4.
Trajectory Determination: Evolutionary Optimization

Determining the minimal time trajectory is not a simple task, especially when considering all the variables involved. The growing demand on autonomous driving systems stimulates research towards new technologies.

Iterative optimization methods have been applied to solve the path determination problem. As a first attempt a classical method based on the gradient descent was used by Carrera in [11], also implemented with the Matlab® Optimization Toolbox [16]. This optimization procedure is organized in a block diagram as shown in Figure 4.1.

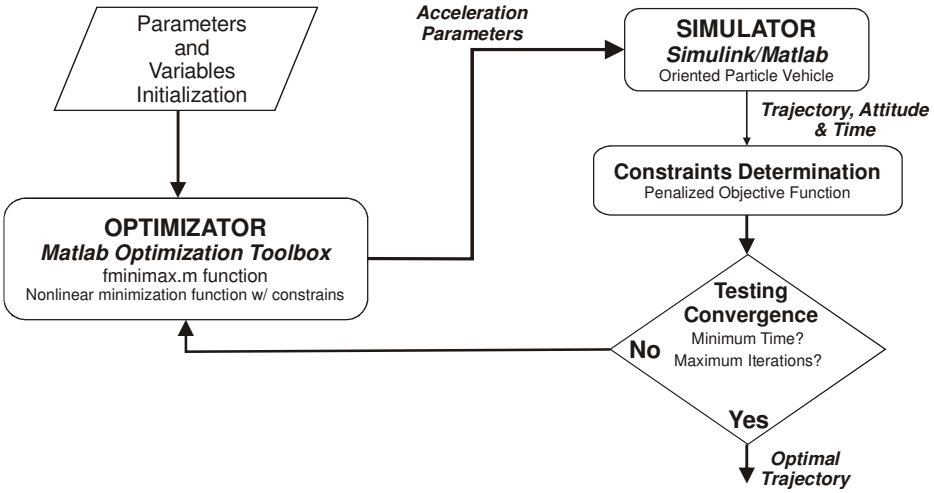

Figure 4.1 – Classic Optimization Block Diagram.

This chapter proposes, analyzes and describes the implementation of an optimization technique based on genetic algorithms, also known as evolutionary computation. In a few words, its approach consists of evolving a set of acceleration profiles which takes the car through the minimum time trajectory of a previously specified track.

4.1. Presentation and Description

Similarly to other computational intelligence techniques, genetic algorithms are inspired on nature. The problem's solution is treated as an

individual among a population of other possible solutions. Inspired on natural selection, the technique consists of analyzing each individual's capacity of attending the natural requisites to reproduce and pass its genetic characteristics to the next generations.

In nature, the better the physical conditions one has, the better the chances of survival. Moreover, the more one reproduces, the higher the probability of passing on the genetic material. This fitness of the natural individual is translated to the solution population as the value of a function to be optimized. This genetic material is a codification of the possible solution relevant characteristics, and those should be changed by reproduction and recombination in order to find a better solution in the generation changes, Figure 4.2 shows the conceptual block diagram.

