Arms races and car races

Julian Togelius and Simon M. Lucas

Department of Computer Science
University of Essex
Colchester CO4 3SQ, UK

Abstract. Evolutionary car racing (ECR) is extended to the case of two cars racing on the same track. A sensor representation is devised, and various methods of evolving car controllers for competitive racing are explored. ECR can be combined with co-evolution in a wide variety of ways, and one aspect which is explored here is the relative-absolute fitness continuum. Systematical behavioural differences are found along this continuum; further, a tendency to specialization and the reactive nature of the controller architecture are found to limit evolutionary progress.

1 Introduction

Evolutionary car racing (ECR) is about using evolutionary algorithms to create and tune controllers, sensors or other parameters for racing cars, in simulation or physical reality. Only a few attempts to evolve controllers or aspects of controllers have so far been made, all quite recently [1][2][3][4]; see [5] for a complete review. Our own work has focused on investigating various controller architectures and sensor representations, and finding ways of developing neurocontrollers with general driving skills that can proficiently race a variety of tracks, as well as specialized controllers that perform very well on particular tracks. We have also argued that car racing is a promising environment for evolving complex and relatively general intelligence, as the task of navigating a basic track is relatively simple to learn, but gradually can be made more and more complex almost without limits, requiring path planning, anticipation, opponent modelling, etc.

All published research on ECR so far has dealt with the case of a solo-racing, or one car on a track a time. This paper addresses the more complex case of two cars competing against each other on the same track at the same time, and includes the possibility of car-car collisions. We will explore different methods of evolving neurocontrollers and sensor setups for successfully competing against another car, and we hope that our results will be useful both for game developers looking to automatically create racing game AI, and computational intelligence researchers seeking to use games and game-like environments to evolve ever more general and complex intelligence.

1.1 Co-evolution

In our previous research, a controller's fitness was defined as the progress a controlled car had made around a track within a pre-specified time, and so depended

only on the controller itself and a few small random factors. Competing against another controller fundamentally changes the problem, so that fitness becomes dependent on the behaviour of both the assessed controller and its competitor. Evolution with such fitness functions is commonly called co-evolution, and has been used in evolutionary computation both to improve evolvability and to study evolutionary dynamics [6][7].

ECR allows the application and exploration of several uncommon forms of co-evolution. According to Dawkins and Krebs, biological co-evolution can be either intraspecific or interspecific and either symmetric or asymmetric [8]; in evolutionary computation terms, the co-evolution can be either between two populations, or individuals in one population, and between contestants using the same or different fitness functions. ECR allows all these types of co-evolution, which is interesting since most competitive co-evolutionary robotics experiments we know of build on predator-prey scenarios, and thus fall in the asymmetric interspecific category [9][10][11].

A second way in which ECR allows uncommon modes of co-evolution is through the existence of a well defined solo fitness function: any controller can be tested both for absolute solo fitness, which means the distance covered when racing without competition, absolute competitive fitness, which is the same thing when having to take the behaviour of another car into account (including the possibility of collisions), and relative fitness, which is defined as how far in front of or behind the competitor a controlled car finishes. Further, absolute competitive fitness and relative fitness can be blended seamlessly. We believe that these characteristics make ECR ideal for exploring co-evolution.

1.2 Scope of this paper

The first set of questions we will try to answer concern the extension of the car racing model and evolutionary approach to two cars: how well will controllers evolved for solo racing do with competition? Will it be possible to co-evolve controllers that do better? Is our controller architecture and sensor setup appropriate for this? Will we be able to evolve human-competitive drivers, and if not, what are the problems with our method?

The second set of questions address co-evolution. Will there be a difference in fitness, and in behaviour, if we evolve for absolute, relative or mixed absolute and relative fitness? What sort of difference will be observed? For example, will controllers evolved for relative fitness turn out to drive more aggressively? Will there be a difference in sensor setups?

