UNIVERSITÉ LIBRE DE BRUXELLES
Faculty of Applied Science

Swarm Intelligence

Particle Swarm Optimization

Alexandre Balon-Perin

Marco A. Montes De Oca

Academic Year 2010 - 2011

Contents

Chapter 1

Introduction

Driving a car is not an easy task, any newcomer in this beautiful world shakes in fear when holding the steering wheel for the first time. That is understandable when one knows what is required to drive a car. Many factors have to be taken into account. The driver must pay attention to the surrounding, the tachometer, the speedometer. He must change gears at the appropriate moment, pay attention to its environment, etc. Thus, a car driver needs a fair amount of concentration and is constantly trying to adapt to the changing environment around the car. This changes are an attempt to reach the best parameters of the car in a given situation. This could be seen as an optimization process where the driver tries to attain the global optimum at each moment. In this report, the implementation of an agent learning how to drive is described. The project has been developed using the tools and code provided by the Simulated Car Racing Championship and the application TORCS (The Open Racing Car Simulator).

Goals of the project:

1. Design a car driving agent as a continuous optimization problem with the Particle Swarm Optimization algorithm.
2. One of the different variants of the PSO algorithm must be selected and parameters settings must also be achieved.
3. Integration of the algorithm in the simulator TORCS

The next section will introduce the Particle Swarm Optimization algorithm as well as its variants. Afterwards, an explanation on the differents parameters will be given and the choice of the selection will be clarified. The report will then go through several objective functions that have been implemented. Eventually, the agent will be compared to the "SimpleDriver", another agent with some basic features to drive a car.

Chapter 2

Particle Swarm Optimization Algorithm

2.1 Theory

Particle Swarm Optimization is a population-based algorithm. The general idea of the algorithm is to use a swarm of particles in which each particule is a possible solution of the problem. All particles are characterized by a position X which is the possible solution and a velocity V which allows the particle to move from its actual position to a new one as we will see later.

First, all the particles are initialized randomly. In this manner, they occupy a good portion of the space of possible solutions. Every iteration, the fitness of each particle is calculated. Two important values are kept in memory throughout the process: the best value of each particle so far and the best value of the entire particle swarm or a neighborhood depending on the topology of the swarm. This last value will hopefully converge to the global optimum.

The position of the particles are then evaluated according to the fitness function. If the particle's current position is better than its previous best position then the value of the position is updated. The velocity factor assures that the swarm is not going to fall into a local optimum.

In this project the PSO variant chosen is the FIPS (the Fully Informed Particle Swarm) which is very simple and gives good results. As Rui Mendes exposed in [Mendes Rui and Neves, 2004], the FIPS does not assume that the best neighbor so far is the only important one. If that was the case an important part of the search space could be totally forgotten.

The position and velocity are calculated as follow

$$v_i^{t+1} = \chi[\![v_i^t + \phi_k U_k^t(\sum_{p_k \in N_i}(pb_k^t)/neigbors) - x_i^t]\!]$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Where
$v_i^t and v_i^{t+1}$are the velocities of the particle i at iteration t and *t+1*
$x_i^t and x_i^{t+1}$ are the position of the particle i at iteration t and *t+1*
phi_k is a constant called the acceleration
χ is another constant called the constriction
pb_k^t is the best position of the neighbor k at iteration t
$\sum_{p_k \in N_i}(pb_k^t)/neigbors$ is the average over all the best positions of the neighbors

The constriction and the acceleration have respectively the values 0.729 and 2.05 has recommended in [Mendes Rui and Neves, 2004].

Algorithm 1: Particle Swarm Optimization

```
input  : The set of data needed to calculate the objective function
output: an array of the best parameters obtained so far
for $i \leftarrow 1$ to $PARTICLES$ do
    particle[[$i$]].currentSolution $\leftarrow$ $objective\_function(i)$
end
for $i \leftarrow 1$ to $PARTICLES$ do
    //FIPS(i)
    getNeighborsAverageBestPosition()
    calculateV()
    calculateX()
    isInRange()
    improveParticlePosition()
end
```

2.2 Application to the problem

This section will clarify the PSO algorithm's integration within the client software. The main part here is the SimpleDriver.cpp file that will be customized in order to use the algorithm. In this file all the function needed to actually drive the car are present. For example, getGear(CarState &cs) will make the modification on the gear according to the car' state. When all the parameters have been modified to take account of the present situation, the client driver send every time step an array with all the parameters required to drive the car.

Apart from the parameters that change during the race such as the gear or the brake, there are many other parameters which are characteristic of the car and stay constant during the race. The PSO algorithm is going to be used to optimize these kind of parameters. A swarm of particle is initialize with a values for each of the parameters that will be optimized. This is represented as an array. Each particle is a possible configuration of the car.

The learning process starts when the first particle takes "control of the car" (the parameters of this particle are injected in the car) the car is driven for an entire lap. Then, the game is restarted and another particle becomes the driver. When the whole set of particles has driven the car at least once the improvement part takes place. The position of the particles are than updated according to the fitness function. The previous steps can be iterate many times in order to reach the global optimum.

Chapter 3

Fitness Function

This fitness function is calculated as follow

$$fitness = lapTime_k^t + \alpha * damage_k^t$$

Where $lapTime_k^t$ is the time used to finish one lap by the particle k at iteration t $damage_k^t$ are the damage of the car after one lap driven by the particule k at iteration t α is a coefficient to modulate the importance of the damage

Three fitness functions were tried:

1. $\alpha = 0$
2. $\alpha = 0.3$
3. $\alpha = 1$

The first one gave good time results but the car was totally destroid with an average damage of 6000. The third function allowed the car to finish the lap with almost no damage but the laptime was sometimes worse than the SimpleDriver. Eventually, the second function was the one which gave the best global results with few damage and an acceptable time for the lap.

