POLITECNICO DI TORINO

Master Degree in Mechanical Engineering

Master's Degree Thesis

Development of an adaptive real-time control strategy for plug-in hybrid electric vehicles

Academic Supervisor:
Prof. Daniela Anna Misul

Graduate:
Alessia Musa

Academic Year 2018-2019

Abstract

This study is focused on the development of an adaptive real time control strategy for (plug-in) hybrid electric vehicles. This controller aims to select in real time the optimal control strategy in terms of power flow and gear number. More in details, it compares an unknown driving mission with a set of known driving cycles for which optimal set of rules is previously identified. At each instant, once the pattern recognition is performed, the set of rules is applied to the unknown driving cycle. The set of rules is identified by means of a pre-existing Clustering Optimization and rule extraction tool (Core). The Core tool associates a specific rule, in terms of power flow and gear number, to each cluster by selecting the most frequent action adopted by the dynamic programming.

Acknowledgements

I would like to express my gratitude to my academic supervisor Prof. Daniela Anna Misul for her encouragement and support. I would like to thank to Dr. Claudio Maino for his assistance, patience and his helpful advices. My sincere thank goes to my friends Irene, Giusy and Armando for the immense support they give me during these years and for their continuous encouragement. Special thanks also to Matteo and Mattia who shared this journey with me. Finally, I must express my profound gratitude to my parents and sisters.

Contents

Acknowledgements II

1 Introduction 1
- 1.1 Regulations . . . 1
- 1.2 Pollutant emissions and greenhouse gas . . . 2
- 1.3 Alternatives to conventional vehicle . . . 3

2 Global control strategy optimization 8
- 2.1 Dynamic programming . . . 8
- 2.2 Genetic algorithms . . . 11
  - 2.2.1 History and brief introduction . . . 11
  - 2.2.2 Population . . . 13
  - 2.2.3 Fitness function . . . 14
  - 2.2.4 Selection . . . 14
  - 2.2.5 Crossover . . . 15
  - 2.2.6 Mutation . . . 16
  - 2.2.7 Termination condition . . . 17
  - 2.2.8 Comments . . . 18

3 Model 19
- 3.1 Time grid . . . 19
- 3.2 Engine . . . 21
- 3.3 Electric machine . . . 22
- 3.4 Battery . . . 23
- 3.5 Torque coupling device . . . 23
- 3.6 Pre-processing phase . . . 24
- 3.7 Optimization phase . . . 29

4 Control strategy 31
- 4.1 CORE TOOL . . . 31
  - 4.1.1 Training . . . 31

4.1.2 Validation phase . . . 37
4.1.3 Multiple training . . . 37
4.1.4 Modification to termination condition . . . 38
4.1.5 Rule extraction . . . 38
4.1.6 Modifications to rule extraction . . . 39
4.1.7 Sensitivity on the number of individuals . . . 42
4.2 ACORE . . . 43
4.2.1 Definition of training driving cycles' set . . . 43
4.2.2 Current driving cycle evaluation . . . 49
4.2.3 Rule assignment . . . 51

5 Results 55
5.1 Core results . . . 55
5.1.1 Population size . . . 55
5.1.2 Rule extraction methods comparison on M class . . . 56
5.1.3 Sensitivity analyses . . . 58
5.1.4 Different discretization . . . 61
5.1.5 Sensitivity on crossover factor . . . 63
5.2 Acore results . . . 63

6 Conclusions and future works 68
6.1 Conclusions . . . 68
6.2 Future works . . . 69

Bibliography 70

List of Tables

4.1 Values of main parameters of GA 33
4.2 NEDC-WLTP comparison [19], [20] 46
4.3 Definition of A Solution's number of segments 49
4.4 Definition of B Solution's number of segments 49
4.5 Simulation time comparison 49
4.6 Simulation time comparison 51

List of Figures

1.1 Series HEV 5
1.2 Single shaft layout 6
1.3 Double shaft layout 6
1.4 Double drive layout 6
1.5 HEV classification based on e-machine position 7
2.1 Dynamic programming [8] 11
2.2 Genetic algorithm's terminology 12
2.3 Roulette wheel selection [15] 14
2.4 Tournament selection [15] 15
2.5 One point crossover [15] 17
2.6 Multi point crossover[15] 17
2.7 Uniform crossover [15] 17
2.8 whole arithmetic recombination [15] 17
3.1 Torque coupling device [4] 23
3.2 Battery model [4] 28
4.1 Training flow chart 32
4.2 Parent combination method, 1000 individuals, 10 generations 35
4.3 Parent selection method, 1000 individuals, 10 generations 36
4.4 Hybrid method, 1000 individuals, 10 generations 36
4.5 Mode 39
4.6 Comparison between the mode function and the modified mode function 40
4.7 Comparison between the mode function and the datasample function 41
4.8 Acore flow chart 43
4.9 Ardc driving cycle 44
4.10 Etc driving cycle 44
4.11 Whvc driving cycle 45
4.12 Nedc driving cycle 45
4.13 Wltp driving cycle 46
4.14 Etc cycle recognition 50
4.15 Ardc cycle recognition 51
4.16 Wltp cycle recognition 52

4.17 Wltp cycle recognition 52
4.18 Zoom on the Nedc cycle 53
4.19 Zoom on the Wltp cycle 53
4.20 Comparison between different time steps applied on Etc cycle 54
4.21 Comparison between different time steps applied on Wltp cycle . . . 54
5.1 Class comparison 56
5.2 Class comparison 57
5.3 Choices comparison on medium class 57
5.4 Choices comparison on medium class 58
5.5 250 individuals, 160 generations, Frequency default method 59
5.6 250 individuals, 160 generations, Alternated Frequency method 59
5.7 700 individuals, 50 generations, Frequency default method 60
5.8 700 individuals, 50 generations, Alternated frequency method 60
5.9 1500 individuals, 30generations, Frequency default method 61
5.10 1500 individuals, 30generations, Alternated Frequency method 61
5.11 Different discretization - M class 62
5.12 Different discretization [19 21 3]- M class 62
5.13 Crossover factors comparison 63
5.14 Clust 2 cycle recognition 64
5.15 Clust 2 cycle recognition 64
5.16 Clust 2 zoom 65
5.17 Etc zoom 65
5.18 Clust 8 cycle recognition 66
5.19 Clust 8 cycle recognition 66
5.20 Clust 8 zoom 67
5.21 Zoom on Nedc cycle 67

Chapter 1

Introduction

In the recent years it has been observed a tightening in the regulations of CO_2 and pollutant emissions in the transport sector. Europe's answer to emission reduction challenge is low- emission mobility to improve local air quality, lower the green house gas emissions and find alternatives to the petroleum solution.

1.1 Regulations

The first CO_2 emission target for passenger car was signed in 1998/99 through a voluntary agreements between three manufacturer associations (ACEA,JAMA and KAMA) and the European Commission. The imposed target was of 140g/km of CO_2 emissions weighted on the manufacturer's fleet to be reached by 2008. From the 2004 the reduction program became mandatory because the manufacturer stopped respecting the signed agreements. There are two separate sets of regulations for passenger cars and light commercial vehicles. For passenger cars, the regulation adopted in April 2009 fixed a fleet-average CO_2 emission target of 130 g/km to be reached by the 2015 while the long-term target is 95 g/km to be reached from 2020. For light commercial vehicles, the regulation imposed in May 2012 fixed a fleet-average CO_2 emission target of 175 g/km to be phased-in by 2017 while the long-term target is 147 g/km.
Until September 2017 emissions are tested over the New European driving Cycle (NEDC). However, this cycle did not seem to be representative of the real driving conditions and for this reason it has been replaced by the Wltp cycle.

1.2 Pollutant emissions and greenhouse gas

A greenhouse gas is able to absorb infrared radiation emitted from the surface of the Earth and then to re-radiate it back to the Earth's surface, contributing in this way to the greenhouse effect. The most important greenhouse gases are Carbon dioxide and methane.
The *Global warming potential* GWP is the index used to measure the green house gas capability to traps gas in the atmosphere.It is evaluated over a specific time horizon by using the radiative forcing i.e an index that measure the influence a chemical species has in modifying the balance of incoming/outgoing energy in the atmosphere. The GWP compares the radiative forcing of a certain gas and the radiative forcing of reference species (CO_2).

$$GWP_i = \frac{\int_0^{TH} RF_i(t)dt}{\int_0^{TH} RF_{ref}(t)dt} \tag{1.1}$$

where RF_i is the radiative forcing of the gas under investigation while RF_{red} is the radiative forcing of carbon dioxide. By definition, CO_2 has a GWP equal to one. The *Carbon dioxide equivalent* CDE is computed as follows:

$$CDE = \sum_i (GWP_i \cdot M_i) \tag{1.2}$$

where M_i is the mass of the gas under investigation.
Pollutant emissions divide in primary and secondary pollutants. Primary air pollutants coming from the combustion system are:

- CO, HC, NO_x, PM due to the incomplete combustion process;
- SO_X and metal compounds due to additives and / or chemical species in the fuel;
- Coming from the lubricant oil or materials due to machine pats' wear.

The primary pollutants, if exposed to the action of the sun or to other components with which they react, form secondary pollutants:

- Photochemical smog;
- acid rains;
- ground-level ozone.

1.3 Alternatives to conventional vehicle

The solutions explored are different. The electric vehicles present some advantages over the conventional vehicle such as zero CO_2 tank to wheel emissions and high energy efficiency. However they show several drawbacks:

- limited driving range;
- limited deployment of recharging infrastructures;
- added weight due to battery,
- costs.

