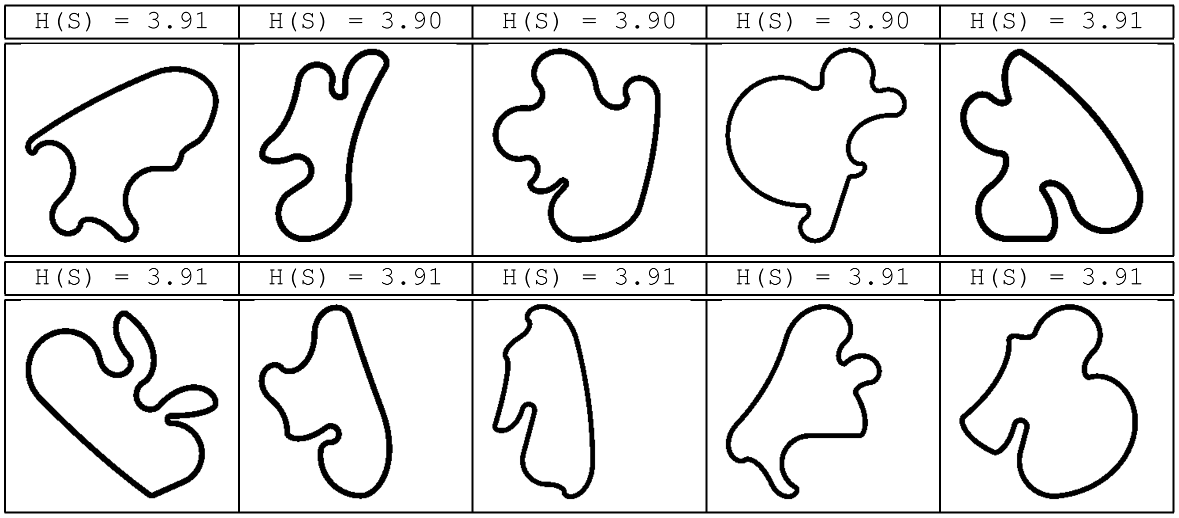
Interestingly, the best solutions evolved by the multi-objective genetic algorithm have very high (sometime near-optimal) entropy values both in terms of curvature and speed profiles suggesting that multi-objective evolution is probably the best approach to tackle this application.

VIII. Validation

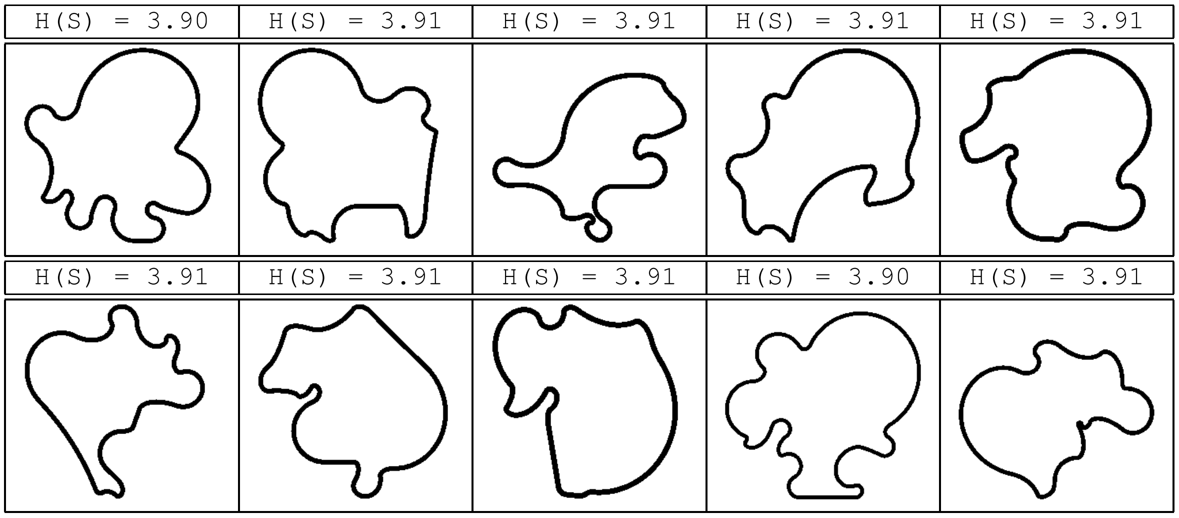
We performed a preliminary validation of our approach on human subjects. Our aim was to test whether an agreement exists between our fitness definitions and users preferences. For this purpose, we designed two surveys, one focused on



