Automatic Adaptation to Generated Content Via Car Setup Optimization in TORCS

Markus Kemmerling and Mike Preuss

***Abstract*— The car setup optimization problem as employed for a recent competition is a real-valued, 22 variable gray box problem (some dependencies are known) which challenges optimization algorithms in many ways. Runs must be short, there is a considerable amount of noise, and evaluation times for each solution candidate have to be determined by the algorithm itself. We take this as example problem and ask what happens if a user or a procedural content generator provides new tracks on which the standard cars are almost undriveable. Consequently, we suggest to use an optimization algorithm to adapt the cars to the track in almost real-time (minutes). We investigate how the CMA-ES, a modern evolutionary strategy, fares in this context and suggest some means to adapt it to the requirements. Attempts to improve the results using special noise handling methods unfortunately fail, most likely due to the very hard time constraints. Additionally, we perform a basic test with humans driving the optimized cars and have a short look at the properties of the cars changed for improving their performance.**

I. Introduction

Procedural Content Generation (PCG) is becoming a buzzword in games research nowadays, see [1] for a recent overview. It is astounding to see how computer programs are getting better and better at aiding game developers in generating interesting content, be it levels/maps, AI opponents, or other game components. However, there is also a problematic issue that occurs if parts of the game are generated anew or modified: Other constituents of the game may cease working or at least be heavily influenced by the changes. For example, setting up a map with features unseen before for a strategy game may render the existing AI opponents useless as they cannot handle the additions appropriately. This can happen regardless of who actually performs the changes, a PCG program or a human developer. It could even be a user who sets up new game components by means of provided editors, and without worrying too much about e.g. balancing issues. That balancing can be very fragile may be seen from the fact that some game companies invest months in getting it right. However, below balancing is pure usefulness. But how to get the overstrained game components 'back on track'? To our knowledge, this issue has not found much interest in games research yet, despite its importance given that PCG will probably be applied more and more often.

In the following, we present a concrete problem and a possible solution, implying that our solution could also be helpful in other, similar cases. The problem occurs in the context of the TORCS car setup optimization contest [2], where the task is to change the properties of a car as defined by 22 real-valued variables in order to drive the largest possible distance in a given time interval. This reminds of the car setup problem often reported for Formula-1 cars which are adjusted to the properties of the current track for days before a race starts. Next to the high dimensional search space and the expected interactions between different variables, this optimization problem possesses two main difficulties:

- Limited time. A budget of only 1 million tics is available, where for medium sized tracks around 5,000 tics are needed to drive one round.
- Noise. The optimized car switches its properties 'on-the-fly' when a new evaluation is started, meaning that measuring may begin at different positions distributed over the track. Evaluation time (in tics) is controllable, but for very short evaluations, we cannot drive a full round and will thus race on different portions of a track.

Besides, the racing controller (the AI component actually driving the car) is fixed and its internal structure largely unknown to contestants, but its behavior undoubtedly has an influence on the outcome. We thus have three independent but interacting game components in this setting, one controller, several tracks (which are also unknown during the competition), and a car which is subject to an optimization algorithm contestants have to provide. The car setup competition has already been run once with a slightly different

Fig. 1. Hairpin bend on Alpine1 in the TORCS game.

Markus Kemmerling and Mike Preuss are with the CI Group, Dept. of Computer Science, Technische Universität Dortmund, Germany
email: first-name.surname@tu-dortmund.de

978-1-4244-6297-1/10/$26.00 ©2010 IEEE

setup at the GECCO 2009 conference, and it became clear that the optimization problem to solve is quite difficult.

However, what happens if not only standard tracks are employed on which the used controller can be expected to drive well but if the tracks are modified? This resembles what a user or PCG system would do. We may end up with tracks that are 'undrivable' for a certain car/controller pair so that the effort of modifying the tracks would be lost as racing the track would be no fun any more for a human. Adjusting tracks in a way that they respect specific environmental conditions as e.g. weather effects is easy in TORCS, and we will give examples in the following section. These changes can completely break the driving/recovery behavior of existing controllers. This results in a situation where a human can drive a track and a controller that would usually handle the opponents can not, because the car reacts in a way that was not foreseen when creating the controller. Undoubtedly, setting up a race between a human and some uncunning controllers is not satisfying. Modifying a controller is highly time-consuming and can not easily be automatized. Thus we pose the question how we can reliably find a car setup that allows the fixed controller to act normally again.

Our proposed solution to this problem is to apply the *covariance matrix adaptation evolution strategy* (CMA-ES) [3] (we employ the variant documented in [4]) as a standard evolutionary optimization method for real-valued functions. Evolutionary algorithms are good candidates for moderately noisy optimization problems as they are known to have a certain robustness against distortions that stems from their rank-based and not gradient-dependent selection mechanisms. Other evolutionary methods as investigated e.g. in [5] and [6] would be available as well but we rely on the CMA-ES here as the function evaluation budget is very tight and rapid convergence to *any* good solution shall thus be weighted higher than pursuing a possibly better solution for too long. For a recent survey of evolutionary optimization techniques for uncertain environments, see [7].