Hybrid vehicles can take advantage of two energy sources, a thermal engine and one or several electric machines [2], [3]. Electric machines can act as motor or generator: in the first case they draw energy coming from the batteries to accelerate the vehicle while in the second case they recover energy from the vehicle in order to recharge the batteries. Hybrid electric vehicles combine the advantages of either battery-power electric vehicles and conventional ones but also compensate their shortcomings.The main advantages are:

1. Regenerative braking: the generator is able to recover and store the kinetic energy when the vehicle is braking;
2. Idling reduction by turning-off the engine at stops and lower speed conditions;
3. Ice downsizing/downspeed thanks to the electric machine assistance: the vehicle can be equipped with a smaller engine without compromising its performance. This is a great merit especially considering that, with reference to the engine operating map, usually the conventional vehicles operate at partial load: this means that typically engines are oversized if compared with the actual power requirements.
4. Auxiliaries and accessories can be powered electrically;
5. ICE efficiency improvement: the electric machine assists the engine when it operates in sub-optimal conditions;
6. more control over the engine's operating point and transience: this results in benefits for emissions and drivability.

The main drawbacks are:

1. higher costs compared with conventional vehicles;

2. added weight due to secondary power source and energy storage system;
3. complex control system is required;
4. components high costs.

There are different levels of hybridizations among hybrids including micro, mild, full and plug-in. Full, plug-in and most mild have the function of regenerative braking.

1. **Micro** The vehicle is equipped with a starter and employs the start & stop technology. At complete stops, the engine is turned off and restarts when the driver releases the brake pedal.

2. **Mild** In this configuration the alternator is equipped with stronger electric components than the Micro HEV. The electric motor, the starter and the battery pack are larger than those used in the micro HEV.

3. **Full** The electric components (battery pack, electric motor and starter) are the same used in the mild HEV but larger in size. However, it uses a more complicated control system in order to optimize the efficiency.

4. **Plug-in** The plug-in type allows the external charging of the battery and are designed for longer distances and the battery pack used is larger than the previous solution.

Series configuration A hybrid with a series architecture is characterized by the fact that the propulsion to the wheels is guaranteed only by the electric motor while the internal combustion engine is used only to produce electricity through the connection with the generator. In Figure 1.1, a typical series architecture is shown.
The power required by the vehicle can be supplied by the battery pack or by generator, since both can supply electricity directly to the electric motor. The engine can be turned off or on depending on the battery state of charge. When the ice is used as the primary power source, this systems shows significant inefficiencies because the output of the engine must be converted into electricity. This results in poor efficiencies especially during highway driving. Moreover, in order to satisfy the minimum acceleration requirements when the battery charge is completely depleted, the engine must be big enough. However, a large and heavy storage system increases costs and reduces the vehicle performance.This system shows very good efficiency

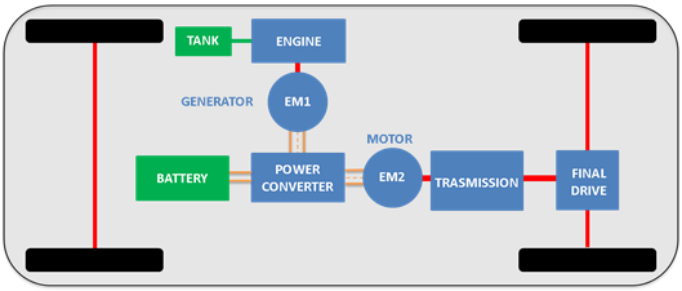


Figure 1.1: Series HEV

during start and stop phases: in fact it is able to minimize inefficient engine operation conditions and it is also able to maximize regenerative braking efficiency. This system is characterized by very low pollutant emissions.
Parallel configuration In the parallel configuration both the engine and the electric machine can supply their power to the driven wheel. This configuration presents several points in its favor:

1. the traction motor is smaller that the series configuration;
2. the generator is not required;
3. frequent power multiconversions are not necessary

The architecture of a parallel hybrid vehicle usually consists of a single electric machine, which can be installed on the front or rear axle based on the desired configuration, and an internal combustion engine.The ice constitutes the main energy source. One of the main features of this configuration is the high flexibility, in fact it enables different operative modes of the vehicle: pure thermal, pure electric, hybrid mode. However it requires a complex control strategy for the energy management. The classification of the parallel vehicle can be made considering the position of the electric machine or the type of connection.The possible layouts are:

- Single shaft 1.2;
- Double shaft (the electric machine is coupled with a devoted transmission) (Figure 1.3);
- Double drive or through the road because the connection in between the engine and the electric machine is the road. In this configuration it is possible to have traction on both axle of the vehicle (Figure 1.4).

Considering the position of the electric machines (Figure 1.5), the possible alternatives are:

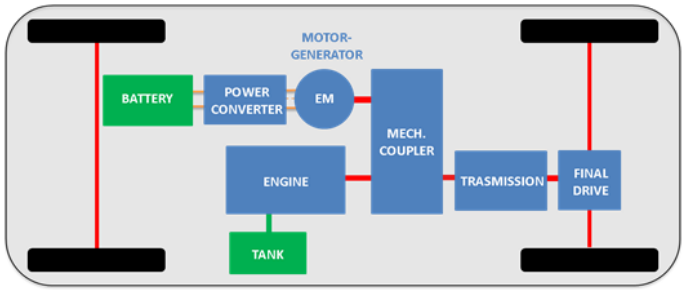


Figure 1.2: Single shaft layout

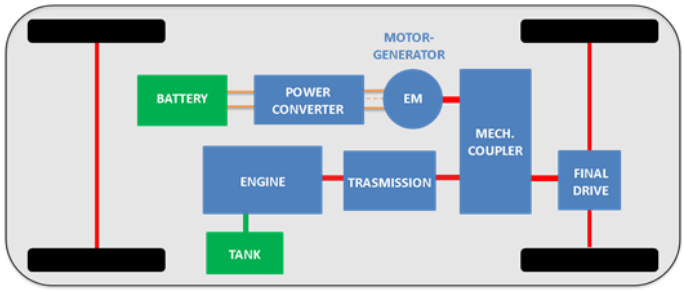


Figure 1.3: Double shaft layout

- P1: electric machines connected to the engine. If the electric machine is located in the front side, the regenerative braking efficiency results to be low because of the energy losses between the transmission and the e-machine.
- P2: e-machine between the engine and the transmission unit (it can be decoupled by means of a clutch enabling the pure electric functioning);
- P3: e-machine in between the transmission and the differential unit;
- P4: e-machine on the secondary axis while the engine is on the primary one.

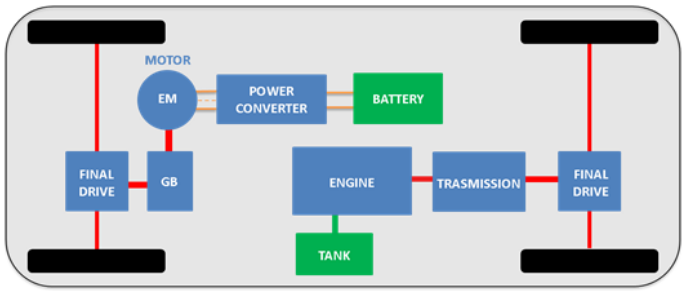


Figure 1.4: Double drive layout

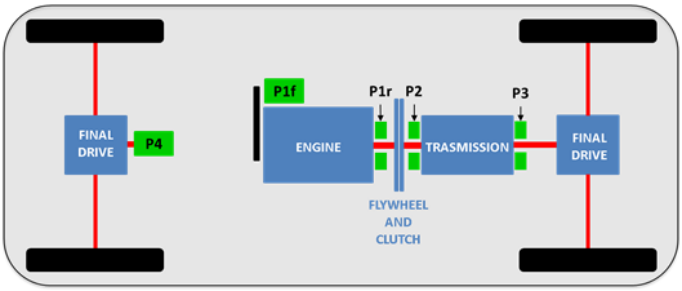


Figure 1.5: HEV classification based on e-machine position

Chapter 2

Global control strategy optimization

A global optimization technique has basically the capacity to search the global optimal control strategy by means of a numerical approach but it requires a priori knowledge of the problem in terms of driving cycle's features and characteristics. The final aim of this kind of optimizer is to recognize the "best path" starting from a given state and proceeding with the successive ones. The main drawbacks of a global optimization control strategy can be found in its complexity and in the high computational time required: for these reasons it can not be implemented real-time but it is an useful tool for the off-line analysis of the problem.