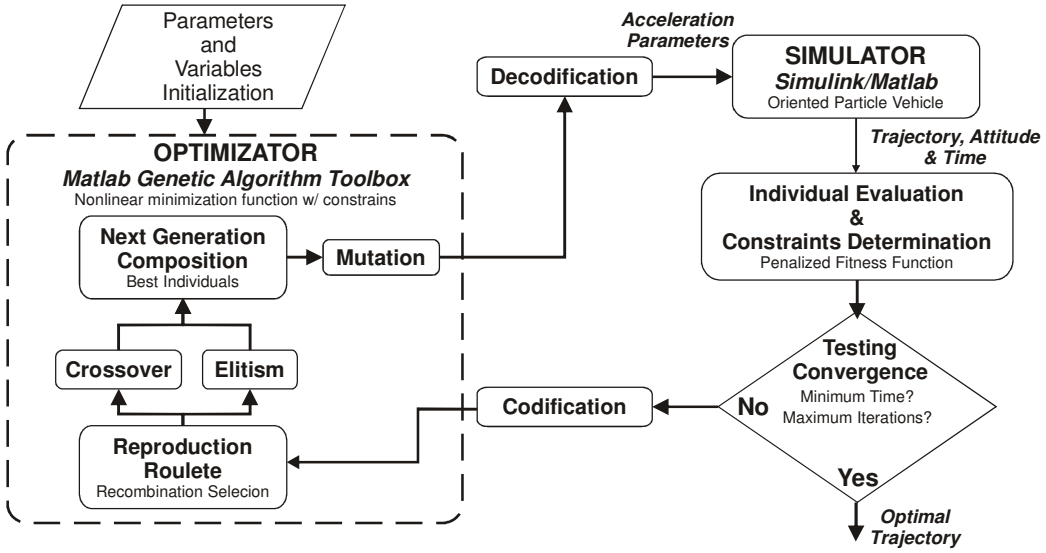

Figure 4.2 – Genetic Algorithm Optimization Block Diagram.

4.1.1. Initial Estimative Determination: Center Line Trajectory

As any heuristic optimization method, genetic algorithms can achieve better results if an initial estimation is given. A good initial estimation, called seeding, is important for convergence, especially in problems with a vast domain.

Considering the car trajectory, a natural seed is the center line trajectory, since the random acceleration profiles would hardly take the vehicle to the track end correctly. Acceleration profiles that result in the center line trajectory can be easily obtained by analytical calculations of the car's speed (V_{Max}) on a curve stretch i, the curve radius (R) and maximum lateral acceleration (a_{LMax}). The

profile parameter $\bar{a}_{NO}$ for this track stretch i is shown in Equation (4.1). Figure 4.3 shows an example of a center line trajectory and Figure 4.4 shows the acceleration profiles for this trajectory.

$$V_{Max}(i) = \sqrt{R(i) \cdot a_{LMax}}$$
$$\bar{a}_{N0}(i) = \frac{V_{Max}(i)^2}{R(i) \cdot g} \tag{4.1}$$

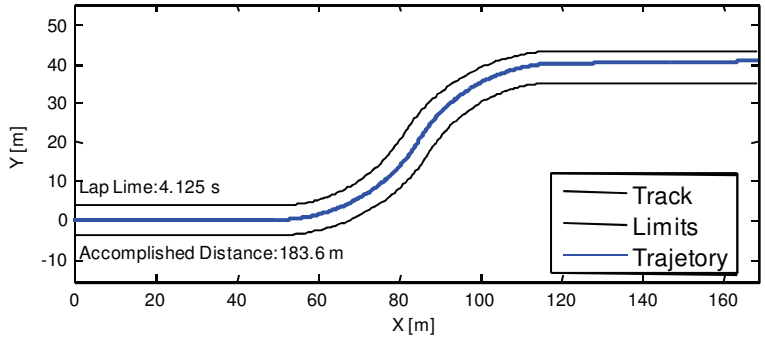

Figure 4.3 – Center Line Trajectory.

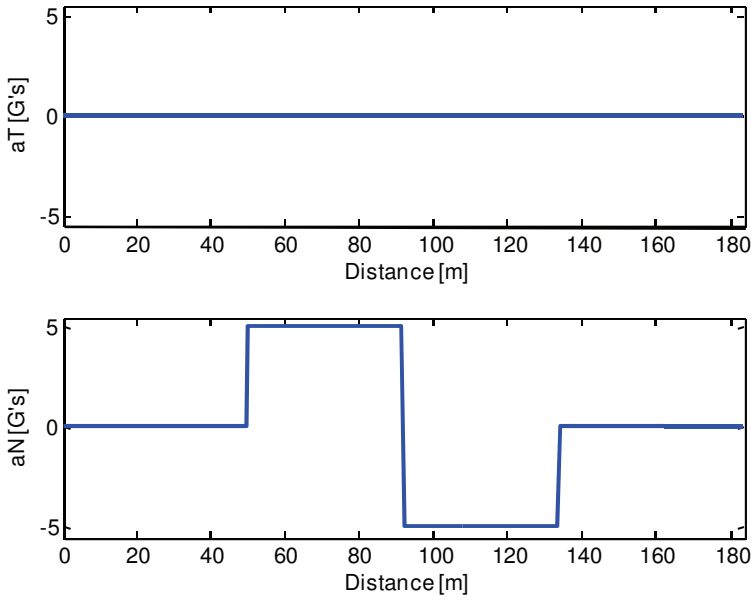

Figure 4.4 – Correspondent Acceleration Profiles.