In the second part of the paper, we investigate if a recently proposed CMA-ES variant, the *uncertainty handling CMA-ES* (UH-CMA-ES) [8] poses a viable alternative to the 'plain' CMA-ES for the car setup optimization problem. We attempt to identify under which conditions either algorithm is preferable, hoping that these recommendations may be useful also for similar applications. Before concluding, we shortly review the changes done to the cars for improving them and report from a basic user test done with unoptimized and optimized cars.

II. The Car Setup Problem in TORCS

TORCS[1], The Open Racing Car Simulator, is an open source multi platform car racing simulation that was started in 1997. It contains more than 50 cars, more than 20 tracks and makes use of a physics engine. The latter comprises a simple damage model and the simulation of different car properties and aerodynamics.

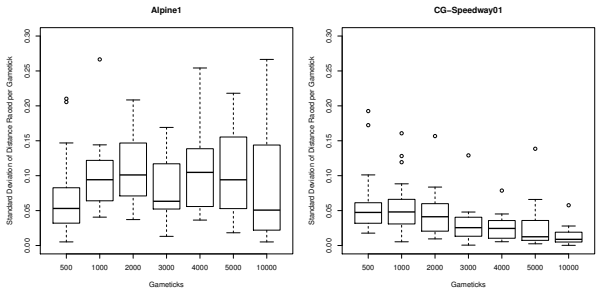


Fig. 2. Standard deviation of distance raced per gametic for a latin hypercube design with 20 elements. The low values for small gameticks are due to the start-up of the car

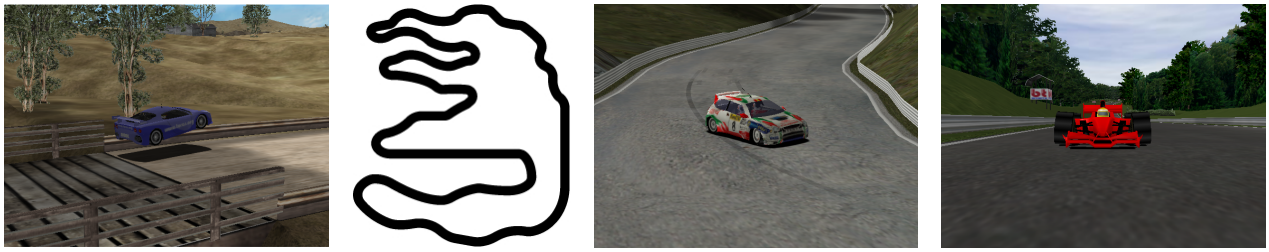
In order to adapt cars to new content, we manually created variants of the standard track *Alpine1* (see Figure 3(b)). *Alpine1* is by itself a difficult track for car optimization. With a length of more than 6 km, it is one of the longest contained in the TORCS distribution, and it consists of very different sections. The highest point of the track is around 140 m higher than the final stretch, meaning that only a small part is flat. After starting the round, a winding road with hairpin bends leads uphill. Downhill, the turns are wider and the track provides high speed sections, and finally two long flat straights (the 'bridge' and the final stretch). The motivation behind choosing this track was to obtain a demanding base track which could be modified in a way that Formula-1 cars could loose the race (for all standard tracks of the TORCS distribution except *Dirt2*, Formula-1 cars beat all other cars[2]).

We already stated that the problem contains uncertainties, especially if the sampling time is below the time to drive one round. To get an impression how this affects the measured fitness on different tracks, figure 2 shows how the noise level corresponds to the evaluation time for two tracks. *Alpine1* is on the left and the right side shows *CG-Speedway01*, which is shorter and has less variation in its sections. 20 car settings of the default car (a touring car) and controller were generated with a *latin hypercube design* (LHD) and each of them were sampled 11 times for a different number of gametics. The standard deviation of the distance raced per gametic within the 11 evaluations is considered as noise measure for this purpose.

As new content, we established two modifications of *Alpine1*. One is considered to be a cross-country track. We decreased the friction of the surfaces to simulate mud and dirt, and we increased the roughness to get bumps. The second variant resembles a snow track with frozen sections. For that, we decreased the friction and increased roughness and rolling resistance slightly for the snow parts (uphill and downhill) and decreased the friction even more for the ice parts (flat straights). According to the observations of test

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

[2] This reveals current weaknesses of TORCS, especially of the damage system, as crashes should render an F1 car useless, but still it drives on



(a) Touring Car (b) Alpine1 (c) Rally Car (d) Formula-1 Car

Fig. 3. The three car used in the optimization and the shape of Alpine1.