2 Methods

2.1 Simulation Environment

The experiments reported in this article were done in a slightly updated version of the simulator used in [5]. The 2-dimensional simulator is intended to, qualitatively if not quantitatively, model a standard radio-controlled (R/C) toy car

(approximately 17 centimeters long) in an arena with dimensions approximately 3*2 meters, where the track is delimited by solid walls. The simulation has the dimensions 400*300 pixels, and the car measures 20*10 pixels.

A track consists of a set of walls, a chain of waypoints, and a set of starting positions and directions. Cars are added to a track in one of the starting positions, with corresponding starting direction, both the position and angle being subject to random alterations. The waypoints are used for fitness calculations.

The dynamics of the car are based on a reasonably accurate mechanical model, taking into account the small size of the car and bad grip on the surface, but is not based on any actual measurements [12][13]. While the dynamics of the car itself are fairly straightforward, the collision handling has been subject to much tuning and exception-handling in order to get a behaviour that feels right for the human player and cannot easily be exploited in an unintended way by the evolutionary algorithm. A collision between two cars is basically handled as a fully elastic collision, but the orientations of the cars are also disturbed, depending on which parts of the cars collided.

2.2 Sensors

The car experiences its environment through four types of sensors: the speed sensor, the waypoint sensor, a number of wall sensors, and a number of car sensors. The speed sensor is simply the speed of the car. The waypoint sensor gives the difference between the car's current orientation and the angle to the next waypoint (but not the distance to the waypoint). When pointing straight to a waypoint, this sensor thus outputs 0, when the waypoint is to the left of the car it outputs a positive value, and vice versa.

The wall sensors are modelled on "range-finders" similar to sonars or IR sensors, where each sensor has an angle (relative to the orientation of the car) and a range, between 0 and 200 pixels. The output of the wall sensor is zero if no wall is encountered along a line with the specified angle and range from the centre of the car, otherwise it is a fraction of one, depending on how close to the car the sensed wall is. The car sensors work exactly like the wall sensors, with the crucial difference that the output depends on whether and how far along the line another car is detected. A small amount of noise is applied to all sensor readings, as it is to starting positions and orientations.

2.3 Controller Architecture

The controllers in the experiments below are based on neural networks. More precisely, we are using multilayer perceptrons with three neuronal layers (two adaptive layers) and *tanh* activation functions. A network has at least three inputs: one fixed input with the value 1, one speed input in the approximate range [0..3], and one input from the waypoint sensor, in the range [$-\Pi..\Pi$]. In addition to this, it has eight inputs from wall and car sensors, in the range [0..1]. All networks have two outputs, which are interpreted as driving commands for

the car. Both the neural network and sensor configuration of a controller are directly encoded together in the genome as an array of real numbers.

2.4 Co-Evolutionary Algorithm

For the co-evolutionary algorithm, a modified $(\mu + \lambda)$ evolutionary strategy with $\mu = 50$ and $\lambda = 50$ without self-adaptation) was used. (This algorithm is based on the EAs used in [3] and [5]. It is possible that the addition of crossover and/or self-adaptation could make evolution more efficient, but we chose to leave these out for the sake of conceptual simplicity and minimizing the number of parameters to tune.) The difference between the co-evolutionary algorithm used here and a standard evolutionary strategy is in the fitness calculation. There are two types of primitive fitness defined: absolute and relative fitness. The absolute fitness of a controller C is calculated as the number of waypoints it has passed, divided by the number of waypoints in the track, plus an intermediate term representing how far it is on its way to the next waypoint. An absolute fitness of 1.0 thus means having completed one full track within the alloted time. In the evolutionary experiments reported below, each car was allowed 700 time-steps (enough to do two to three laps on most tracks in the test set). Relative fitness is defined as the difference in absolute fitness between C and the car it is competing against. Both the absolute and relative fitness values for a given controller was calculated as the mean of three trials of the controller on each of the tracks.