4.1.2. Chromosome Codification

As mentioned before, although an optimal trajectory is the main goal here, optimization variables are actually the acceleration profiles. Consequently,

chromosomes should contain any relevant information on the parameterization of those profiles, as seen in Figure 4.5 and detailed in Chapter 2.

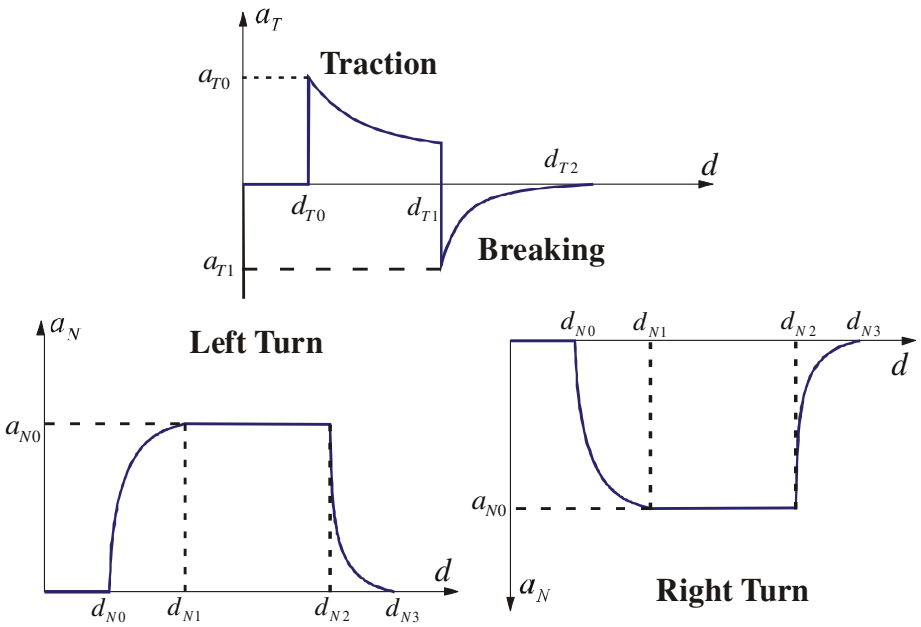

Figure 4.5 – Acceleration Profiles.

Codification of the acceleration profiles into a chromosome can be seen in Figure 4.6, where the 2-D array is lined up as a 1-D vector for an easier implementation of genetic operators.

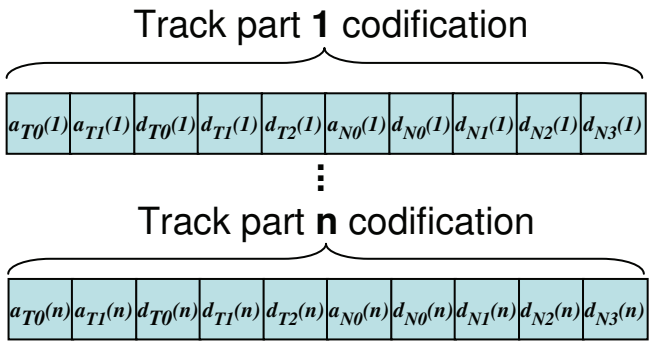

Figure 4.6 – Chromosome Codification.

4.1.3. Genetic Operators Definition

Considering the chromosome as an array with different parameters values for each track stretch, each operator must preserve the domain aspects within the array sectors. Figure 4.7 shows those sectors for a single track stretch with its respective features. The parameter $l_{TP}(i)$ is the length of the i^{th} track stretch.