2.1 Dynamic programming

Dynamic programming was introduced by Richard Bellman in 1950 and it is applied in several fields, from economics to engineering applications. The term "dynamic" is used because the time has a great relevance in this process and because the order of actions may be decisive. In technical term, dynamic programming is a multi-stage decision process. In order to better understand, we can start from physical system represented at each time instant by a certain quantity, let us say a vector. As time passes, this system is submitted to modifications due to deterministic or stochastic reasons. A "decision process" is a process in which at any time instant it is possible to decide which type of transformation will be applied to the system. Multi-stage decision process refers to a sequence of allowable decisions. Dynamic programming founds its application in problems in which the objective or cost function can be divided into a series of stages where each stage stage is strictly related to the previous one. This approach is also "enumerative" : each action can be selected from a number of variables that affect the system transformation. Each set of selected action (that

can be referred as "policy") is only a subset of a largest set of variables. For this reason all the selected actions have to be combined together and the problem is to determine the maximum or the minimum of a given function (optimal policy). This function is necessary in order to evaluate some properties of the system. To maximize or minimize a given function, the partial derivatives are considered and then the resulting system of equations is computed for the maximum or minimum point [1]. The cost function of a discrete problem defined over N stages is expressed as follows:

$$J = C_N(x(N) + \sum_{k=0}^{N-1} C_k(x(k), u(k), w(k)) \tag{2.1}$$

where $x(k)$ is the state vector, $u(k)$ is the control signal, $w(k)$ is a known quantity and k is a particular stage ($K = o$,$1, ..., N-1$). At each stage it is also possible to impose some constraints to the state variable and to the control signal.It is possible to define the optimal control U_{opt} that minimizes the objective function J. The optimal cost-to-go function is obtained by the minimization problem iterated from the last stage to the first one:

$$J_{opt}(x(k)) = \min_{u(k)}(J_{opt}(x(k+1)) + C_k(x(k), u(k), w(k))) \tag{2.2}$$

At this point some clarifications have to be highlighted. The analytic solution of a large number of equations is a big issue and the problem of the course of dimensionality has to considered. Moreover the actual solution may not represent the global maximum or minimum but only a boundary region around the solution. An optimal policy maximize or minimize a certain function formulated following a preassigned criterion, respecting the principle of optimality : "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision"[1]. The mathematical model used to describe the problem is a key point. If it is too realistic, the system results in too complicated equations, not easy to solve but the more it is realistic, the more accurate it will be. Dynamic programming solves a given problem subdividing it into simpler sub-problems, iteratively solves these subparts finding the optimal solutions. Mathematically this capability is expressed by the Bellman equation that evaluates the decision selected for the current time instant, considering the offset from the starting decisions and the remaining one. This kind of approach requires a dynamic system, control and state grids and a certain objective function. A control variable describe the operations performed by the system while state variable explicates the actual condition of the system. Ones that these two grids are defined, the dynamic programming algorithm computes all possible combinations between these quantities. This method has some remarkable advantages:

- total management of non linear and non differential problem;
- convergence to optimal solutions.

But it shows also several shortcomings:

- There is not a general formulation of the dynamic programming algorithm: each problem requires a specific formulation;
- Memory requirements necessary to store the solutions of the intermediate steps;
- Computational time required convergence;
- A priori knowledge of the driving mission (features and characteristics);
- Discretization requirements
- Curse of dimensionality i.e. dynamic programming can not be applied directly when the state space is large.

In the problem under investigation, dynamic programming consists in two phases: a backward and a forward phase. In the backward phase it starts from the last stage proceeding backwardly until the last stage is reached, calculating the optimal cost-to-go function and exploring all possible combinations between state and control variables.In the forward phase, the optimal trajectory is computed,from the initial state to the final one taking only the feasible solutions among all possible combinations resulting from the backward phase, always knowing the optimal cost-to-go function at each stage. It should be highlighted that this approach requires a consistent amount of computation time and that increasing the number of states and inputs has a direct impact on the memory requirements because they increase exponentially. Moreover the quality of the grid discretization of the inputs and of the states affect the accuracy and the feasibility of the solution. The optimal cost-to-go function is determined looking for the specific trajectory that minimize the cost function. This can be seen using a simple example. Let us consider Figure 2.1. The letters represent 11 cities and the final goal is find the shortest path from A to K. Proceeding backwardly, the first cities considered are H, I and J: the shortest trajectories to K are respectively 5, 3 and 7. Then it is considered the city E: EH is 4, EI is 5 and EJ is 6 but the shortest path EK is obtained passing through I. It can be noted that the shortest path from A to K is computed considering the shortest trajectory for every node and that the cost function in this problem is stage-wise additive (it is computed as a sum of cost functions for each stage).

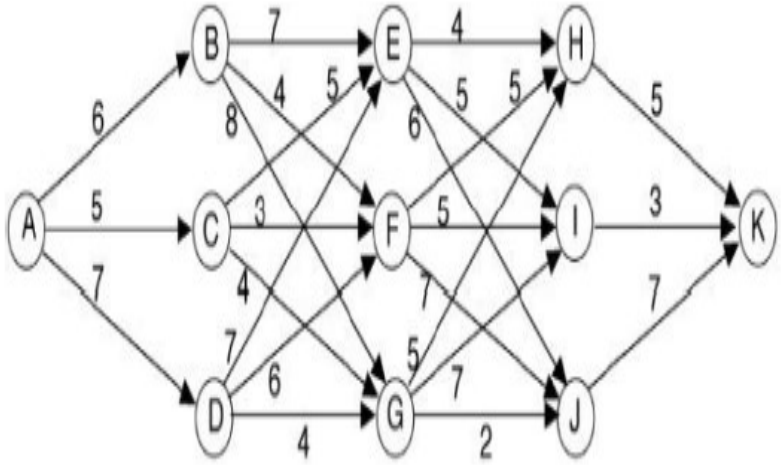


Figure 2.1: Dynamic programming [8]

2.2 Genetic algorithms

2.2.1 History and brief introduction

Evolutionary computation is a sub-field of algorithms used for global optimization that take inspiration by biological evolution. It is implemented on computer systems in order to solve problems, implementing techniques such as evolutionary algorithms and genetic one. Evolutionary programming was introduced by Lawrence J.Fogel while genetic algorithms where introduced by john Henry Holland whose main work is *Adaptation in Natural and Artificial systems*. Holland's career focus on the study of nonlinear systems i.e systems whose behaviour cannot be described using one of its subsystems (this phenomenon is known as *emergence*). Holland's work highlights the connection between emergence, individual and organizational adaptation. In common language, adaptation is connected to biological process where organism progresses by reorganizing and mutating its genetic material in order to survive to environment.
The population is made up of all the individuals. In contrast to the natural equivalent where the population size is dynamic, the algorithms have opted for the vast majority of cases for populations of fixed size. It should be highlighted that it is the population, and not the individuals, the main subject of evolution: it varies, evolves,improves the fitness of its own genotypes, moves towards local or global minima; on the contrary, individuals are static entities that are born and die with certain characteristics without modifying them in any way during their existence.

Before talking specifically about this topic, it is necessary introduce a basic terminology (Figure 2.2) .

- The population is a set of individuals;
- Chromosome: array with which is solution is coded;
- Gene: element position of the chromosome.

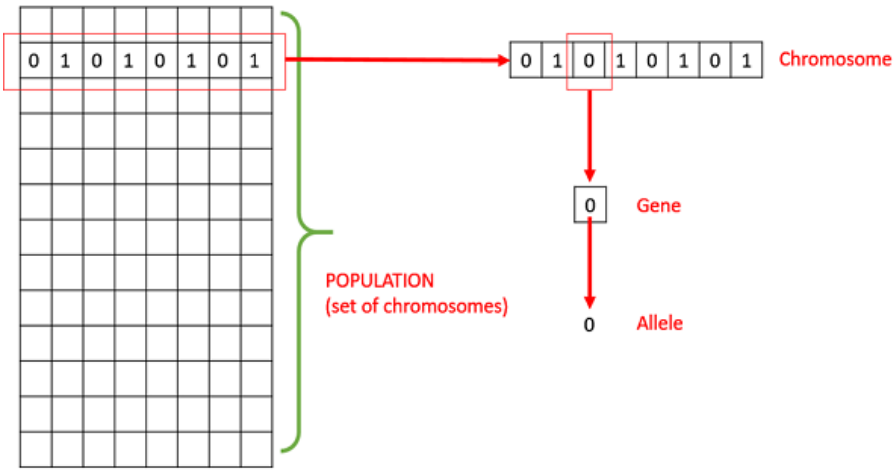


Figure 2.2: Genetic algorithm's terminology

The procedure implemented within a genetic algorithm consists of different steps:

- Generation of genes and chromosomes;
- "Creation" procedure (population inizialization);
- Fitness function and selection;
- Genetic operators (mutation);
- Reproduction mechanism (crossover);
- New generation.

The individual can be coded in different ways: binary representation, integer and real values. The binary representation is used when the problem results in solutions based on boolean logic; real values are used for continuous variables while the integer representation is used for discrete variables.
The starting point is a population of randomly generated solutions.The population is made up of all the individuals. In contrast to the natural equivalent where the population size is dynamic, the algorithms have opted for the vast majority of cases for populations of fixed size.
Selection refers to the choice of individuals who will become parents and through reproduction will give life to a new generation. This imposes the survival mechanism of the strongest among individuals: the main idea is that by preferring high quality solutions compared to worse alternatives, the individuals considered to be better, with higher fitness values, will have greater chances of reproductive success. It should be pointed out that even those considered unsuitable will have a reduced possibility of contributing to the formation of the next generation; this is because it is also important that the population maintains sufficient heterogeneity, we do not know if an individual considered unfit in a generation can not, by combining with a different individual, form a totally new and possibly better than all others.Variation operators correspond directly to those in the natural environment: mutation and recombination, in order to generate new and different individuals, starting with their parents.

2.2.2 Population

Population initialization (creation) can be done basically in two different ways:

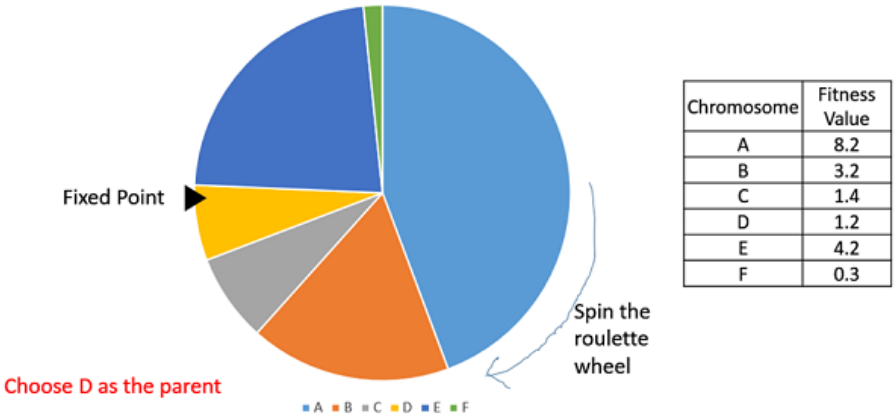
- randomly: the population consists of randomly generated candidate solutions;
- heuristically: it has been observed that this alternative could result in a population composed of too similar individuals.