TABLE I
DISTANCE RACED IN KM BY THE STANDARD CARS ON THE THREE TRACKS WITHIN 100,000 GAMETICS.

Car	Alpine1	Cross-Country	Snow
Touring	101.88	26.84	60.79
Rally	89.40	63.95	59.16
Formula 1	118.69	2.51	43.33

players, this track is well drivable using rally cars, adding some interesting effects like sliding to the usual playing experience. In TORCS, tracks can easiliy be modified by editing XML files used to describe track properties. Each straight and each turn has its own entry in the file which specifies the length, the angle, the slope, the surface and others. Track segments also refer each to a surface structure which is specified as well in a file and determined by the properties friction, rolling resistance, roughness, roughness wavelength, and others that we did not change.

As base cars for the following optimization process, we chose three different cars as representatives of the three major types present in TORCS, a touring car, a rally car and Formula-1 car (see Figures 3(a), (c) and (d)). The three selected cars preformed quite differently on the three versions of the track. Table I shows the distance raced within 100,000 gametics. The rally car accomplishes the cross-country modification very well, but is much slower than the others on the unmodifed road track. The touring car and, especially, the Formula-1 car are mostly undriveable on the cross-country track due to their low ride-height and hard springs. The standard controller is only able to drive a few kilometers before getting stuck. However, this problem is as much rooted in the track properties as in the controller behavior as its recovery behavior (e.g. after sliding and turning direction) tends to fail. When the car stands disadvantageous in front of a crash barrier, the controller uses too little force to drive backwards and thus is not able to turn again. As long as the cars stay on course, the snow track works as 'leveler' where the three groups of (unmodified) cars do not show consistent speed differences.

As it is the case for tracks, cars are also specified in XML files. The competition software provides an interface to change a subset of their propterties during a race:

- gear ratios
- angles of front/rear wing
- brake system
- anti-roll bars
- wheels (toe, camber)
- suspension (springs, ride height)

There are 22 properties normalized to $[0, 1] \subset \mathbb{R}$ provided to the optimization, and these can be adjusted and simulated for a freely specified time (in gametics) in order to find the best setting for a track and a controller.

III. FIRST SOLUTION: CMA-ES

We now face the following PCG-related optimization problem: A content provider has established 2 new tracks for the TORCS environment (2 different variants of the *Alpine1* track) and we need to make sure that the existing standard cars can be driven well by an existing controller and/or a human player. In order to do that, we need to take 2 decisions:

- Select a suitable optimization algorithm
- Determine the number of tics used for every evaluation of a solution candidate

As it is known that evolutionary algorithms perform well even in the presence of noise [8], [7], [5], we focus exclusively on these methods. They benefit from the use of populations, which results in an implicit averaging over space, and from the genetic repair effect resulting from recombination. Ranking-based selection of individuals provides additional robustness, because the uncertainty can affect the selection only if the ordering of individuals is changed, regardless of absolute objective function values. Evolution strategies as the modern CMA-ES [9] are non-elitist, since (overrated) individuals survive only one generation. Despite their obvious advantages, it seems that these algorithms has only rarely been applied in the game context up to now.

As the car setup competition package already provides a simple *genetic algorithm* (GA) as default optimization algorithm, we first conduct a comparison between our method of choice and this 'baseline performance' GA. It shall be considered that this is not a fair comparison of two algorithms because we take the GA 'as is' and do not try to better adapt it to the problem. Rather, the GA serves as a 'litmus test': Our method should be reliably better. If not, it is obviously not suitable for the problem. Of course, it would also be

a viable alternative approach to improve the GA instead of using a CMA-ES. However, we disregarded this possibility as we presumed that the CMA-ES already contains many useful features (as learning variable interactions by means of the covariance matrix) which are missing in the GA.

To get a more complete picture, the original *Alpine1* track is also taken into account, although we are primarily interested in optimized cars for the modified tracks. Since we consider the case when a user/program creates a new track and the game must be rapidly adapted to it, we use the contest setting with only 1 million tics. In real time, this optimization takes only few minutes on modern computers.

The (Evolutionary) Optimization Algorithms Compared

- CMA-ES: In the *covariance matrix adaptation evolution strategy* (CMA-ES) in each generation (iteration) λ new search points, $\vec{x}_i \in \mathcal{R}^n$ are generated according to a multivariate normal distribution with mean $\vec{m}$ and covariance matrix $\sigma^2\mathbf{C}$. In other words, the mean vector is perturbed by normally distributed zero-mean mutations. The mutations have covariance matrix $\mathbf{C}$ and are scaled by step-size σ. For $i = 1, \ldots, \lambda$,