(a)



(b)



(c)

Fig. 8. The best tracks evolved for each run using the entropy of speed profiles when the tracks are encoded using (a) 5 control points, (b) 10 control points, and (c) 15 control points; for each track is also reported H(S), the entropy of the speed profiles.

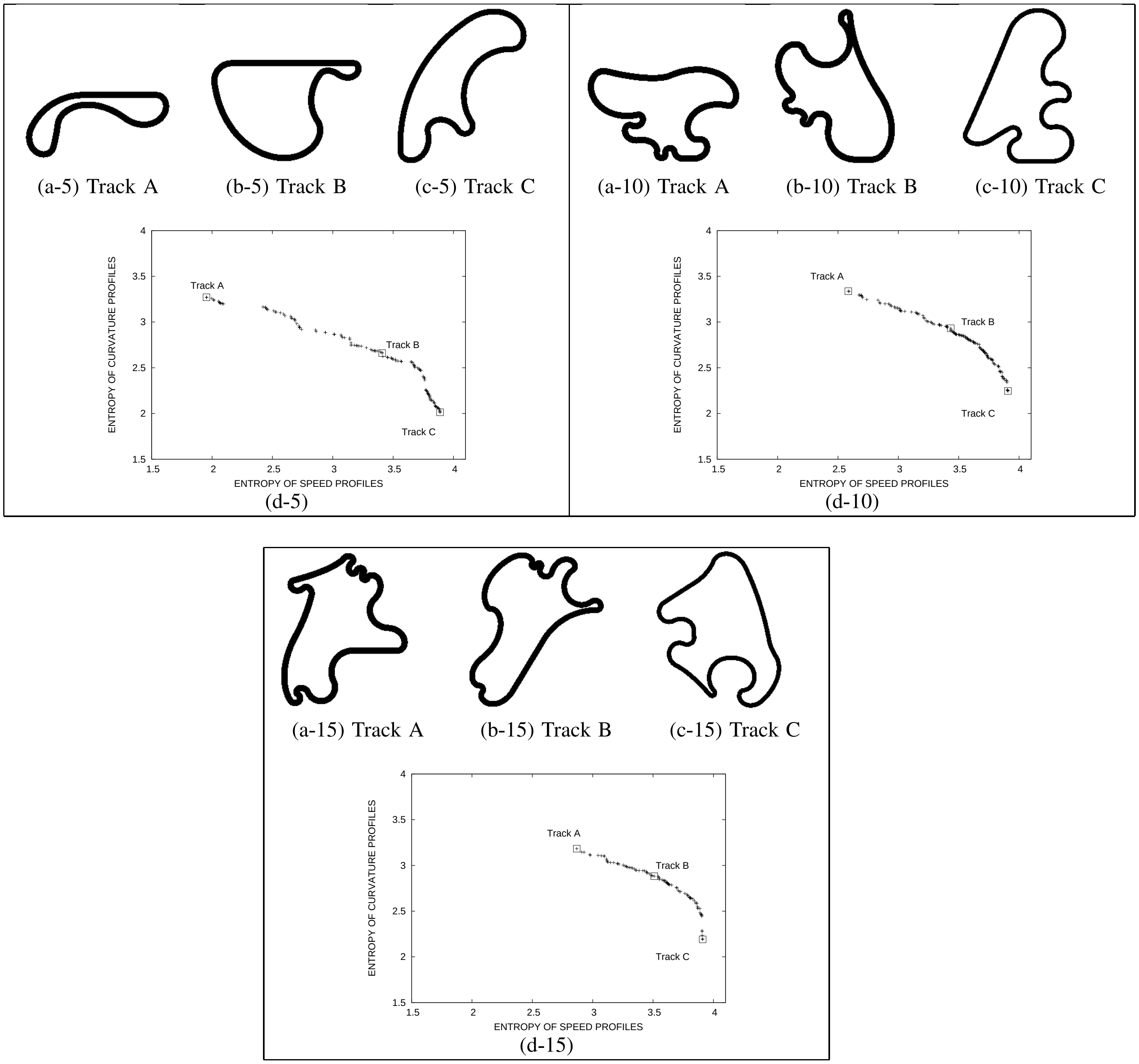


Fig. 9. Multi-objective genetic algorithms applied to evolve tracks encoded using 5, 10, and 15 control points; to represent the tracks: Track A, Track B, and Track C are examples of track evolved for different sections of the pareto; (d-5), (d-10) and (d-15) are examples of final pareto front evolved after 300 generations.

the visual properties of evolved tracks, one focused on the actual playing experience.

A. *Design of the Surveys*

The two surveys were completely anonymous, the only personal information required from the subjects was (i) their age and (ii) how often they play racing games (choosing from *often*, *occasionally*, or *never*).

The first survey focused on the subjects visual preferences. It consisted in a paper form with 20 black and white pictures of tracks, evolved based on their curvature profile; the pictures were organized in pairs and they were similar to the ones used in this paper. Of the ten pairs, (i) two compared tracks evolved using 5 control points, (ii) two compared tracks evolved using 10 control points, (iii) two compared tracks evolved using 15 control points, while (iv) four compared tracks evolved using a different number of control points. For each one of the ten pairs printed on the paper form, the subjects had to choose the track they liked best; the subjects could also select none of the tracks; for each pair a small area was also provided so that people could leave a comment if they wished. This survey basically aimed at testing the visual users preferences and roughly simulated the race setup in TORCS where the user selects the track based only on its plain shape.

The second survey focused on the player experience and involved 42 human subjects. For each subject, the survey involved three playing sessions. The first session consisted of a *warm-up* session where the subject can experience the control system of TORCS driving in a simple track provided with the standard distribution of TORCS. Then, in the following two playing sessions we asked the subjects to complete two laps on two different tracks evolved by the multi-objective genetic algorithm. At the end of the playing sessions, subjects had to answer to the following questions on the two tracks tested: (i) which one they *preferred*, (ii) which one they found more challenging, and (iii) on which one they would like to race again.