There are also different population models:

- Generational: in this case, 'n' (n defines the size of the population) off-springs are generated and they will replace the entire starting population at the end of each iteration.
- steady state: if n is the size size of the population, m off-springs are generated with m<n and in each iteration they will replace m individuals of the previous population.

2.2.3 Fitness function

It is assigned a fitness value to each candidate solution (the score is assigned according to the objective function) in order to reproduce the Darwinian theory of "Selection of the fittest".
The fitness function evaluates each candidate solution assigning a score. The score represents the goodness of that solution according to the considered problem where the objective is to maximize or minimize a certain objective function.
If individuals have almost equal fitness values,they share almost equal portions of the pie. This leads to almost equal probability of getting selected as new parent. This results in a loss of selection pressure towards the fitter candidate solution and so poor genetic algorithm improvements. In the *Roulette wheel selection method (Figure 2.3*, you can imagine to have a roulette wheel, where each pie has a size proportional to the candidate's score. The probability to be selected is given by the individual's score divided by the total fitness of the solutions. As the wheel is turned, a random selection is performed.
In the *Tournament selection (Figure 2.4)*, n candidate solutions are selected in a



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figure 2.3: Roulette wheel selection [15]

random way. These n solutions are compared and the best one is selected as new parent. The procedure is repeated until the new offspring is generated.

2.2.4 Selection

The fitness function is crucial to perform the fitness proportional parent selection. Using this method, the chance to become a parent are strictly related to candidate solution score. Fittest individuals have good probability to be chosen, having the possibility to hand down their features to the next generation.

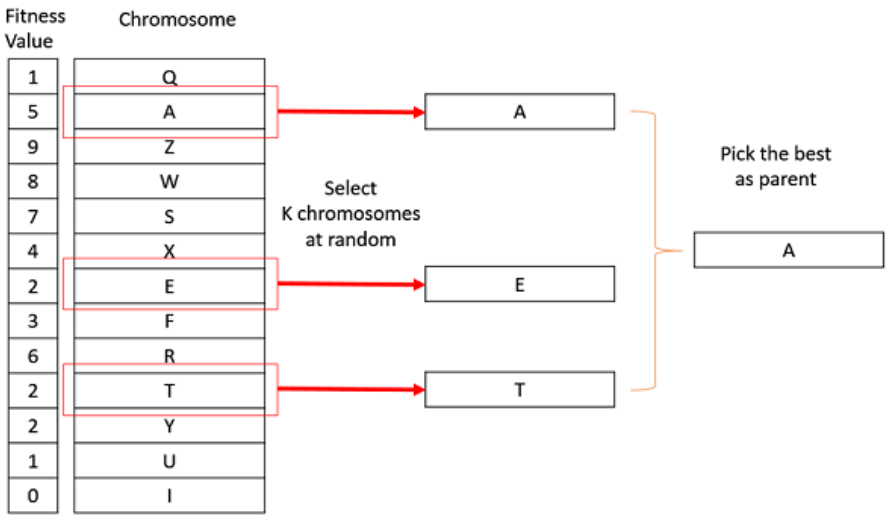


Figure 2.4: Tournament selection [15]

The best individual is always selected elite factor): in this way the algorithm results in a generation whose final individual has at least a score equal to the previous generation. This shrewdness is necessary because reproduction and mutation could have as shortcomings the lost of best individuals. It has been observed that the elite factor can speed up the process of convergence.

2.2.5 Crossover

The principle behind the crossover operator is the same as we find in nature: two individuals with characteristics desirable for evolutionary purposes, through reproduction, form one or more new individuals who have combined characteristics of their parents.
In genetics the equivalent is the fusion between two gametes (sperm and ovum) in the formation of the zygote; the gametes in turn are formed thanks to the process of meiosis during which a eukaryotic cell originates four cells. In the genetic algorithms, the operation is the same.
Different portions of the information that make up the selected individuals as parents are transmitted to the offspring in a combined manner. In this way there is a high probability that the new individual has an improvement over the parents.
This type of operator is binary, it uses the information coming from parents for the children generation. It is stochastic considering the randomness in the choice of which traits of the offspring come from a parent and which on the other.
This operator is used differently depending on the family of evolutionary algorithms:

in evolutionary programming it is never used while in genetic algorithms it represents the main research operator.
The main crossover operators used are:

1. One point:within the chromosome is selected a crossover point that divides it in two parts. This is done for both parents. The offsprings' chromosome is created by swapping the parents' chromosome parts (Figure 2.5).

2. Multi-point crossover: it is a generalization of the method 1 but in this case there are multi-point crossover (Figure 2.6).

3. Uniform crossover: each gene of the offsprings is selected by flipping a coin (Figure 2.7).

4. whole arithmetic recombination: the offsprings chromosomes are the results of the parents' weighted average (Figure 2.8).

$$y = \alpha \cdot x_1 + (1 - \alpha) \cdot x_2 \tag{2.3}$$

where x_1 and x_2 represent the chromosome arrays of two parents.

2.2.6 Mutation

Mutation is a variation operator; when applied to a genotype, it modifies a value by forming a new individual, different from the original one. This operator is unary (it acts on a single individual to produce a different one) and stochastic.
The algorithm works on the assumption that, after an unspecified period of time, it will converge on a global optimum. The possibility that this happens in a finite time is guaranteed by the ability of the algorithm to explore, theoretically, every possible candidate for the solution; the mutation in fact allows to "jump" from one point to another of this population's space without a pre-established logic. The mutation factor also presents its equivalent in natural evolution: the commonly accepted theories propose genetic mutations as the triggering cause of the diversification of living beings; these may be due to errors in the processes that occur on the genetic material or external interventions, such as pathologies or radiation.
The mutation mechanism at the beginning assumes a higher value in order to enlarge as much as possible the solution domain; then, as the algorithm proceeds, it decreases to reach convergence.

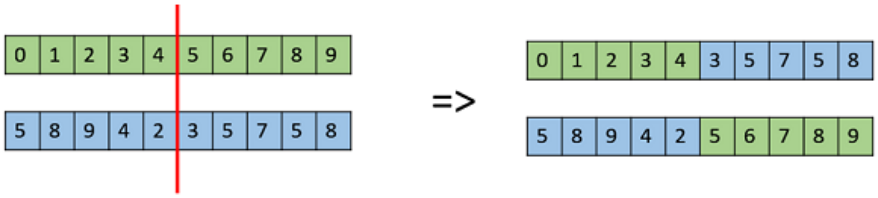


Figure 2.5: One point crossover [15]

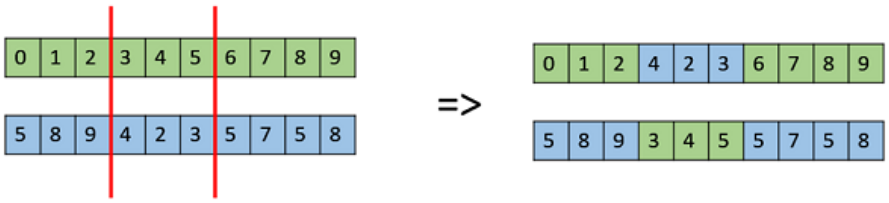


Figure 2.6: Multi point crossover[15]

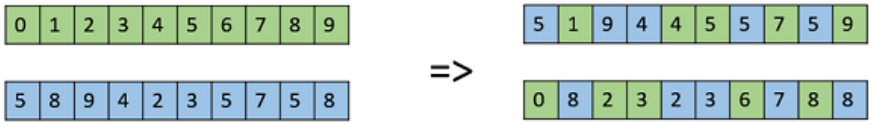


Figure 2.7: Uniform crossover [15]

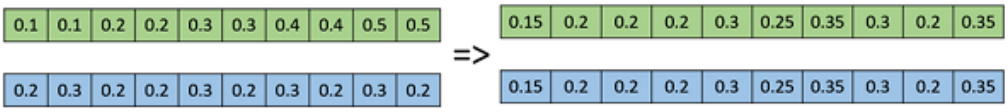


Figure 2.8: whole arithmetic recombination [15]

2.2.7 Termination condition

The main challenge of the genetic algorithms is the proper definition of a termination condition.

Direct methods

Direct methods ([10]) stop the algorithm if a pre-defined condition is satisfied. The main methods are:

- Maximal time budget: the condition is satisfied if a given time budget is consumed;
- Maximal number of generations: the criterion is satisfied if the pre-defined number of generation is reached;

- Hitting bound: this termination criterion is fulfilled if the final solution is close enough to the known optimum defined by the benchmark optimizer;
- K-iterations: this condition is fulfilled if there are no improvements in k consecutive iterations.

Derived termination conditions

Derived termination conditions ([10]) use auxiliary quantities obtained from the solutions of the current generation.

- Running mean: the condition is satisfied if the difference between the best solutions of both the current and the previous generation is equal or less than a certain value;
- Standard deviation: the criterion is fulfilled if the standard deviations of all solutions of the current generation is equal or less than a pre-defined value;
- Best-worst: the condition is satisfied if the difference between the best and the worst solution of the current generation is equal or less than a certain threshold;
- POP-Var: the criterion is fulfilled if the variance of all solutions of the current generations is equal or less than a certain value.

2.2.8 Comments

These algorithms have different merits. In fact they always provide a solution and are easy to implement. They also allow you to choose a trade-off between calculation time and precision and can be applied to various problems.
However, genetic algorithms do not guarantee that the solution offered after a certain number of iterations is the optimal one. Moreover, the number of the generations and the genetic operators (2.2.5, 2.2.6) can be determined only by tests, evaluating the performance and the quality of the results obtained.