$$\vec{x}_i = \vec{m} + \sigma\mathcal{N}(\vec{0}, \mathbf{C}) \; . \tag{1}$$

  After the evaluation of all search points, mean, step-size and covariance matrix are adapted using the better half of the points and given recombination weights. The mean is set to a weighted mean of the selected search points. For adapting the step-size, a search path is followed over roughly $n/3$ generations and its length is evaluated. A long path leads to increments of the step-size, a short path leads to a decrement. A search path is also used to update the covariance matrix. The covariance matrix is changed in order to increase the likelihood of selection mutation steps to appear again.
- GA: We employ the GA provided with the contest software. It utilizes a population of size 25 and evaluates each individual for 2,000 gametics, which is our only adjustment. The initial population is uniformly distributed randomly generated. A tournament selection with tournament size two determines the parents, selecting the best of two randomly chosen individuals. Crossover and mutation is used to create offspring. With probability 0.9, a pair of parents is recombined with uniformly distributed one point crossover. After that, a uniformly distributed random number between -0.025 and 0.025 is added to each parameter with probability $\frac{1}{22}$.

Experiment 1: Can we adapt a car to a newly generated track in adequate time by means of the CMA-ES, outperforming the GA?

Pre-Experimental Planning. To find good parameter settings for the CMA-ES, we preformed a grid test with three dimensions, namely starting points, initial mutation step-sizes and population sizes. Each combination of 20 *latin hypercube design* (LHD) generated starting points, initial step-sizes 0.083, 0.11, 0.17, 0.22, 0.33, and population sizes 8, 11, 13, 19, 26 was run 11 times on seven different tracks (*Alpine1, CG-Speedway01, C-Speedway, Dirt1, Dirt4, E-Track-2, Mixed01*), employing the touring car. Since the CMA-ES used only 2,000 gametics to evaluate each individual in the optimization process, the returned car setting was validated nine times for 10,000 gametics. The mean of the distance raced per gametic in the nine final evaluations is considered as 'real' fitness of the car setting and treated as the algorithm's performance. Figure 4 documents the result of the parameter test. Each subfigure shows the median performance of each parameter combination on two tracks. The most promising combination, presented by the red dot, is starting point (0.869, 0.618, 0.86,0.086, 0.514, 0.373, 0.724, 0.488, 0.059, 0.907, 0.918, 0.479, 0.332, 0.795, 0.219, 0.009, 0.428, 0.421, 0.025, 0.766, 0.253, 0.529), step length 0.11 and population size 19. For this point, the range between observed minimum and maximum and between the 1st and 3rd quartile of the obtained performances is represented by the dotted and solid lines, respectively. Note that the task of this test has not been to select a good starting point for the optimization process. Indeed, the best point obtained is valid only for the touring car and would be different for the others. However, we wanted to detect

- how important the starting point is in relation to the other two parameters, and
- if the best starting points are good for several tracks.

It can be assessed from the results of the grid test that the starting point is at least as important as the two other parameters, and (as seen in figure 4) that good starting points seem to stay good for many tracks (the red marked point is non-dominated or only dominated by a small margin).

The first five dimensions of the individuals represent gear ratios. During simulation, wrongly set gear ratios (higher follows lower) are skipped. If this is untreated, it leads to over 90 per cent of the best found individuals containing invalid gear ratios. We suggest to handle that simply via bringing the ratios in correct order via sorting before the evaluation starts and test the success of this transformation on 51 CMA-ES parameter combinations, selected from the full performance spectrum. The aggregated results (11 repeats) are compared to the outcomes of the original grid test per track as documented in table II. As seen from the results, gear ratio sorting mostly leads to a performance increase but rarely to a decrease, so that we use this simple repair mechanism for all following optimization runs.

TABLE II
IMPROVEMENT VIA GEAR SORTING TRANSFORMATION.

Track	# Improvements	# Degradations
Alpine1	47	4
CG-Speedway01	43	8
C-Speedway	44	7
Dirt1	36	15
Dirt4	38	13
E-Track-2	44	7
Mixed01	40	11