Note that, since Survey A focuses on the visual characteristics of tracks, its collected data can only be analyzed in terms of curvature profiles; in contrast, since Survey B focuses on the actual playing experience, its collected data can only be analyzed in terms of speed profiles.

B. Analysis of the Visual Preferences

The first survey involved 59 human subjects randomly selected from the students attending the course of Videogame Design and Programming[20] at the Politecnico di Milano.

We analyzed the collected data to test the agreement between the subjects preferences and the track fitness used to evolve the tracks (i.e., the entropy of curvature profiles). In particular, to avoid any possible bias, we initially considered only pairs in which both tracks were encoded with the same number of control points. The analysis shows that of all the 349 collected preferences, slightly more than the 60% (or 210) agreed with the fitness measures, i.e., the entropy of the track curvature profile. We tested whether this result is statistically significant by applying the same procedure used in several previous works [24], [25]. More precisely, we computed the p-value as the probability according to a binomial distribution (with p=0.5) of obtaining the same number of successful outcomes or more over the given number of trials. In this case, the probability of obtaining 210 or more successful outcomes over 349 trials is $5.5 \cdot 10^{-5}$, indicating that there is a statistically significant agreement between the fitness and the subjects preferences.

We repeated the analysis on the three data sets containing only one type of encoding (the one using 5 control points, the one using 10 control points, and the one using 15 control points). The results show that the agreement between the subjects' preferences and the fitness diminishes as the number of control points increases. When we consider only the tracks encoded using 5 control points, the agreement is around the 69.8% (i.e., 81 over 115 cases agree with the fitness); the agreement drops to the 56.4% with 10 control points (i.e., 66 cases over 117 agree) and to the 54.3% with 15 control points (i.e., 63 cases over 116 agree).