Track part **i** codification

		$a_{T0}(i)$	$a_{T1}(i)$	$d_{T0}(i)$	$d_{T1}(i)$	$d_{T2}(i)$	$a_{N0}(i)$	$d_{N0}(i)$	$d_{N1}(i)$	$d_{N2}(i)$	$d_{N3}(i)$
Minimum Value	**Actual**	0	-5	0	0	0	-5	0	0	0	0
	Normalized	0	-1	0	0	0	-1	0	0	0	0
Maximum Value	**Actual**	1.5	0	$l_{TP}(i)$	$l_{TP}(i)$	$l_{TP}(i)$	5	$l_{TP}(i)$	$l_{TP}(i)$	$l_{TP}(i)$	$l_{TP}(i)$
	Normalized	1	0	1	1	1	1	1	1	1	1

Figure 4.7 – Chromosome Sectors' domains.

4.1.3.1. Reproduction Selection

The reproduction selection operator considered here is the fitness roulette wheel. It introduces some random factor also existing in the natural process, but, instead of having equal probabilities, the roulette slices are proportional to the fitness function value of each individual in a generation. Therefore, the fittest solution has more chance to reproduce and consequently pass on its genetic material. Figure 4.8 illustrates the concept of fitness roulette wheel, where percentages represent the chances of each individual to be selected for reproduction.

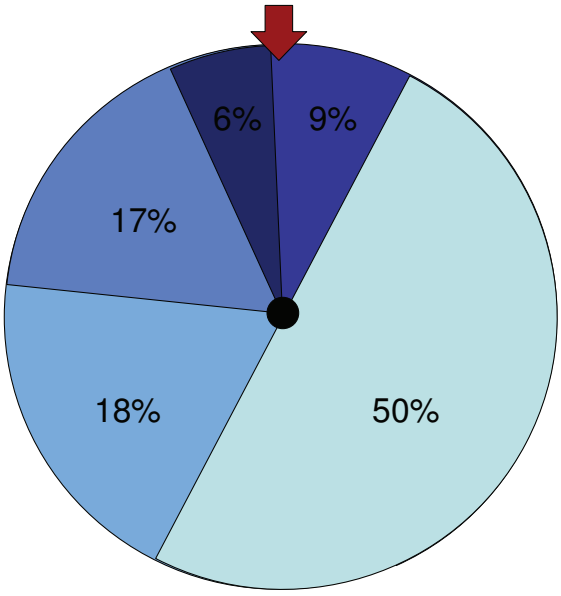

Figure 4.8 – Fitness Roulette Wheel.

4.1.3.2. Crossover

The natural process consists of joining half of the genetic material from each parent randomly divided to generate the descendant's DNA. This composition is like a mix of genetic code parts. In the computational recombination, several methods are used to mime the same effect, concerning chromosome structure and data format.

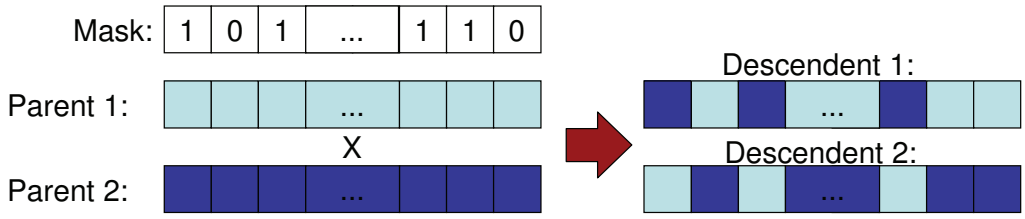

Figure 4.9 – Scattered Crossover Algorithm.

4.1.3.3. Mutation

The natural mutations happen randomly and introduce unexpected changes to one or more genetic characteristics of an individual. Mutation is important for recovering genetic variability in a saturated population. Likewise, the computational mutation method has the same purpose. Its algorithm randomly chooses some of the generated individuals to change some part of their chromosome also randomly chosen. The mutation rate m_R controls the percentage of the population submitted to mutation.

4.1.4. Restrictions and Multi-Objective Fitness Function

In the presented problem, the variable to be minimized is the vehicle lap time from the start to the end of the defined path. Each individual should follow some accelerations constraints in order to be faithful to the vehicle kinematics and to guarantee that the obtained trajectory can be followed. Track limits are clear constraints that must also be respected.