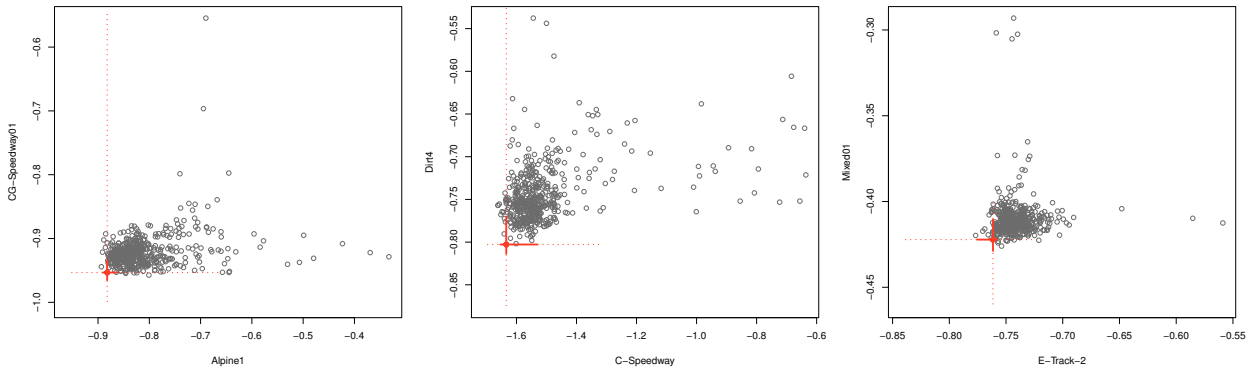


Fig. 4. Pareto comparison plots of different parameter combinations for the CMA-ES. Each point resembles the median performace on two tracks of a specific combination consisting of starting point, initial steplength and population size. The red point marks the best found setting of a starting point, step length 0.11 and population size 19. The dotted and solid red lines show the range between observed minimum and maximum, and 1st and 3rd quartile of the observed performance, respectively

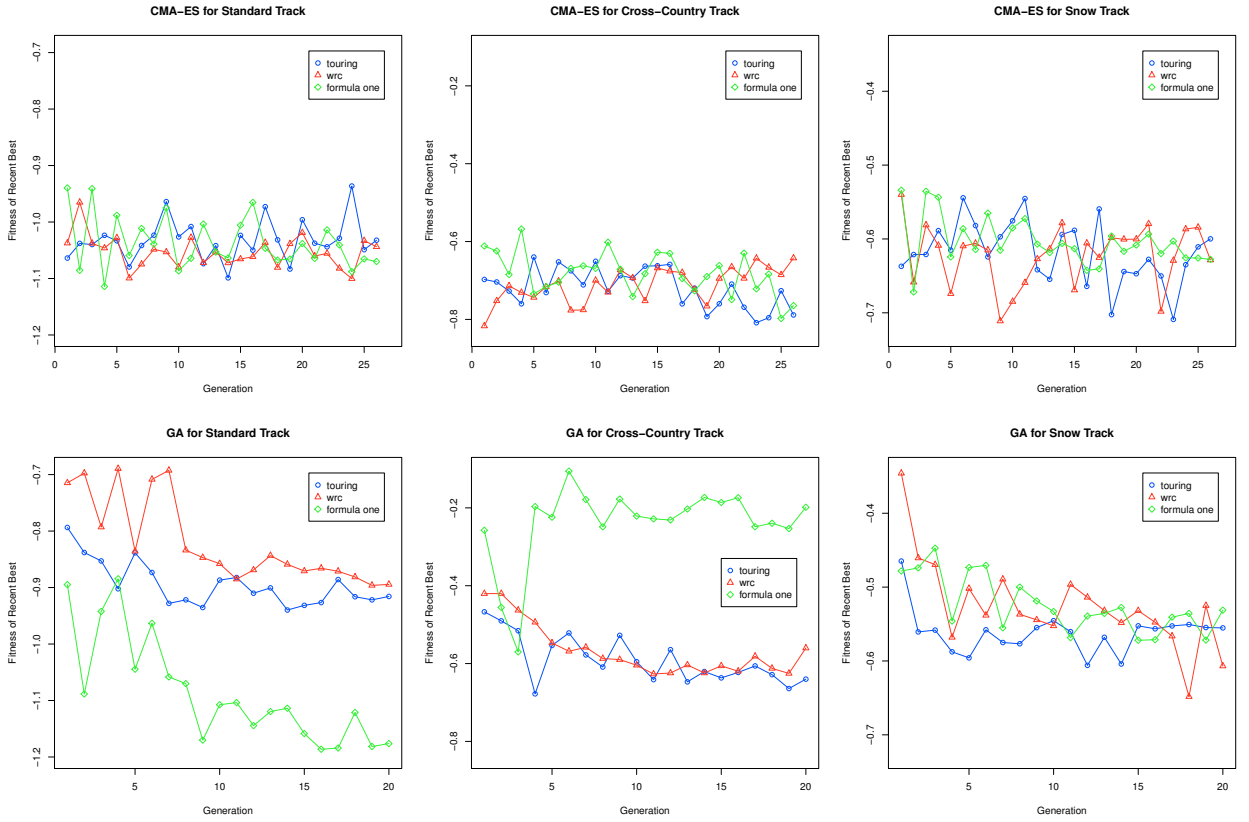


Fig. 5. Best runs of CMA-ES (top row) and GA (bottom row) for the three cars on standard Alpine1 Track, cross-country and snow modification. Graphs show the (unvalidated) fitness value of the population's best individual. Note the different scaling.

Task. We require that the optimized cars perform significantly better than the standard cars and the GA-optimized cars in most of the cases, which is attested by means of random permutation testswith 95% confidence limits.