We also analyzed the relation between the subjects' preferences and their experience with racing games. The results show that the agreement between the subjects' preferences and the curvature-based fitness increases with the subjects' experience. In subjects with an extremely poor experience with racing games, only the 52% of the preferences agrees with the fitness and the correlation is not statistically significant (p-value>0.05). However, when subjects who declared to play racing games often, we have that the agreement reaches the 64.4% and, when we restrict the analysis to tracks encoded with 15 control points, the agreement reaches the 73.3%. Noticeably, both these results are statistically significant with a p-value equal to 0.018. Overall, these results suggest that the curvature-based fitness we proposed might not be the best way to capture the preferences of inexperienced players.

C. Analysis of the Playing Preferences

The second survey involved 42 human subjects, randomly selected from the visitor of the Politecnico's open-university day, who raced on two different tracks, randomly selected among two different sets. The first set of tracks contained six tracks randomly selected from the Pareto fronts evolved in the experiments discussed in Section VII-C using a multi-objective algorithm. The second set contained other six *control* tracks, with lower fitness values (both in terms of speed and curvature profiles), that were randomly selected from the early populations. To avoid any bias related to the playing order, during each test, a script randomly selected whether the experiment would start with a track from the Pareto front or with one from the control group. After a subject completed two laps, on each track, the subject was asked (i) which track he/she found more challenging, (ii) which track he/she would race again, and (iii) which racing experience he/she preferred; in addition, subjects were also allowed to leave a short comment to motivate their choices.

Our data show that the evolved tracks are the most challenging basically for all the human subjects (92.86%); when asked which track they would play again, 30 out of 42 human subjects (71.43%) prefer the evolved tracks from the Pareto front. However, when evaluating their actual racing experience on the two tracks, solely 22 subjects over 42 (52.38%) prefer the evolved tracks. The comments left by the subjects suggest that the evolved tracks might be too challenging for inexperienced players who generally preferred racing on the simpler control track. This bias results in a rather low 52.38% preference toward the tracks from the Pareto front. It should be noted however that, notwithstanding their low performance on the evolved tracks and the higher perceived challenge, several of such inexperienced players still wish to rerace the more difficult evolved track. This suggests that for these subjects the evolved tracks were challenging but not frustrating and thus they were encouraged and willing to try them again.

[20]http://games.ws.dei.polimi.it

To rule out the bias due to inexperience we repeated the same analysis on the the data collected from the most experienced subjects. When we consider only the 14 subjects (33%) with more experience in racing games, we note that, (i) all the subjects find the evolved tracks more challenging, (ii) most of them (12 out of 14 or 85.71%) would choose the evolved track to play again, and (iii) the evolved tracks are preferred by 10 subjects out of 14 (71.43%). These results suggest that, although overall preferred by only 52.38% of the subjects, the evolved tracks are very appealing to subjects with some experience in racing games. In fact, 71.43% of the subjects prefer them and 85.71% of the subjects wish to replay them.

Finally, we applied the same statistical procedure used in the previous survey. The analysis shows that all the results are statistically significant (p-value$<$0.05) except for the one showing that 52.38% of subjects prefer the evolved track.

IX. Conclusions and Future Research Directions

We introduced an evolutionary approach for the automatic generation of tracks for a high-end open-source car racing simulator (TORCS) [4]. Our approach is inspired by the seminal works that Togelius et al. [5], [6], [7] did on the evolution of racing tracks for a simple 2D arcade racing game which, to the best of our knowledge, is also the only published work on the evolution of racing tracks. In particular, our track encoding is an extension of the *radial representation* proposed in [6]. As in [6], we use an indirect encoding and represent a track as a set of control points in a polar coordinate systems; however, while in [6] only radial coordinates are evolved, in our case, evolution works both on the radial coordinates and on the angular coordinates; in addition, our representation also includes, for each control point, the slopes of the corresponding track tangent line.