Chapter 3

Model

The present study takes advantage of a pre-existing Optimization tool for layout and control strategy of HEVs [4].

3.1 Time grid

The problem under observation is discretized either in space that time domains. The time discretization implies the creation of a grid.This grid can be realized following different approaches.
In the *node grid* approach a grid is realized with $N_{in} + 1$ nodes. At each node, it is assigned a number to each control variable. The system variables sv are functions of the control variables n_c and of the state variables n_s:

$$sv(i) = f(cv_1(i), ..., cv_{n_c}(i), sv_1 i, ..., sv_{n_s}(i)) \tag{3.1}$$

This approach is not implemented because this grid is characterized by a low number of points, resulting in poor accuracy.
On the other hand, a too refined grid can not be implemented because of high computational time demand.
A *interval-grid* approach consists in the division of the grid in N_{in} intervals whose length is chosen in such a way to reduce the number of time steps. Over the interval, the control variables are assumed to be constant and the system variables are evaluated on either extremes of each time interval. The intermediate variables are computed thanks to linear interpolation. Applying this approach, it is possible to consider the physical discontinuity that could occur at each node, reducing the simulation time and the memory requirements.
The vehicle model equations are written by means of two sub-control variables for each variable sv, considering in this way both sides of each node:

- For the leftmost extreme of the interval

$$sv_l(i) = f(cv_1(i), ..., cv_{n_c}(i), sv_{1,l}(i), ..., sv_{n_{s,l}}(i)) \tag{3.2}$$

- For the rightmost extreme of the interval

$$sv_r(i) = f(cv_1(i), ..., cv_{n_c}(i), sv_{i,r}(i), ..., sv_{n_{s,r}}(i)) \tag{3.3}$$

being $cv_{j,l}(i) = cv_{j,r}(i)$ for each cv_j.

The considered input variables are the vehicle velocity V_v and the slope of the road. The reference vehicle has been tested by means of a dynamic model while a kinematic model has been developed in order to estimate the HEV performance. The velocity time history of the dynamic model is the input variable in the kinematic model. This procedure guarantees the vehicle ability to satisfy the required power demand.

In the present study, the main control variables considered are the gear number and the powerflow. They can assume a finite number of discrete variables N_i, in fact the space domain of each control variables is represented by the following set:

$$S_{cv,i} = 1, ..., N_i \tag{3.4}$$

The gear number can assume values from 1 to 6 because the considered vehicles in this study are equipped with a 6 transmission ratios gearbox. The powerflow expresses how the power is split among the power sources (engine and electric machines) on the front/rear axles. The values that the PF control variable can assume depend on the specif hybrid architecture considered. In the present study it is considered a parallel architecture and so the PF control variables can assume only four values: pure electric, pure thermal, battery charging and power split.
In order to fully describe the control variables, sub-control variables are defined. The GN sub-control variable is the transmission ratio while the PF sub-control variables give information about power split between front and real axles and between engine and electric machines.
In order to univocally determine the powertrain control, it is created a global control variable, the working mode, that includes all combinations of the two control variables.

$$S_{wm} = S_{cv} \tag{3.5}$$

The variables used as state variable are:

- engine state
- battery state of charge

The engine state is set to be equal to 1 when the engine state is on, 0 viceversa. The battery state of charge or SOC ranges between two values, SOC_{min} and SOC_{max}. It has to be highlighted that the SOC evolution is obtained considering the working-mode time history while the engine state is obtained by the instantaneous working mode value.

3.2 Engine

The main power source in parallel hybrid electric vehicle is the internal combustion engine. In a hybrid vehicle the engine is used more efficiently with respect to a conventional vehicle, in fact is runs at high power for longer period of times. In this study it is used a four stroke ignition compression engine; this means that the operative phases are four (suction, compression, expansion and exhaust). In this kind of engine the air only is compressed and this causes its temperature to increase. The fuel is directly injected within the cylinder just before the combustion and is ignited by the elevated temperature of the air.
Between the start of injection and the start of combustion there is a delay due to physical and chemical phenomena as:

- jet atomization;
- air-fuel vapor mixing;
- preliminary reaction steps.

It is characterized by high expansion ratio and lean air/fuel mixture. The primary pollutants emitted by diesel engine are:

- NOx due to high temperatures reached in the diffusive combustion phase;
- HC due to fuel retained within the nozzles sac/spray holes and in the neighborhoods of the reaction zone where the mixture is too lean;
- soot particles due to agglomeration of carbon particles produced in the hydrocarbon cracking phase.

Look-up tables are used to model the engine performances; the fuel mass flow rate of fuel $\dot{m}_{fc}$ is evaluated by interpolating 2d map and used for CO_2 emissions evaluation.

$$\dot{m}_{fc} = map_{fc}(P_{e,mech}, \omega_e) \tag{3.6}$$

$$CO_2 = 2.65\frac{\dot{m}_{fc}}{\rho} \tag{3.7}$$

3.3 Electric machine

The main components of the electric propulsion system are:

- electric machine;
- power converters;
- electronic controllers.

The electric machine can propel the vehicle or or recharge the battery pack. Usually the (P)HEVs are equipped with three phase electric machines; the most common are:

- Asynchronous: this type is characterized by great overload capacity;
- Synchronous permanent magnet: this type presents a higher efficiency than the previous type.

In the present study the selected electric machines are part of the brushless permanent magnet category (the technical data can be found on the UQM Technologies website).
The reason behind this choice is due to the advantages of these systems:

- Compactness due to the intrinsic characteristics of the magnets (high-energy density);
- High efficiency: no wasted power to produce excitation, absence of brushes and commutator;
- Simplicity in the control and in the maintenance.

However this category shows also different shortcomings:

- limited maximum speed;
- high costs due to magnets,
- limited constant power range.

The Optimal-layout tool selects the maximum power of the machine defining in this way the size of the component.The maximum power is necessary to scale the efficiency maps that are used to simulate the energy losses due to the power conversion.

3.4 Battery

Lithium ion batteries are characterized by high cell voltage, high energy efficiency and long life. The positive electrode consists of a metal oxide containing lithium while the negative electrode consists of carbon material. The electrolyte is a liquid solution of dissociable lithium salt and an organic solvent.
During the discharge phase the lithium ions migrates from the negative electrode to the positive one through the electrolyte; during the charge phase the process is reversed.
One of the main drawback of the lithium-ion battery is the reduction in their charge capacity (and so reduction of their deliverable power) as the charge/discharge cycles increase [11].

3.5 Torque coupling device

The torque coupling device is used to couple the torque of the engine and of the electric machine. The vehicles considered in this study are equipped with a two-shaft gearbox (Figure 3.1).

Referring to the contact point, the velocity correlation between the two shaft is

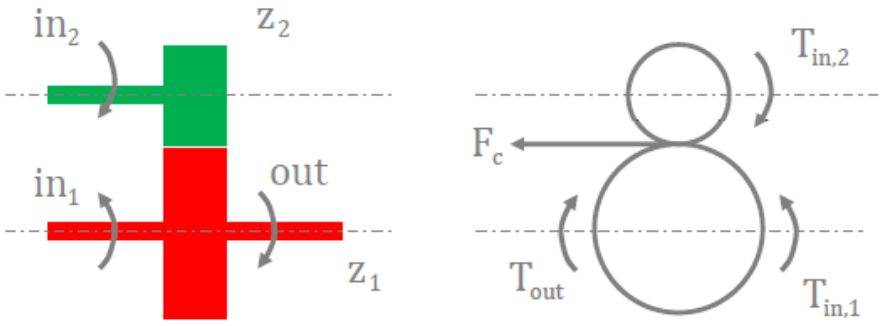


Figure 3.1: Torque coupling device [4]

computed as follows:

$$\omega_{out} \cdot z_1 = \omega_{in,1} \cdot z_1 = \omega_{in,2} \cdot z_2 \tag{3.8}$$

where z_1 and z_2 indicates the number of teeth of the two gears.
Introducing the speed ratio $\tau = \frac{R_1}{R_2} = \frac{z_1}{z_2}$, the previous equation can be rewritten as:

$$\omega_{out} = \omega_{in,1} = \frac{\omega_{in,2}}{\tau} \tag{3.9}$$

Considering for example the shaft 2 and and neglecting loss sources, the transmitted force at the contact point is computed considering the input torque at the shaft:

$$F_c = \frac{T_{in,2}}{R_2} \tag{3.10}$$

The equilibrium equation at the gear of the shaft 1 can be written as follows:

$$T_{out} = T_{in,1} + F_c \cdot R_1 = T_{in,1} + \frac{T_{in,2}}{R_2}\dot{R}_1 = T_{in,1} + T_{in,2} \cdot \tau \tag{3.11}$$

The previous equations can be written as:

$$\omega_{out} = \frac{\omega_{in,1}}{k_1} = \frac{\omega_{in,2}}{k_2} \tag{3.12}$$

where $k_1 = 1$ and $k_2 = \frac{z_1}{z_2} = \tau$.

3.6 Pre-processing phase

For a given control strategy it is possible to define the vehicle configuration, for either extremes of each grid's interval. The user can selects the tool's inputs, specifying the vehicle, the powertrain, the cycle and the controller to be used. The optimizer defines a specific control strategy near optimal according to the criteria defined at the beginning of the study. In this case the configuration of the vehicle is defined without consider the SOC and it is independent from the control strategy: it is evaluated is a pre-processing phase decreasing in this way the computation time. In the pre-processing phase, it has been developed a zero-dimensional kinematic model of the vehicle and the interval-grid approach has been used. For each configuration and for each extreme of each interval, the intermediate and the control variables are defined.
The pre-processing phase consists of different steps:

- Acquisition of input variables;
- Components maps generation;
- Sizing of the battery;
- Generation of configuration matrix;
- Definition of components requirements;

- Feasibility check.