When the primitive fitnesses of all the controllers have been calculated, they are normalized, so that they are all in the range [-1..1]. The final fitness value of each controller is then calculated by blending the two primitive fitness values: $fitness = p * absfit + (1 - p) * relfit$ where p is the proportion of absolute fitness, a constant set at the beginning of the evolutionary run. It could be argued that only evolution with completely relative fitness constitutes co-evolution.

There are three mutation operators: Gaussian mutation of all neural connection weights, Gaussian mutation of all sensor parameters (angles and lengths), or sensor type mutation. Each time the mutation method of a controller is called, numbers drawn from a Gaussian distribution with a standard deviation of 0.1 are added to both neural connection weights and sensor parameters. With a probability of 0.4, a sensor type mutation is also performed, meaning that one of the sensors has its type changed from car to wall or wall to car.

At the start of an evolutionary run, all controllers have four wall sensors and four car sensors, pointing in random directions and with random ranges, and the neural connection weights are initialized to small random values.

2.5 Competition tracks

In order for the competitions to be more challenging, and to prevent the controllers from adopting strategies that would only work on a single track, three different tracks were used to evaluate every trial (see figure 1). While we have

previously shown [5] that controllers can be evolved that proficiently race a diverse collection of tracks, this seems to require a lengthy process of incremental evolution if the tracks are both clockwise and counter-clockwise. But if all the tracks have the same direction, like the three tracks chosen for the present experiments, it is possible to evolve a good controller for these tracks using standard evolution.

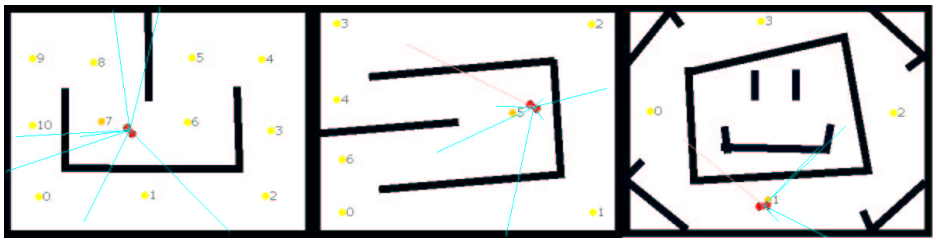

Fig. 1. The three tracks used in the experiments, including waypoints. Each track also shows a sample car with evolved sensors (discussed in section 3.2)

3 Experiments

3.1 Giving solo-evolved controllers some competition

10 separate solo-evolutionary runs were made according to the setup described in the Methods section above. Each evolutionary run lasted for 200 generations. (The mean fitness was zero at generation 0 of every evolutionary or co-evolutionary run in this paper; fitness growth graphs have been omitted to conserve space.)

On average, the best individual of the last generation of each of the evolutionary runs had fitness 2.49 (with standard deviation $\sigma = 0.23$), and used 5.7 ($\sigma = 0.67$) wall sensors and 2.3 (σ=0.67) car sensors. The best run resulted in a best controller with fitness 2.67, and the best controller of the worst run had fitness 1.89. Most of the evolved sensor setups consisted in a relatively even spread of medium-range wall sensors pointing forward and diagonally forward, and the few car sensors pointing backward.

One of these controllers, with fitness 2.61 (0.13), was selected for further testing. When put in a competition with another car controlled by a copy of the same controller, average fitness dropped to 1.23 (0.6). Behavioural analysis shows that the two cars collide repeatedly at the beginning of almost every trial, as they don't have any method of detecting and reacting to each other's presence. Depending on starting conditions, the outcome of the competitions vary, but usually one or both of the cars is either driven to collide with the wall, or spun around so that it starts driving the track the wrong way. A car that starts going the wrong way is usually, but not always, unable to turn around and start

driving in the correct direction again; a car that crashes into the wall usually gets stuck. This is because of the controller design rather the game mechanics, as it is perfectly possible for a human player to back away from the wall and continue driving in the right direction. In many trials, however, one of the cars managed to escape the collisions in the right direction and proceeded to make its way smoothly around the track.