In addition, each car has a set of acceleration limits above which it slides. Those critical lateral and longitudinal accelerations are defined by the car's Friction Ellipse, presented in [9] and detailed in Chapter 2.

Classical optimization methods usually define existing restrictions as inequalities. Methods based on Genetic Algorithms usually present faster convergence by modeling restrictions as penalties in the final value of the fitness

functions. This approach can also be considered as a multi-objective fitness function problem.

In the proposed representation, the first part of the fitness function is a dynamic simulation of the vehicle digital model. Since a great number of repetitive dynamic simulations are calculated throughout the evolution process, the oriented particle vehicle model detailed in Chapter 2 is used..

The trajectory and speed information compose the fitness function calculation. The distance achieved before the car leaves the track limits is called d_{Acc} and the entire track distance is d_T. The percentage of the track distance travelled along a specific trajectory is called $d_{\%}$.

Another relevant constraint also obtained through simulation is the achieved acceleration in relation to the Friction Ellipse. Once the car model exceeds the Ellipse's acceleration limits, P_{FE} assumes a true boolean value, which lessens the fitness of that individual.

The weights used in the fitness function had to be tuned through several tests. The fitness function to be minimized is defined in Equation (4.2), where V is the vehicle's longitudinal speed, a_N is the lateral acceleration array and n is its length.

$$d_{\%} = \frac{d_{Acc}}{d_T}$$

$$P_{FE} = \begin{cases} 1 & ; \text{Friction Ellipse Respected} \\ 0 & ; \text{Friction Ellipse Violated} \end{cases}$$

$$f = t_{Lap} \cdot 20^{\max(0;(1-d_{\%}))} + 1000 \cdot P_{FE} - \frac{\sum_{i=1}^{n} |a_N(i)|}{n} - \frac{\max(V)}{10} \tag{4.2}$$

The determination of the best values for each genetic algorithm parameters depends especially on the chromosome model chosen. However, it is also affected by the characteristics of the system being optimized. All parameters used in the genetic optimization are shown in Table 4.1.

Table 4.1 – Genetic Optimization Initial Parameters.

Population	1000	-
Generations	300	-
Selection	Normalized Roulette	-
Reproduction	Scattered Crossover	Uniform Rate: 80%
Mutation	Random Number	Uniform Rate: 50%
Steady-State	Elitism	GAP: 10%
Initial Population	Seed + Random Variation joined with Intact Seed	

4.2. Validation Tests

The validation tests for the developed genetic algorithm consist of running the optimization method for a small track stretch. In order to create a representative and yet simple track for the test, an "S" curve preceded and followed by straight lines was defined. This small route creates both right and left acceleration profiles, allowing a detailed analysis of the optimization output trajectory regarding the center line seed trajectory.

Therefore, with the aim of correctly evaluating the output gain, the seed trajectory is shown in Figure 4.10.

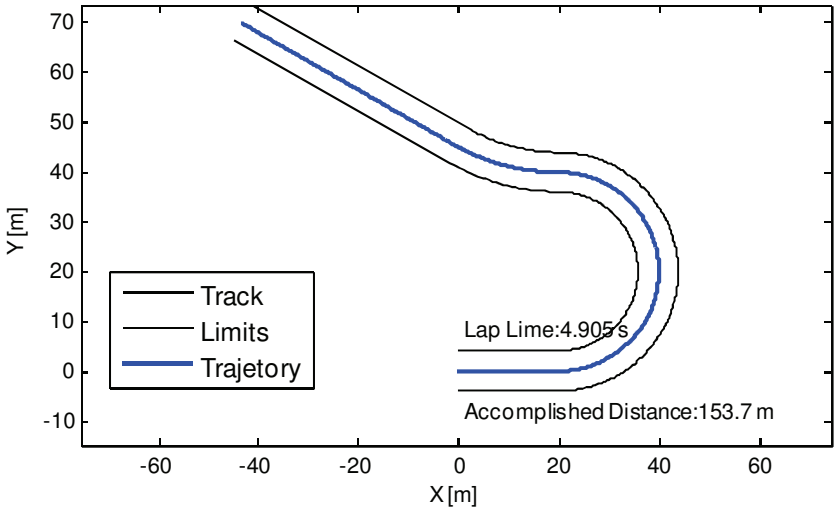

Figure 4.10 – Center Line Seed Trajectory.