The matrix of configurations stores the values of the vehicle configurations defined as the combination of the working modes (gear number and power flows) and state variables (engine state and battery state of charge).
The configuration space is defined as:

$$S_{CONF} = S_{GN} \times S_{PF} \times S_{ES} \tag{3.13}$$

And the number of all possible configuration is expressed as

$$N_{CONF} = N_{GN} \times N_{PF} \times N_{ES} \tag{3.14}$$

The specific configuration selected by the user is analyzed by the tool in order to generate the maps related to each component.
Each map is generated considering the reference map provided by the manufacturing partner, using a scaling factor.
For the engine:

$$ScalingFactor = \frac{V_{engine}}{V_{engine,ref}} \tag{3.15}$$

For the electric machine:

$$ScalingFactor = \frac{P_{peak,EM}}{P_{peak,ref_EM}} \tag{3.16}$$

The total vehicle power demand P_v is computed considering several contributions:

$$P_v = P_{v,roll} + P_{v,grade} + P_{v,drag} + P_{v,inertia} \tag{3.17}$$

The rolling resistance $P_{v,roll}$ is expressed as:

$$P_{v,roll} = V_v \cdot (m_v \cdot g \cdot r_v \cdot \cos \rho_r) \tag{3.18}$$

The grade resistance is given by:

$$P_{v,grade} = m_v \cdot g \cdot \sin \alpha_r \cdot V_v \tag{3.19}$$

where m_v is the vehicle mass, r_v the vehicle rolling resistance coefficient, g the gravitational acceleration, α_r the slope of the road and V_v the velocity of the vehicle.
The aerodynamic drag resistance is expressed as follows:

$$P_{v,drag} = (\frac{1}{2} \cdot \rho_{air} \cdot c_x \cdot A_v \cdot V_v^2) \cdot V_v \tag{3.20}$$

where A_v is the frontal area of the vehicle, c_x is the aerodynamic drag coefficient and it takes into account the shape of the vehicle, ρ_{air} is the density of the air. The inertia contribution is given by:

$$P_{v,inertia} = (m_v + \frac{I_{wh}}{R_{wh}}) \cdot V_v \cdot \dot{V}_v \tag{3.21}$$

where m_v is the vehicle mass, I_{wh} the wheel inertia and R_{wh} is the dynamic wheel radius.
The power sign defines the vehicle working conditions (traction or braking). The mechanical power is split between front and rear axles.
The mechanical power at the front wheels $P_{wh,f}$ is:

- in traction:
$$P_{wh,f} = (1 - \delta) \cdot P_v \tag{3.22}$$
- in braking
$$P_{wh,f} = \gamma_{fr} \cdot P_v \tag{3.23}$$

The mechanical power at the rear wheels $P_{wh,r}$ is:

- in traction:
$$P_{wh,r} = \delta \cdot P_v \tag{3.24}$$
- in braking
$$P_{wh,r} = (1 - \gamma_{fr}) \cdot P_v \tag{3.25}$$

The power at the final drive level is expressed as follows:

- front axel
$$P_{fd,f} = \begin{cases} P_{wh,f} \text{ in traction} \\ (1 - \gamma_{br}) \cdot P_{wh,f} \text{ in braking} \end{cases} \tag{3.26}$$
- rear axel
$$P_{fd,r} = \begin{cases} P_{wh,r} \text{ in traction} \\ (1 - \gamma_{br}) \cdot P_{wh,r} \text{ in braking} \end{cases} \tag{3.27}$$

Considering the inertial power related to each component, it is possible to evaluate the required power at the front powertrain level $P_{pt,f}$ and the required power at the rear powertrain level $P_{pt,r}$.

For the transmission:

$$P_{tr,in} = I_{tr} \cdot \dot{\omega}_{tr} \cdot \omega_{tr} \tag{3.28}$$

For the engine:

$$P_{e,in} = I_e \cdot \dot{\omega}_e \cdot \omega_e \tag{3.29}$$

For the electric machine

$$P_{em,in} = I_{em} \cdot \dot{\omega}_{em} \cdot \omega_{em} \tag{3.30}$$

$$P_{pt,f} = (P_{fd,f} + P_{tr,in}) \cdot \eta_{tr}^{k} + P_{e,in} + P_{em,in} \tag{3.31}$$

where η_{tr} is the transmission efficiency and k is equal to 1 if the vehicle is braking, -1 otherwise.

$$P_{pt,r} = P_{fd,r} + P_{em,in} \tag{3.32}$$

The angular velocity at the front powertrain level $\omega_{pt,f}$ is computed as follows:

$$\omega_{pt,f} = \tau \cdot \tau_{fd,f} \cdot \frac{V_v}{R_{wh}} \tag{3.33}$$

where $\tau_{fd,f}$ is the speed ratio of the front final drive and R_{wh} is the dynamic radius of the wheel.
Similarly, at the real level:

$$\omega_{pt,r} = \tau_{fd,r} \cdot \frac{V_v}{R_{wh}} \tag{3.34}$$

where $\tau_{fd,r}$ is the speed ratio of the final drive. Each components is analyzed in order to verify if it is capable or not to supply power and velocity requirements by looking at the previously generated maps. The maximum power of the battery is evaluated with the optimal layout tool taking into account the electric power of the electric machines and the architecture of the vehicle.The number of cells is determined starting from the maximum power of the battery:

$$N_{cell,p} = \frac{V_c N_{cell,tot}}{500} + 1 \tag{3.35}$$

The number of cell in parallel is computed by imposing as maximum admissible voltage of the battery 550V:

$$N_{cell,p} = N_{cell,p} N_{cell,s} = \frac{P_{max}}{V_c \cdot I_{c,max} - R_c \cdot I_{c,max}^2} \tag{3.36}$$

$$N_{cell,s} = \frac{N_{cell,tot}}{N_{cell,p}} \tag{3.37}$$

The total mass is calculated considering the mass cell and it is taken into account an additional 70 % in order to consider the cooling system:

$$m_{bat} = 1.7 \cdot m_c \cdot N_{cell,tot} \tag{3.38}$$

The voltage V_{bat}, the resistance R_{bat}, the capacity C_{bat} and the energy content E_{bat} are evaluated as:

$$V_{bat} = V_c \cdot N_{cell,s} \tag{3.39}$$

$$R_{bat} = R_c \cdot N_{cell,s} \tag{3.40}$$

$$C_{bat} = C_c \cdot N_{cell,p} \tag{3.41}$$

$$E_{bat} = C_{bat} \cdot V_{bat} = C_c \cdot V_c \cdot N_{cell,tot} \tag{3.42}$$

The limit thresholds of the SOC are imposed to be [0.4-0.8]; the battery life is assumed to be 150000 Ah and the maximum current transferable through a cell is 120 A. The battery is modeled as an equivalent resistance circuit (Figure 3.2) where the resistance and the open-circuit voltage of the battery are computed by interpolating their 1D maps that are battery SOC functions. The system is assumed to be temperature-independent i.e constant temperature of the battery.

The battery is designed in order to satisfy the power requirement of each electric

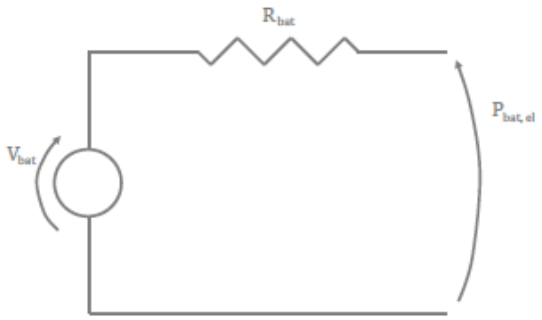


Figure 3.2: Battery model [4]

machine:

$$P_{batt,el} = (\sum_{i=1}^{n} P_{n,el}) \cdot \eta_{inv}^{k} \tag{3.43}$$

where n indicates the number of electric machines, η_{inv} represents the efficiency of the inverter, $k = 1$ in the charge phase and is set equal to -1 in the discharge phase.

The first requirement to the system is dictated by the electric power demand. The battery current is derived from the power balance applied to the circuit considering the specific power demand required $P_{bat,el}$:

$$P_{bat,chem} = V_{bat} \cdot I_{bat} = P_{bat,el} + R_{bat} \cdot I^2 \tag{3.44}$$

The battery current is obtained

$$I_{bat} = \frac{V_{bat} - \sqrt{V_{bat}^2 - 4 \cdot R_{bat} \cdot P_{bat,el}}}{2 \cdot R_{bat}} \tag{3.45}$$

The maximum battery current is obtained as

$$P_{bat,el,max} = \frac{V_{bat}^2}{4 \cdot R_{bat}} \tag{3.46}$$

The equivalent battery capacity and the current flowing through the battery are used to compute the SOC value:

$$SOC = SOC_0 - \int \frac{I_{bat}}{C_{bat}} dt \tag{3.47}$$

In all those cases in which the electric power demand $P_{bat,el}$ exceeds the limits imposed by the maximum battery power $P_{bet,el,max}$, the relative combination of control variables is imposed to be not admissible.

3.7 Optimization phase

The outputs coming from the pre-processing phase (total feasibility matrix, fuel consumption configuration matrix, feasibility matrix of the battery) are used in the optimization phase. Once defined the objective function as:

$$J = C_N(x(N) + \sum_{k=0}^{N-1} C_k(x(k), u(k), w(k)) \tag{3.48}$$

the optimal policy is determined by:

$$p^*(t) = \min_{u(t) \in S_{cv}^*(t)} J(p(t)) \tag{3.49}$$

$$SOC_{end} \geq SOC_0 \tag{3.50}$$

where $p(t)$ is the generic policy, S_{cv} indicates the set of admissible control variables, SOC_{end} and SOC_0 indicate respectively the SOC at the end and at the beginning

of the driving mission.
The dynamic programming is the global optimization strategy implemented in order to find the optimal control strategy. It consists of two phases.