One of the major difference with respect to the work of [5], [6], [7] is in the way we evaluate racing tracks which, in our case, is based on the *maximization of the track diversity* both in terms of shape (e.g., the types and number of turns and straights) and in terms of achievable driving speed (as opposed to the fitting of a target player profile [5], [6], [7]). In particular, for a given track, we derive its curvature profile and its speed profile; then, we compute the track fitness using either the entropy of its curvature profile or the entropy of its speed profile.

We performed three sets of experiments involving single-objective and multi-objective genetic algorithms to evolve tracks encoded using 5, 10, and 15 control points. Our results show that both single-objective and multi-objective approaches generate a wide variety of feasible TORCS tracks with a high degree of diversity both in terms of shape and in terms of achievable speeds. In particular, multi-objective evolution can successfully evolve rich Pareto fronts containing large numbers of feasible tracks which provide a good trade-off between the two competing objectives (i.e., having a rich structure while also providing an articulated driving experience in terms of achievable speed).

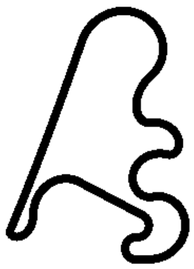
It is worth examining computational cost of the overall process. In our framework, one run of most demanding configuration (consisting of 300 generations using a population of 150 individuals) using *only one core* of a 2 quad-core Xeon (2.66 GHz), 8GB of RAM, takes between 1.5 to 3 hours. Although the overall process might appear “expensive”, it should be noted that it is instead rather inexpensive when compared to the cost required by a human designer to come up with a TORCS track.

We also performed a preliminary validation of the proposed approach with human subjects to test whether there exists an agreement between our fitness definitions and users' preferences. For this purpose, we designed two surveys, one focused on the subjects' visual preferences and one focused on their playing experience. Our preliminary results suggest that there is a statistically significant agreement between the subjects' visual preferences and our fitness definitions; overall, it appears that the agreement is stronger for simpler tracks (encoded with fewer control points) and when more experienced players are considered.

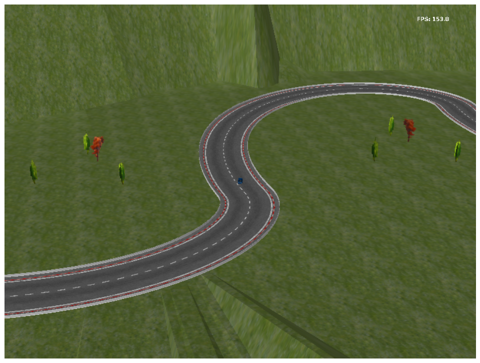
The results of the second survey, focused on the playing experience, suggests that the (evolved) tracks with a higher degree of diversity are perceived as more challenging (according to 92.86% of the subjects). Such level of challenge, however, does not prevent subjects from finding our evolved tracks appealing. In fact, although only 52.38% of all the subjects prefers racing on an evolved track, 71.43% of them wish to race again the (evolved) track with higher diversity instead of the control track with a lower degree of diversity. More interestingly, subjects with some experience in racing games show a stronger preference toward tracks with a higher degree of diversity. In fact, our results show that our evolved tracks have been preferred by 71.43% of the subjects that are familiar with racing games and 85.71% of those subjects wish to play them again.

As a proof of concept, while we were performing the experiments discussed in this paper, we selected two of the tracks evolved using the multi-objective genetic algorithm for the 2010 Simulated Car Racing Championship, an event joining three scientific competitions held at the ACM Genetic and Evolutionary Computation Conference (GECCO-2010), at the IEEE World Congress on Computational Intelligence (WCCI-2010), and at the IEEE Conference on Computational Intelligence and Games (CIG-2010). Figure 10 and Figure 11 show the track `Wild-Speed` and the track `Rocky` which were used during the second leg and the third leg of the championship. The evolved tracks were not modified and their shapes are just the result of the evolutionary process. We added however some scenic elements to make them more attractive for the competition attendees (for instance, we added a terrain, some trees, and changed the roadbed).