A randomly created population is created from this seed – much of it with higher lap times or smaller accomplished distances. The complete optimization process is as computationally expensive as any other iterative method. In a non-

parallel processing unit, a track of a hundred meters and four track stretches implies about two hours of computational processing. After some tests, it is possible to tune the crossover and mutation rates for a faster convergence.

Some results can be seen in Figure 4.11. Besides a lap time reduction of about 20%, the resulting trajectory indicates atendency of chasing the curve's tangents. Resulting acceleration profiles are shown in Figure 4.12

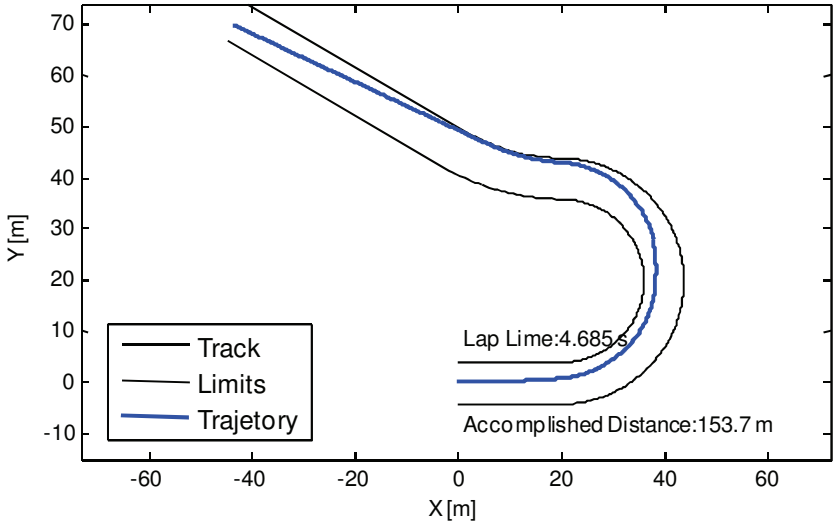

Figure 4.11 – GA Optimization 1st Validation Test: Optimized Trajectory.

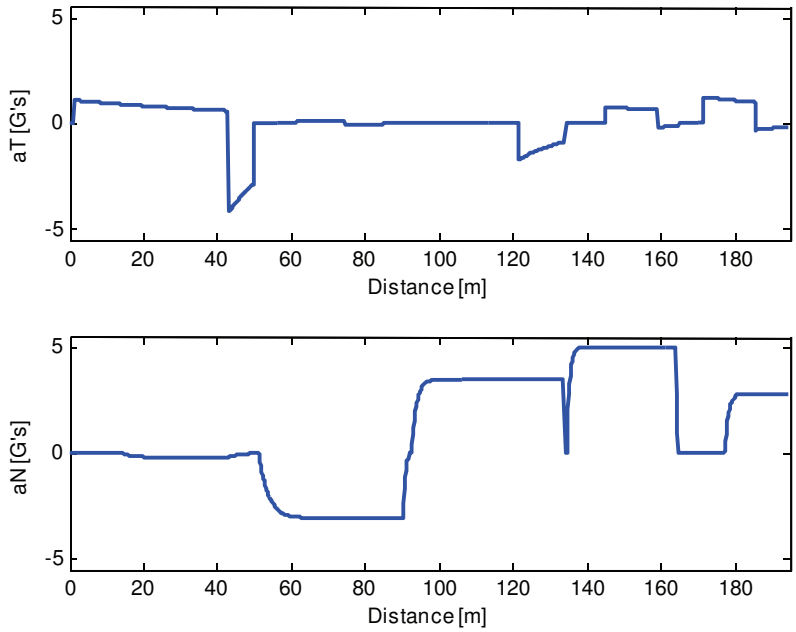

Figure 4.12 – GA Optimization 1st Validation Test: Acceleration profiles

In the applications' chapter, other analyses on the optimized trajectories will be detailed.