Backward phase

In the backward phase the procedure starts from the last interval:the cost function is evaluated considering all the combination of the state variables (SOC and engine state) at the end of the interval and all the combinations of control variables found in the interval.The fuel consumption in this interval is computed by reading from the configuration matrix the fuel consumption at the extreme of the interval and integrating them over the time interval.Also the engine state at the end of the interval is read and thanks to the battery model it is computed the SOC value and a score is assigned. The tool detects the best combination of control variable for all possible combinations of state variables. The procedure is iteratively repeated moving backwardly in the time grid, from the last to the first interval.

Forward phase

In the forward phase the tool moves forwardly from the first interval to the last one, using the results obtained from the backward phase; it reads from the configuration matrix the results corresponding to the combination of control variables detected during the backward phase and calculates the battery soc by means of the battery model. The evaluated soc and the engine state obtained at the end of the first interval are used as input of the second interval in order to detect the best control strategy. The procedure proceeds in this way until the end of the selected driving mission.

Chapter 4

Control strategy

The tools presented in this chapter are real time optimizers. They use the results obtained with the global optimizer described in the previous chapters. These tools are characterized by lower computation time and lower memory requirements.

4.1 CORE TOOL

The acronym CORE stands for Clustering optimization rule extraction [4], [5]. It is a real time and rule based optimizer that uses as internal solver the genetic algorithm and exploits the results obtained by the dynamic programming identifying an appropriate set of rules for a given driving cycle.

4.1.1 Training

Each real-time optimizer requires a training phase in which the previously defined objective function is taken into account. In this phase there are not particular constraints in the soc of the battery in order to have more freedom in the identification of the optimal training driving cycle (this means that at the end of the driving mission the soc level could achieve greater value than those of the beginning).

The input states of the tool are represented by the discretization of the input varibles namely velocity, velocity variation and SOC. The output states are constituted by the discretization of the control variables, gear number and powerflow. The input states are connected to the output ones by means of a rule defined by the global control strategy (dynamic programming). Core consists of two main algorithms:

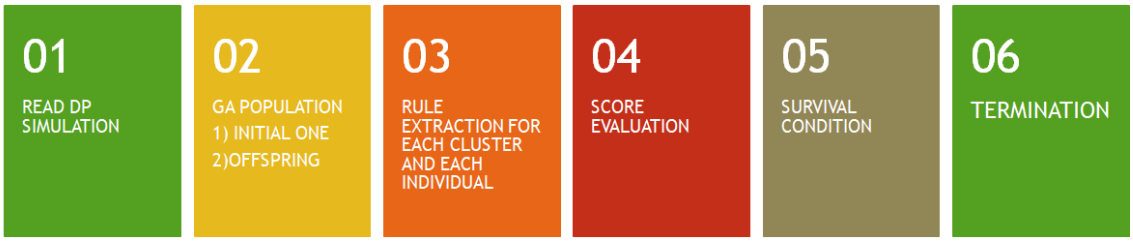


Figure 4.1: Training flow chart

- CO:Clustering of optimal driving conditions
- RE: Rule extraction

Clustering of optimal driving conditions

The CO algorithm is used to generate a 3d map. This map is necessary to classify within particular clusters the data. Each specific cluster collects data with similar features and characteristics. Each input variable represents one axis of this map and it is discretized into a number of states: this means that each axis is divided into non-equispaced segments obtaining in this way a mesh of clusters. The number of segments per each axis is given to the tool as input. The mesh can be seen as a series of parallelepipeds whose number is given by the product of the states of the input variables.

$$N_c = \prod_{j=1}^{N_i} S_j \tag{4.1}$$

where N_c is the number of clusters, N_i the number of input states and S_j is the number of states i.e the number of segment in which each axis is divided.
The lower and the upper thresholds of each input variables are defined at priori: $v = [0\ 150]$, $\frac{dv}{dt} = [-40\ 20]$, the SOC level $= [0.4\ 0.8]$.
Once the map is generated, the points of the actual driving mission are distributed within the correct cluster.
For each cluster is identified a rule, i.e the optimal combination of gear number and powerflow, by means of the rule extraction algorithm.

Rule extraction algorithm

In this step the results of dynamic programming are crucial. Because each cluster occupies a portion of the space, there the actions performed by the dynamic programming not necessarily are unique, in fact it is possible to have more powerflows selected by the dynamic programming in that portion, all of them feasible.
As first guess, it is selected the most frequent action that becomes the unique policy used for that cluster. In all that cases in which the clusters remain empty because no points occur within them, it will be selected a backup rule (the pure electric mode). Applying a backup rule is not optimal and in order to avoid this drawback it is necessary impose the correct discretization of the map, trying to find the best solution that allows to not leave too much point of the driving mission with a backup solution that could not be the optimal one.

Genetic algorithm

The performance of this method are correlated to the mesh of the 3d input domain: different discretization could have as consequence that some points might fall into different clusters. The internal solver used within this tool is the genetic algorithm and the previously defined procedure is performed for the whole population.
The genetic algorithm is adopted in order to define the best size of the grid for each input variable in order to minimize the objective function (minimization of fuel consumption and CO_2 emissions ttw). The aim of the ga is to find the best individual and so the best grid size.
The main parameters involved are listed in Table 4.1:
The crossover factor determines the number of individuals affected by crossover.

Population dimension N_{ind}	1500
Generations N_g	20
Crossover factor χ	0.7
Fraction of mutated genome μ_1	0.1
Elite children	1

Table 4.1: Values of main parameters of GA

$$N_{ind,cross} = f_{cross} \cdot (N_{in,pop} - N_{elite}) \tag{4.2}$$

where f_{cross} is the crossover fraction, $N_{in,pop}$ is the number of individuals of the starting population, N_{elite} the number of elite children. The number of individuals affected by mutation is determined as follows:

$$N_{mut} = N_{in,pop} - N_{elite} - N_{ind,cross} \tag{4.3}$$

In the Matlab environment, each individual is reproduced by means of an array that represents the genome of the individual. The genome length is computed as:

$$N_k = \sum_{j=1}^{N_i} (S_j - 1) + 1 \tag{4.4}$$

An individual keeps all the informations in terms of grid size necessary to generate the mesh. In details, it consists of different elements, each one represents the length of the segments associated to the input variable i.e velocity, velocity variation and soc. The analysis is performed over a population of 1500 individuals, two orders of magnitude greater than the genome's length.
At the beginning this population is generated randomly: this means that the length of each element of each input variable that forms each individual is imposed randomly. Using the information kept within each individual, it is generated a map and for each cluster is applied the most frequent action performed by the dynamic programming. So, the number of segments of each axis is imposed at priori but the length of each segment is different passing from one solution (individual) to another. Once the grid is built and the rules are assigned to each cluster, all the individuals are evaluated in order to estimate fuel consumption and the feasibility of all the components. The fuel consumption is evaluated considering the current rule and extrapolating the consumption associated to that rule in the configuration matrix; this is done for each time instant of the driving mission and for the whole population. Also the feasibility of the vehicle is evaluated considering the rule in the current cluster. This evaluation involves the assignment of a score to each candidate solution and when all the individuals are evaluated for the total driving mission, the assigned scores are used in order to perform a fitting selection based on the minimization of CO_2 tank to wheel.
The successive generation are not generated randomly. The best individual of the previous generation , i.e. the individual with the minimum score, is always selected while the other individuals are affected by crossover and mutation. The mutated ones are affected by a mutation factor. In the code this is realized multiplying the selected array by a scale factor that contracts as the number of generations increase in order to reach convergence, as it is suggested from genetic algorithm literature.

$$\gamma = \frac{1}{N_{gen}} \tag{4.5}$$

$$F_{mut} = 1 - \gamma \cdot N_{gen,act} \tag{4.6}$$

where γ is a constant factor while F_{mut} is the mutation factor, $N_{gen,act}$ represents the actual generation's number.
The others are affected by a crossover factor i.e each individual is the results of a

mixture of elements coming from two different individuals of the previous generation selected according to their scores.
The selection procedure is realized taking into account the score of each individual and applying the roulette wheel selection. Let us suppose to have a wheel where each candidate solution occupies a pie whose size is strictly related to its score: the lower the score, the greater is the portion of the pie occupied by that individual. Then we can suppose to tune the wheel, selecting in this way a certain individual: obviously the greater the portion of the pie, the greater the chance to be selected as future parent. Once the candidate parents are selected, their genes are mixed using two approaches. The first approach is labeled as *Parent Combination* and implies the linear combination of the parents' genes. Mathematically this relation can be expressed as follows:

$$y = x_1 \cdot \alpha + (1 - \alpha) \cdot x_2 \tag{4.7}$$

In Figure 4.2, the results obtained by applying the Parent combination method are showed. The simulation is performed over the Wltp cycle with an heavy duty vehicle. As can be seen, the convergence condition is reached after few generations because of too similar candidate solutions. The second approach is labeled as *Parent*