From this experiment, it can be seen that the problem of racing two cars concurrently is sufficiently different from the problem of solo-racing that the performance of a solo-evolved controller is catastrophically compromised when tested in competition conditions.

3.2 Co-evolving controllers: The absolute-relative fitness continuum

50 evolutionary runs were made, each consisting of 200 generations. They were divided into five groups, depending on the absolute/relative fitness mix used by the selection operator of the co-evolutionary algorithm: ten evolutionary runs were performed with absolute fitness proportions 0.0, 0.25, 0.5, 0.75 and 1.0 respectively. These were then tested in the following manner: the best individuals from the last generation of each run were first tested for 50 trials on all three tracks without competitors, and the results averaged for each group. Then, all five controllers in each group were tested for 50 trials each in competition against each controller of the group. Finally, the number of wall and car sensors were averaged in each group. See table 1 for results.

Proportionabsolute	0.0	0.25	0.5	0.75	1.0
Absolute fitness solo	1.99 (0.31)	2.09 (0.33)	2.11 (0.35)	2.32 (0.23)	2.23 (0.23)
Absolute fitness duo	0.99 (0.53)	0.95 (0.44)	1.56 (0.45)	1.44 (0.44)	1.59 (0.45)
Relative fitness duo	0 (0.75)	0 (0.57)	0 (0.53)	0 (0.55)	0 (0.47)
Wall/car sensors	5.8 / 2.2	5.6 / 2.4	5.2 / 2.8	4.2 / 3.8	6.4 / 1.6

Table 1. The results of co-evolving controllers with various proportions of absolute fitness. All numbers are the mean of testing the best controller of ten evolutionary runs for 50 trials. Standard deviations in parentheses.

Analysis It is clear that, when driving without competitors, the co-evolved controllers on average have lower absolute fitness than the solo-evolved controllers. Behavioural inspection suggests that the co-evolved controllers drive more carefully, seldom accelerating to top speeds, and take corners more conservatively. A similar but smaller difference in absolute solo-fitness seems to exist between the groups of co-evolved controllers, with controllers evolved more for absolute fitness performing better than controllers evolved more for relative fitness. The controllers within a group perform similarly, and the lower fitness comes from driving slower around the track rather than crashing into walls or losing direction.

The difference between controllers co-evolved with different fitness mixes becomes clearer when we measure performance in competition with other controllers from the same group, where controllers evolved mostly for absolute fitness generally get about half a lap farther than those evolved mostly for relative fitness. Behavioural analysis confirms that this is because the cars more often collide at the start of a trial, often forcing one or both of the cars to crash against the wall or spin around and lose track of which direction it is going. Often, the controllers evolved with low (0 or 0.25) proportions of absolute fitness actively look for trouble by trying to collide. (See figure 2).

There seems to be little consistency in evolved sensor setups, samples of which can be seen in figure 1 (wall sensors are blue; car sensors are pink; each car is travelling forwards in direction of the waypoints). We found one controller in the group evolved purely for relative fitness that had only wall sensors and no car sensors, and another one in the group evolved for purely absolute fitness! There is no obvious tendency towards fewer or more car sensors at either end of the fitness mix, and the data is too scarce to prove any more subtle tendency. When looking at all 50 controllers together, every controller has at least three wall sensors, and there is always at least one pointing mostly forward. On average, the cars have twice as many wall sensors as car sensors, and when car sensors are present, there seems to be at least one pointing mostly backward; overall, more car sensors point backward than forward.