Setup. Each of the three cars is optimized 11 times on all 3 track variants with both algorithms. According to the grid test, we set initial step-size and population size to 0.11 and 19 for the CMA-ES, respectively. To better understand the applied car modifications, we deviate from the test and choose the particular standard car setting as starting point. All other parameters are set to default values as recommended in [9]. As a car setting may be overrated due to noise, we additionally investigate the best setting found in the last generation. Since the CMA-ES creates each generation from a mean, the last calculated mean is most likely the best approximation for the optimum. Therefore, we validate the best ever, the recent best and, in case of the CMA-ES, the recent mean for 100,000 tics. On standard *Alpine1*, this resembles approximately 14 rounds. The distance raced in this time is considered as 'real' fitness value. Within the optimization, car damage is reported but does not influence the driving. This must be done to prevent accidents which may affect the next evaluations. For the final scoring, the simulation is consistently restarted and the damage model is active. Both algorithms use the following objective function to determine car fitness.

$$F(x) = \begin{cases} -\frac{\text{distance raced}}{\text{gametics}} & \text{, if} \quad \text{damage} < 100 \\ \text{damage} & \text{, else} \end{cases} \tag{2}$$

With enabled damage model, a car would be taken out of race when its damage exceeds 10,000. The raced distance is normalized by gametics to make fitness values with different simulation times comparable and is negated to enable minimization. Distance, damage and tics correspond only to the particular simulation, they are not cumulated through the optimization process.

Results/Visualization. For comparison, table I shows the distance raced within 100,000 tics by the standard cars. Tables III(a), (b) and (c) present the optimization results after 1 million tics. For each of the 11 repeats, the car settings obtained as overall best, last population's mean or last population's best were validated for 100,000 tics and we show the median raced distances. Figure 5 documents the fitness development during CMA-ES and GA best runs as perceived by the algorithms (unvalidated fitness) on the three tracks. The three cars are represented by different colors: Blue for touring, red for rally and green for Formula-1. Note that graphs of different tracks have different y-axis scalings. Bootstrap permutation tests ([10], section 14.5) with 20,000 resamples between the unoptimized and the CMA-ES optimized validated car configurations on all 9 track and car combinations attest that significant improvements are achieved with the rally car on the original track and the Formula-1 car on the cross-country track, whether significant losses happened for the touring car and the Formula-1 car on the original track. The same test between the CMA-ES and GA results show significant better performance of the CMA-ES in all cases but the touring and rally cars on the snow track and the Formula-1 car on the original track. In the latter case, the GA is significantly better.

Observations. As expected, the optimization of the three cars for the two modified versions of *Alpine1* results in better cars, although these are not in all cases statistically significant. On standard *Alpine1*, both algorithms failed to generate better settings, except for the rally car, which has been improved by the CMA-ES for about 10%. For the cross-country modification, the CMA-ES created better cars, especially for the touring and the Formula-1 car, and the GA is outperformed by a huge margin on this track. Although the median of the distance raced with the best touring and rally cars created by the CMA-ES on the snow modification does not differ much from the performance of the standard cars, the best generated settings actually performed much better. Whether the GA has positive and negative outliers (Formula-1 car on original and cross-country track), the CMA-ES runs for different cars on the same track seem to stay closer together.

All figures clearly show the roughness in the fitness development due to noise. According to the random population initialization of the GA, it starts in worse regions than the CMA-ES, which initial population mean is the particular standard car. But mostly, a trend to better regions is observable. Since the CMA-ES starts in better regions in the search space, the progress is not as clear as for the GA. On the modified tracks, we identify a trend towards smaller fitness values. In particular, this is the case for the touring and the Formula-1 car on the cross-country modification.

Discussion. We can conclude that for the modified tracks, our PCG-driven optimization leads to dramatic improvements at least for the cars that obviously have problems to cope with them at all (Formula-1 and touring on the cross-country track and Formula-1 on snow). The manually set original car parameters are however hard to improve on a standard track. If improvement is possible at all, the CMA-ES performs much better than the GA in all but one case. However, it seems that adapting the cars to new tracks well instead of just making them usable again is a very hard problem if the available budget is as low as 1 milion gametics. The task that was set before the experiment started has been fulfilled to a large extent. Concerning the comparison to the GA, however, the CMA-ES did not as well as expected compared to the original car settings as only few significant improvements were achieved.