Chapter 4

Parameter Selection

The parameters fixed for one particle can be classified in five groups:

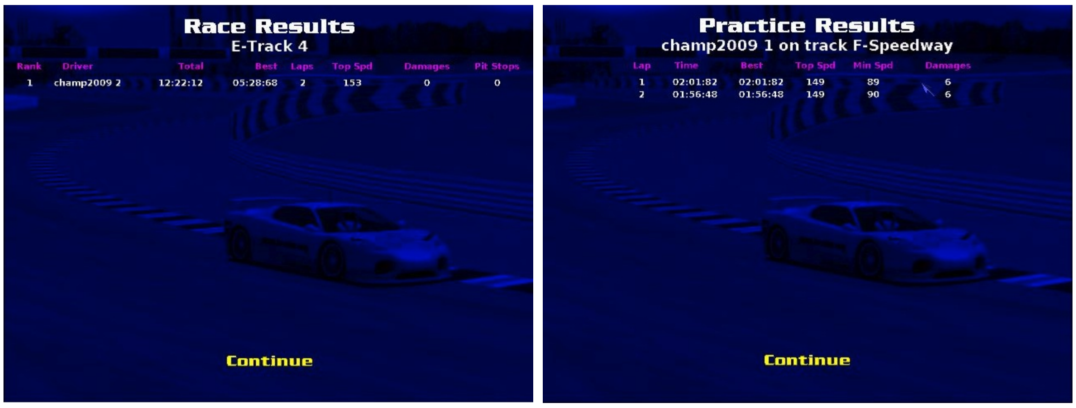
- *GEAR* concern the rpm at which the gear must be changed
- *STUCK* concern the angle and number of step for which the car is assumed to be stuck
- *ACCELERATION/BRAKE* are constants used to know when to accelerate or to brake
- *STEERING* concern the steering command when the car goes at high speed
- *ABSFILTER* concern the release of the ABS when the car brakes

Within these five groups five parameters have been chosen to be optimized by the PSO algorithm. The choice of these parameters was done in order to maximize the control of the algorithm on the motion of the car. A change in one of the parameters chosen has a big influence on the resulting drive. The range of the parameters are chosen empirically, they are more or less centered on the values of the SimpleDriver.

1. **maxSpeedDist** is the minimum distance from track border to drive at maximum speed (range: [25 ; 75])
2. **absSlip** is the minimum slip to prevent abs (range: [1 ; 3])
3. **wheelSensitivityCoef** is the coefficient to reduce steering command at high speed (range: [0.5 ; 1.5])
4. **steerLock** is the angle associated to full steer command (range: [0.5 ; 1.5])
5. **steerSensitivityOffset** is the minimum speed to reduce steering command (range: [50 ; 100])

Chapter 5

Results



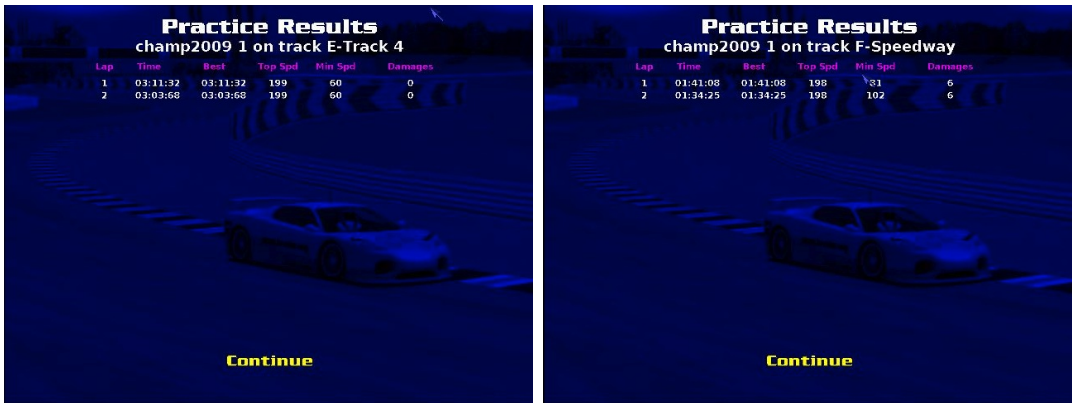
Race Results
E-Track 4

Rank	Driver	Total	Best	Laps	Top Spd	Damages	Pit Stops
1	champ2009 2	12:22:12	05:28:68	2	153	0	0

Practice Results
champ2009 1 on track F-Speedway

Lap	Time	Best	Top Spd	Min Spd	Damages
1	02:01:82	02:01:82	149	89	6
2	01:56:48	01:56:48	149	90	6

Figure 5.1: SimpleDriver



Practice Results
champ2009 1 on track E-Track 4

Lap	Time	Best	Top Spd	Min Spd	Damages
1	03:11:32	03:11:32	199	60	0
2	03:03:68	03:03:68	199	60	0

Practice Results
champ2009 1 on track F-Speedway

Lap	Time	Best	Top Spd	Min Spd	Damages
1	01:41:08	01:41:08	198	81	6
2	01:34:25	01:34:25	198	102	6

Figure 5.2: PSODriver

Bibliography

[Mendes Rui and Neves, 2004] Mendes Rui, J. K. and Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions of evolutionary computation*, 1(1).