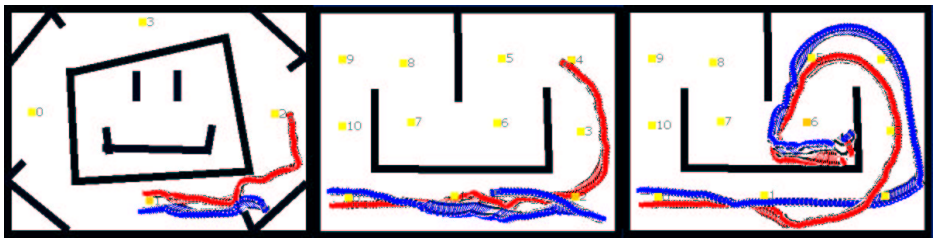

Fig. 2. Traces of the first 100 or so time-steps of three runs that included early collisions. From left to right: red car pushes blue car to collide with a wall; red car fools blue car to turn around and drive the track backwards; red and blue car collide several times along the course of half a lap, until they force each other into a corner and both get stuck. Note that some trials see both cars completing 700 time-steps driving in the right direction without getting stuck.

Fitness mix groups versus each other In order to find out how the controllers evolved with various fitness mixtures performed against each other, we tested all the five controller groups against each other. The slightly surprising results was that the groups performed on average equally well against each other, though with considerable intra-group variation. The absolute fitnesses of the controllers in these encounters were quite low, on average 0.96, which suggest

that all controllers are quite ill prepared to race against controllers from another fitness mix group.

3.3 Co-evolved versus solo-evolved controllers

The 50 controllers co-evolved with various fitness mixes in the section above were now tested against the 10 solo-evolved controllers from section 3.1. For each group, the ten co-evolved controllers competed for 10 trials with each of the 10 solo-evolved controllers. See table 2 for results.

Proportionabsolute	0.0	0.25	0.5	0.75	1.0
Co-evolved	1.41 (0.52)	0.99 (0.44)	1.32 (0.58)	1.23 (0.65)	1.41 (0.45)
Solo-evolved	1.68 (0.56)	1.95 (0.62)	1.74 (0.57)	1.38 (0.65)	1.51 (0.63)

Table 2. Co-evolved versus solo-evolved controllers.

Analysis Observe that there is a small (mostly) but consistent fitness advantage for the solo-evolved controllers over the co-evolved ones. (Both co-evolved and solo-evolved controllers performed significantly worse in these competitions than when tested in solo racing conditions.) The cause of this fitness difference is not completely obvious after looking at a large number of these competitions, but it appears that the solo-evolved controllers (which gain higher fitness than the co-evolved ones in solo trials) simply outrun the co-evolved controllers in many cases, and so avoid many of the collisions, and further corroborate the hypothesis that the controllers tend to be very specialized to compete against controllers similar to themselves. This could be seen either as a shortcoming of the evolutionary algorithm, or as the desired state of things; it could be argued that the co-evolved controllers should have strategies general enough to take on any opponent, or that a their more careful driving style should always make them slower than a solo-evolved controller.

3.4 Evolution with a static target

To investigate whether the tendency to specialization in co-evolved controllers could be used to create controllers that could out-compete the solo-evolved controllers from section 3.1, we modified a copy of the co-evolutionary algorithm to work with a static target. In this configuration, each controller is evaluated by racing three races against randomly selected controllers out of the ten solo-evolved controllers. It should be noted that this is not co-evolution at all, as the target controllers do not evolve. The car controlled by the target controller could instead be seen as an interactive feature of the environment.

The experiments we run with this configuration failed to generate any controllers with better fitness than the target controller. This was despite attempts

at evolving from scratch, starting from a general controller, or starting from a clone of the target controller, and using various mixtures of absolute and relative fitness. Our interpretation of this is that the solo-evolved controller drives the tracks as fast as can be done given its sensing and processing limitations, and that the same limitations hinder the co-evolved controllers from doing any better.

3.5 Human-competitiveness of evolved controllers

A random selection of controllers were tested by competing with a car controlled by one of the authors via the keyboard. It was found that the solo-evolved controllers were generally good contenders, driving at about the same skill level or slightly better than the author, as long as collisions were avoided. However, it was found to be quite easy to learn how to collide with the computer controlled car in such a way that it got stuck on a wall, and then continue to win the race. Most of the co-evolved controllers were pretty simple to beat by just accelerating hard at the beginning of a race and keep driving, as their slower driving wouldn't allow them to catch up.