IV. Second Solution: UH-CMA-ES

Although EAs expose advantages in noisy environments, it is possible to address the uncertainty explicitly. Jin et al. [7] give a survey about the options regarding uncertainty. Hansen et al. [8] derived a uncertainty measure and handling, which they implanted into the CMA-ES. We conjecture that such an explicit uncertainty handling is beneficial for the problem considered here. Since the CMA-ES already showed

TABLE III

MEDIAN PERFORMANCE OF THE CMA-ES AND THE GA OVER 11 RUNS REGARDING THE OVERALL BEST CAR SETTING FOUND, THE LAST POPULATION'S MEAN AND THE LAST POPULATION'S BEST ('RECENT BEST'). EACH ENTRY REPRESENTS 11 CAR SETTINGS AND SHOWS THE MEDIAN OF THEIR DISTANCE RACED (IN KM) WITHIN 100,000 TICS

(a) Alpine1

Car	CMA-ES			GA	
	best	mean	recent best	best	recent best
Touring	100.39	98.22	97.67	87.74	89.93
Rally	100.67	99.45	99.00	80.59	80.64
Formula-1	99.30	98.77	98.38	101.00	101.00

(b) Cross-Country Track

Car	CMA-ES			GA	
	best	mean	recent best	best	recent best
Touring	41.02	60.29	25.39	16.15	26.83
Rally	57.55	64.27	60.06	46.23	45.23
Formula-1	49.76	58.88	44.08	9.58	4.31

(c) Snow Track

Car	CMA-ES			GA	
	best	mean	recent best	best	recent best
Touring	59.58	56.95	51.42	53.77	51.64
Rally	60.92	60.85	60.44	50.17	49.31
Formula-1	10.04	48.17	22.93	4.87	15.99

good performance, we try to obtain an improvement with the technique introduced in [8].

A. The Uncertainty Handling

[8] is composed of two parts, a measurement and a treatment of the uncertainties. For the measurement, a small number of solutions is evaluated twice and the rank for the second evaluation is compared to the original rank. If the rank change exceeds a threshold, the noise treatment is invoked. Two possible treatments have been proposed: increase of the step-size σ (see (1)) and increase of the gametics for a single evaluation. In contrast to the step-size, the gametics are decreased again, if the rank change is below the threshold. All changes are accomplished by multiplication with a factor α as given below.

The two treatments play different roles. Increasing the step-size is cheap and allows to move away from too noisy search space regions. With a small gametics number, it also allows the algorithm to run a larger number of iterations and therefore assists the adaptation of the covariance matrix, which needs a considerable number of iterations. Increasing the gametics is more expensive, but allows to approach an optimum potentially with arbitrary precision. In a final stage of optimization, increasing of the step-size becomes undesirable, as an optimum should be approached closely. These considerations lead to a three stages approach as explained below.

B. A Failed Experiment

We have to admit that despite several different approaches to improve the results of the CMA-ES by means of the UH-CMA-ES, no clear advantage has been obtained. This may lead to two conclusions:

- The CMA-ES results are already very good (given the environmental conditions, especially under the very hard runtime constraints). This view is also supported by the results of the EvoStar 2010 competition results, where the inofficial organizers entry of Markus Kemmerling clearly outperformed all other approaches.
- Under tight runtime conditions, it may not be easy to weight the effort put into uncertainty handling and the one put into the optimization itself.

Although we cannot clearly attribute our failure to successfully apply the UH-CMA-ES to one single reason (it could also be a bug in the code we did not detect), we would like to give a short overview of what we tried and thus communicate our limited understanding of the matter.

All in all, three UH-CMA-ES setups have been investigated. For the first setup, most uncertainty treatment parameters are set to the values recommended in [8], and CMA-ES parameters are kept as used in sect. III. Initial evaluation time is 2000 tics, $\sigma = 0.11$, population size 19, $\alpha_\sigma = 1.07$, $\alpha_{time} = 1.2$, $\theta = 0.3$, $c_s = 1$ and $r_\lambda = \frac{2}{19}$. We choose α_{time} lower and θ higher to prevent wasting the highly limited time. Since we have no "frozen noise", solutions selected for reevaluation are not mutated. In the second setup, we adjust $\theta = 0.5$.

The last setup is inspired by the results of some tests with longer runtimes (100 million tics). We divide the available 1 million tics into 3 phases. At first, only σ is adjusted. In the second phase, σ and the simulation time, and in the third, only the tics are adapted according to the noise. We choose 300,000 tics each for the first and second phase, the third phase runs for the remaining 400,000 tics. We start with evaluating fast (in 1,000 tics), in order to cheaply adapt the covariance matrix. Evaluation time is increased during the second and third phase to evaluate more precisely. We set $\alpha_{time} = 1.6$ and $\theta = 0.2$ for a fast increase. In the third phase, the σ adaptation is disabled to amplify convergence towards the optimum. We would like to note that the originally employed setup with 500 tics for one evaluation, $\alpha_{time} = 1.2$, and $\theta = 0.5$ performed especially bad. The short evaluation time seemingly influenced the gear ratio. The resulting cars had a fast acceleration and a very low top speed which is not appropriate for the considered tracks. Therefore, the initial amount of evaluation time was set to 1,000 tics. However, even with this modification, the 3 phase setup did not show a clear advantage.