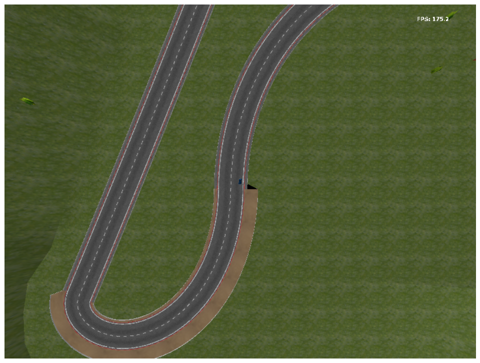
Search-based procedural content generation is a very promising research area and, in our opinion, the automatic generation of racing tracks is a very interesting direction deserving further investigation. In particular, our future research directions include (i) the extension for the evolution of richer



(a)

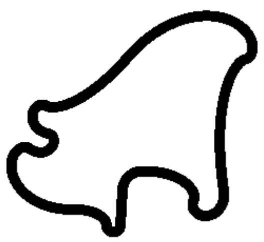


(b)

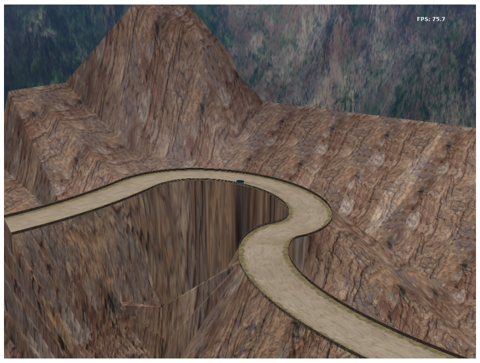


(c)

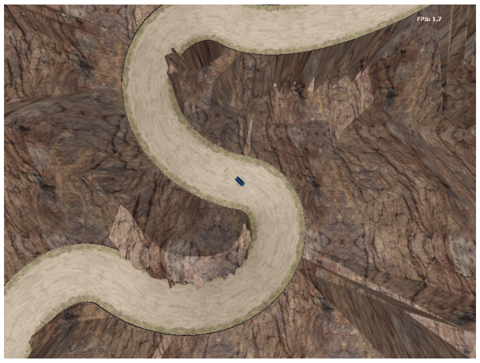
Fig. 10. The track `Wild-Speed` used in the second leg of the 2010 Simulated Car Racing Championship held at WCCI-2010, Barcelona, Spain: (a) the shape of the track, (b) and (c) screenshots of the track in the game.



(a)



(b)



(c)

Fig. 11. The track `Rocky` used in the third leg of the 2010 Simulated Car Racing Championship held at CIG-2010, Copenhagen, Denmark: (a) the shape of the track, (b) and (c) screenshots of the track in the game.

representations (e.g., by including some scenic elements such as the terrain); (ii) an improved analysis with human subjects designed to obtain a better evaluation of the metrics we proposed; last but not least, (iii) the addition of some sort of user intervention in the evolutionary process to hybridize the process with some human guidance.

X. Acknowledgements

The authors wish to thank Marco Colombo for his work on the preliminary implementation of the mapping procedure between genotypes and phenotypes.

References

[1] J. E. Laird and J. Schaeffer, Eds., *Procedural Level Design for Platform Games*. The AAAI Press, 2006.

[2] *MIGS: Far Cry 2's Guay On The Importance Of Procedural Content*, Nov. 2008, http://www.gamasutra.com/php-bin/news_index.php?story=21165.

[3] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *EvoApplications (1)*, ser. Lecture Notes in Computer Science, C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, C. K. Goh, J. J. M. Guervós, F. Neri, M. Preuss, J. Togelius, and G. N. Yannakakis, Eds., vol. 6024. Springer, 2010, pp. 141–150.

[4] "The open racing car simulator website." [Online]. Available: http://torcs.sourceforge.net/

[5] J. Togelius, R. D. Nardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in *Proceedings of the SAB Workshop on Adaptive Approaches to Optimizing Player Satisfaction*, 2006, available at http://julian.togelius.com/Togelius2006Making.pdf.