4 Conclusion

Our main positive finding concerns the effects of changing the type of fitness function. A very clear effect was that controllers evolved more for relative fitness acted more aggressively, but covered less distance both when running solo and when competing with other controllers from the same population, than controllers evolved more for absolute fitness. We could not find any systematic difference between the sensor setups evolved with the various fitness mixtures, but observed a general tendency to point car sensors backwards rather than forward, and the opposite tendency for wall sensors - it seems to be more important to watch your back than to know whats happening in front of you.

A finding that is relevant to the overarching quest to scale up ECR and evolutionary robotics in general is that competitive ECR is a much more complex problem than solo ECR. This can be seen both from the drastic degradation of fitness when solo-evolved controllers are put in competitive environments, and from our great difficulty in evolving controllers that can reliably outperform the solo-evolved ones. It can also be seen from the total inability of all evolved controllers to backtrack upon a frontal collision with a wall, and the relatively poor ability of most evolved controllers to find the correct direction after having been spun around. This points to the need for more complex sensors and neural networks.

However we set up the evolutionary runs, they seem to suffer from overspecialization, where the controllers in a population only learn to race each other. This result is in broad agreement with what has been found in co-evolutionary predator-prey experiments[9][10]. So even though ECR allows us to explore a larger space of variants of competitive co-evolution, it seems that we at present

are stuck with the same basic obstacles to evolving generally good competitive behaviour.

4.1 Future research

One obvious extension of the controller architecture would be to add state to the presently stateless controller; this could be done by adding recurrent connections to the network. The controller could also be given the ability to grow or "complexify" itself as needed during the evolutionary run[11]. This could also be the case for the sensors; we believe that either more sensors of the present kind or some alternate sensor representation will be needed to give the controller the information needed to compete well.

The evolutionary algorithm could be enhanced with the addition of a "hall of fame", where the controllers of a generation compete not only against each other but also against the best controllers of previous generations[7][9][10]. It would be interesting to use evolutionary multi-objective optimization to evolve fronts of pareto-optimal tradeoffs between relative and absolute fitness.

References

1. Floreano, D., Kato, T., Marocco, D., Sauser, E.: Coevolution of active vision and feature selection. Biological Cybernetics **90** (2004) 218–228
2. Tanev, I., Joachimczak, M., Hemmi, H., Shimohara, K.: Evolution of the driving styles of anticipatory agent remotely operating a scaled model of racing car. In: Proceedings of the IEEE Congress on Evolutionary Computation. (2005) 1891–1898
3. Togelius, J., Lucas, S.M.: Evolving controllers for simulated car racing. In: Proceedings of the IEEE Congress on Evolutionary Computation. (2005) 1906–1913
4. Stanley, K.O., Kohl, N., Sherony, R., Miikkulainen, R.: Neuroevolution of an automobile crash warning system. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005). (2005)
5. Togelius, J., Lucas, S.M.: Evolving robust and specialized car racing skills. In: Proceedings of the IEEE Congress on Evolutionary Computation (to appear). (2006)
6. Hillis, W.D.: Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D **42** (1990) 228–234
7. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. Evolutionary Computation **5:1** (1996)
8. Dawkins, R., Krebs, J.R.: Arms races between and within species. Proceedings of the Royal Society of London , series B **205** (1979) 489–511
9. Nolfi, S., Floreano, D.: Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution? Artificial Life **4** (1998) 311–335
10. Bason, G., Bergfeldt, N., Ziemke, T.: Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments. Genetic Programming and Evolvable Machines **6** (2005) 25–51
11. Stanley, K., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. Journal of Artificial Intelligence Research **21** (2004) 63–100
12. Bourg, D.M.: Physics for Game Developers. O'Reilly (2002)
13. Monster, M.: Car physics for games. http://home.planet.nl/ monstrous/tutcar.html (2003)