While in comparison, some improvments have been obtained for single configurations (of car and track), we also have losses for others so that the emerging picture is not very consistent.

Looking out for the reasons of our failure, we have to admit that the resampling mechanism was applied here in a way that does not match the problem well, and that it is not trivial to cure this because reevaluations usually follow the evaluation of the population and may cover only a

small part of the track. Long evaluations (for full rounds) however use up much of the available budget and therefore shorten the number of generations considerably. Moreover, the information hiding enforced by the competition system clearly hinders combining algorithm and problem in a way that very obvious sources of fitness value distortion—like track segment differences—could be treated properly. E.g., it is not possible to let the car drive exactly one round or even to detect the maximum speed of a standard car for the part of the track that is driven within one evaluation.

V. Observed Car Improvements

In the following, we examine some of the changes actually done in car setups within the best obtained individuals. For the cross-country track, higher ratios for the lower gears and harder anti-roll bars are observed. The ride height is increased for the Formula-1 car, whether for the touring car, this is only done at the front. The Formula-1 car needs less and the touring car more suspension course. To gain more grip, the slant of the rear wing is increased for the Formula-1 car. For the rally car, the camber must be increased in the front and decreased in the rear.

For the snow modification, the maximum break pressures for Formula-1 and touring car is decreased. As for the cross-country modification, the anti-roll bar gets more stiff and the ratios for the lower gears are increased. The front ride height of the Formula-1 car is increased and the suspension course decreased. For the touring car, the right hight must also increased in the rear but the suspension course must be increased.

To improve the rally car on *Alpine1*, all gear ratios are increased together with the break pressure, and anti-roll bars and springs are hardened. Unexpectedly, the ride hight is increased. As the rally car has only low ground effect, this barely decreases the grip.

VI. Basic User Test

In order to detect if the obtained best cars for the 2 new tracks lead to considerable improvement in player satisfaction, we performed a basic user test. It has been established as blind test where human testers had to first state how experienced they are with car racing games and then drive 2 versions of the same car and make a preference decision. One car version used the original setup and one represented the best solution obtained by the CMA-ES in experiment 1.

As the collected data contains only 21 individual user preference decisions between 2 cars distributed over 3 cars and 3 tracks, a statistical assessment is not meaningful. However, we would like to give some impressions. Despite the obvious improvements made on Formula-1 and touring cars, the testers did not like driving them on the modified tracks as they were still very difficult to handle. This problem may be rooted in the interface which only allows to use full throttle or brake without any intermediate steps, which is different from the AI controller interface. The rally cars were much easier to drive, and here, the experienced testers attested that the optimized cars were more fun to drive, and especially liked driving on the snow track. This is interesting as the measured improvements were much smaller than e.g. for the Formula-1 car. It seems that there is a correlation between controller and human ability to drive specific cars, but it can be obscured by other effects as general unsuitability of a car for a certain track.

VII. Conclusions

First of all, we state that the treated car setup optimization problem under the given time and interface constraints is very tough, and that the difficulty grows with track length and feature variance. Our first solution applies the CMA-ES without uncertainty handling, supplied with the original configuration as starting point. This combination is almost always better than the default GA-based optimizer provided with the interface and robustly leads to improvements especially in cases where the original car setup is highly unsuitable for a specific track (e.g. Formula-1 on the cross-country track). Additionally, the rally car has been improved considerably on the original *Alpine-1* track. Applying the UH-CMA-ES did not lead to any improvments over the CMA-ES as already discussed. In conclusion, the adaptation of game components to user/PCG-provided new content works reasonably well with the CMA-ES, but improving these results by means of the UH-CMA-ES will require further investigations.

References

[1] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation," in *Proceedings of 2nd European event on Bio-inspired Algorithms in Games (EvoGAMES 2010), in EvoApplications 2010*, ser. Lecture Notes in Computer Science. Berlin: Springer, 2010.

[2] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Car setup optimization competition software manual," Politecnico di Milano, Dipartimento di Elettronica e Informazione, Italy, January 2010.

[3] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 159–195, 2001.

[4] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," in *Proc. 2005 Congress on Evolutionary Computation (CEC'05)*, B. McKay *et al.*, Eds. Piscataway NJ: IEEE Press, 2005, pp. 1769–1776.

[5] D. V. Arnold, *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers, 2002.

[6] D. V. Arnold and H.-G. Beyer, "A comparison of evolution strategies with other direct search methods in the presence of noise," *Computational Optimization and Applications*, vol. 24, no. 1, pp. 135–159, 2003.

[7] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments – a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[8] N. Hansen, S. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 180–197, 2009.

[9] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*, J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, Eds. Springer, 2006, pp. 75–102.

[10] G. P. Mccabe and D. S. Moore, *Introduction to the Practice of Statistics*. W.H. Freeman & Company, 2005.