[6] J. Togelius, R. De Nardi, and S. Lucas, "Towards automatic personalised content creation for racing games," in *Proc. IEEE Symposium on Computational Intelligence and Games CIG 2007*, 2007, pp. 252–259.

[7] J. Togelius, S. M. Lucas, and R. D. Nardi, "Computational intelligence in racing games," in *Advanced Intelligent Paradigms in Computer Games*, ser. Studies in Computational Intelligence, N. Baba, L. C. Jain, and H. Handa, Eds. Springer, 2007, vol. 71, pp. 39–69.

[8] R. Frushtick, "Borderlands Has 3,166,880 Different Weapons," July 2009, http://multiplayerblog.mtv.com/2009/07/28.

[9] A. Doull, "The death of the level designer," Jan. 2008, http://pcg.

wikidot.com/the-death-of-the-level-designer.

[10] E. Hastings, R. Guha, and K. Stanley, "Neat particles: Design, representation, and animation of particle system effects," in *Proc. IEEE Symposium on Computational Intelligence and Games CIG 2007*, 2007, pp. 154–160.

[11] K. O. Stanley and R. Miikkulainen, "Evolving neural network through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[12] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Evolving content in the galactic arms race video game," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, Sept. 2009, pp. 241–248.

[13] J. Marks and V. Hom, "Automatic design of balanced board games," in *AIIDE*, J. Schaeffer and M. Mateas, Eds. The AAAI Press, 2007, pp. 25–30.

[14] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Computational Intelligence and Games, 2008. CIG '08. IEEE Symposium On*, Dec. 2008, pp. 111–118.

[15] M. Frade, F. F. de Vega, and C. Cotta, "Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience," in *Applications of Evolutionary Computing*, ser. LNCS, M. G. et al., Ed., vol. 4974. Napoli, Italy: Springer, 2008, pp. 485–490.

[16] J. Koza, *Genetic Programming*. MIT Press, 1992.

[17] M. El-Nasr, A. Vasilakos, C. Rao, and J. Zupko, "Dynamic intelligent lighting for directing visual attention in interactive 3-d scenes," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 1, no. 2, pp. 145–153, June 2009.

[18] M. Zheng and D. Kudenko, "Automated event recognition for football commentary generation," in *Proc. AISB'09 Symposium: AI & GAMES*, 2009.

[19] C. Pedersen, J. Togelius, and G. Yannakakis, "Modeling player experience for content creation," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 1, pp. 54 –67, mar. 2010.

[20] J. Togelius, M. Preuss, N. Beume, J. H. Simon Wessing, and G. N. Yannakakis, "Multiobjective exploration of the starcraft map space," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. Copenhagen, Denmark, 18-21 August 2010.: IEEE, 2010, pp. 265–262.

[21] N. Sorenson and P. Pasquier, "Towards a generic framework for automated video game level creation," in *EvoApplications (1)*, ser. Lecture Notes in Computer Science, C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, C. K. Goh, J. J. M. Guervós, F. Neri, M. Preuss, J. Togelius, and G. N. Yannakakis, Eds., vol. 6024. Springer, 2010, pp. 131–140.

[22] K. Sastry, "Single and multiobjective genetic algorithm toolbox in C++," Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Tech. Rep., IlliGAL Report No. 2007016, 2007.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 6, no. 2, Apr. 2002.

[24] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Appl. Artif. Intell.*, vol. 21, no. 10, pp. 933–971, 2007.

[25] C. Pedersen, J. Togelius, and G. Yannakakis, "Modeling player experience for content creation," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 2, no. 1, pp. 54 –67, 2010.

[26] C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcázar, C. K. Goh, J. J. M. Guervós, F. Neri, M. Preuss, J. Togelius, and G. N. Yannakakis, Eds., *Applications of Evolutionary Computation, EvoApplicatons 2010: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Istanbul, Turkey, April 7-9, 2010, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 6024. Springer, 2010.