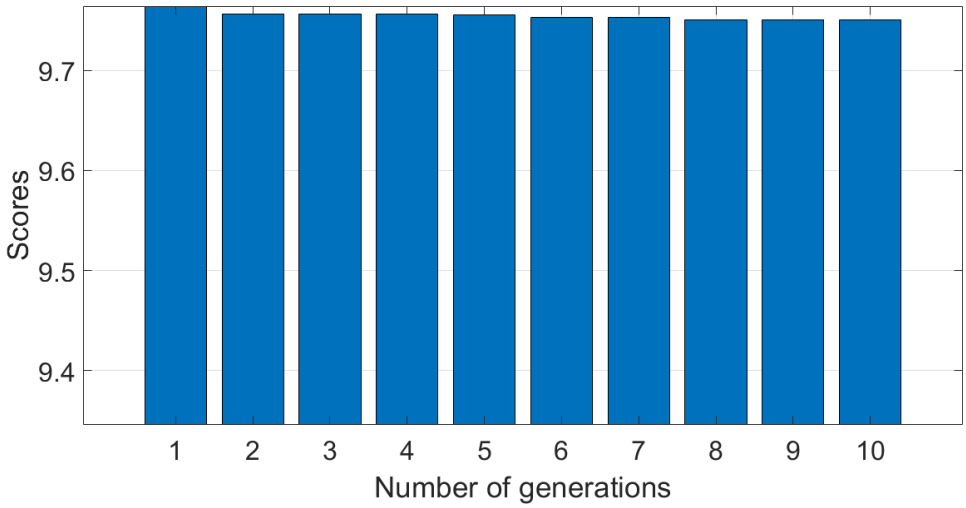


Figure 4.2: Parent combination method, 1000 individuals, 10 generations

Selection and implies the random combination of the parents' genes by flipping a coin. The results obtained through the *Parent Selection approach* are showed in the Figure 4.3. In Figure 4.4, the results obtained by alternating the two methods during even and odd generations are showed. As can be noted, the obtained final score is very close to the final score obtained by applying the *Parent combination*

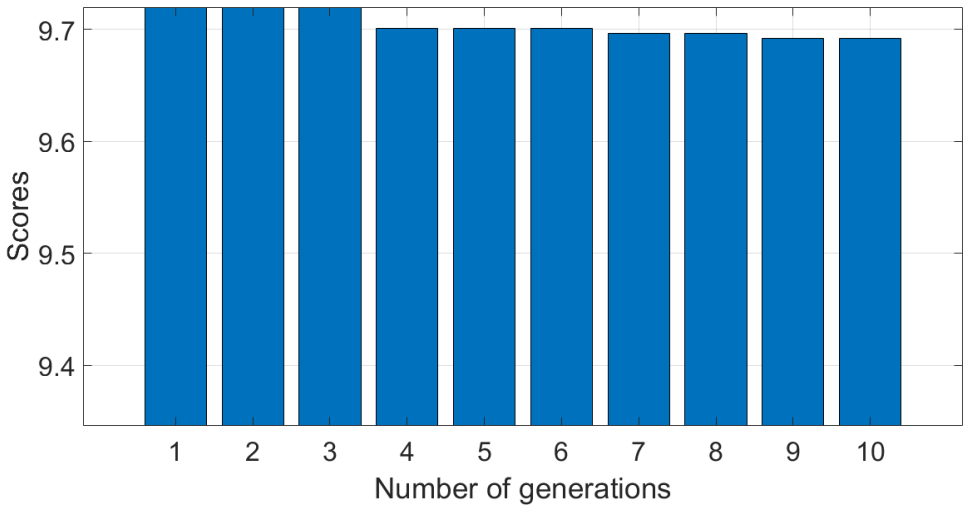


Figure 4.3: Parent selection method, 1000 individuals, 10 generations

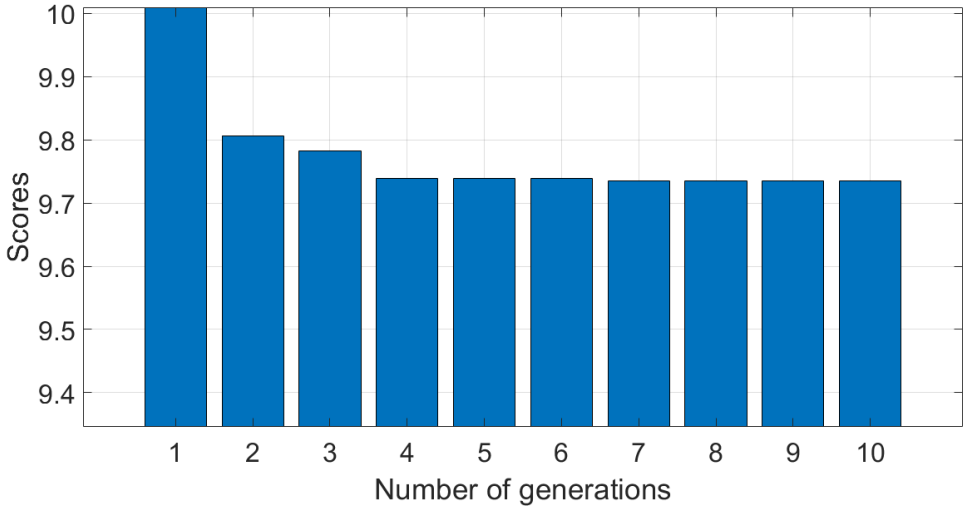


Figure 4.4: Hybrid method, 1000 individuals, 10 generations

method. The termination condition is reached after a pre-defined number of generations [10] and the best element of the last generation represents the best grid.
The correct number of the generations can not be predicted at priori: theoretically a genetic algorithm is thought to improve constantly, however, to select an acceptable number in a weighted way, the factors considered are different such as the extent of improvement, the achievement of the convergence tested on different tests and the distance of the best solution compared to the standard deviation of the population.

4.1.2 Validation phase

In the validation phase, the results obtained from training, i.e the disctretized map and the set of rules, are validated. The points of the actual driving mission are splitted within the clusters and in each cluster it is used the rule extracted from the dynamic programming during the training phase. In this way it is possible to evaluate fuel consumption and CO_2 ttw emissions.
The scores of the training phase represent the kg of CO_2 tank to wheel. At the end of the validation phase the carbon dioxide is evaluated in terms of g/km but this score is greater than the score obtained during the training phase because of a penalty that takes into account that the soc level is not brought back to the level that it had at the beginning. It is possible to validate a driving cycle using a training made on itself, obtaining in this way the best result possible for that driving cycle. It is also possible validate a driving cycle using a training performed over a different driving mission: in this case it is expected a deterioration of the performance of the tool but this deterioration can be mitigated selecting properly the training driving missions.

4.1.3 Multiple training

The training can be performed also on more driving cycles. For each cycle it is repeated the same procedure used before:

- distribute points of the actual driving cycle within the clusters;
- extrapolate the rule for each cluster;
- evaluate CO_2 according to fuel consumption;
- assign a score;

Even if the cycles are inserted following a certain order, when the current cluster is identified, the points that coincide will have the same rule. Once that this procedure is done for all driving cycles, the final score is determined as the mean value of all the scores and the best individual will always be characterized by the minimum score.
The way in which the final score is determined is a critical issue because the weight assigned to the score associated to each driving mission can affect the final results. By weighing the scores of each cycle on the cycle distance, the results are very close to those obtained performing the simple average of the scores. In some cases, slightly

lower final scores have been obtained, however, since this is not repeatable on a large set of cycles, this approach has been abandoned.

4.1.4 Modification to termination condition

Instead of using a pre-defined number of generations, it is calculated an error with respect to the dp. If it is lower than a certain percentage, the loop is broken before the termination condition; if it is greater, the threshold of termination condition is increased and the loop re-starts again.
Type of error:

- Relative error defined as the difference between the actual score and the dp score normalized over the dp score.
- standard error:estimate of the standard deviation. It is equal to the variance of the population divided by the square root of the sample size.

This is done because the improvement obtained with the default number of generations is often only a few percent units or even lower percentages.

4.1.5 Rule extraction

Initially the algorithm selects always the most frequent action performed by the dynamic programming: in this case it is observed that the genetic algorithm, passing from one generation to the successive one, always improves. In fact the most frequent action leads to at least equal or less score with respect to the previous generation.
The mode is the most frequent value present in a certain data set. The entire dataset can be represented on a histogram (Figure 4.5) and in this case the highest bar will represent the mode. Generally speaking it can be considered the most "popular" option.

Not necessarily the mode is unique since it is possible to have the same maximum value at several points. Usually this function is used to show the most common category. In a normal distribution, the mode, the mean value and the median assume the same numerical value.
The mode(s) can be obtained through the frequency distribution of the observed data.
In the matlab environment this is realized thanks to a function called mode.

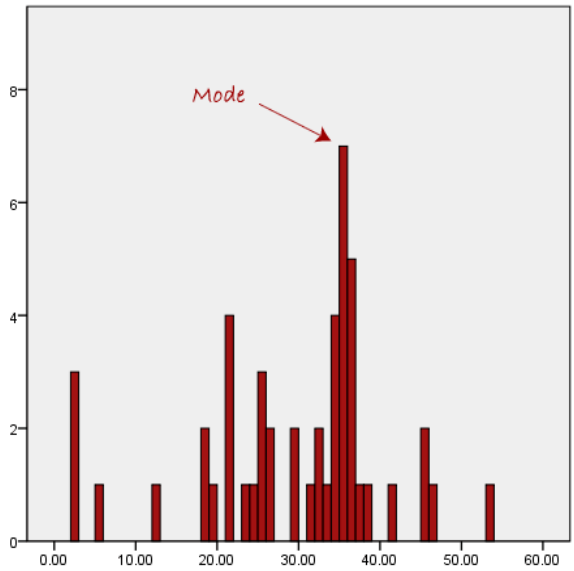


Figure 4.5: Mode

4.1.6 Modifications to rule extraction

In order to select properly the action performed by the dynamic programming, it is necessary a statistical analysis. Two basic concepts need to be clarified before start: population and samples. Population is the collection of all individuals or items under consideration in a statistical study [9]. Sample is that part of the population from which information is collected . A statistical population is the set of measurements corresponding to the entire collection of units for which inferences are to be made. A sample from statistical population is the set of measurements that are actually collected in the course of an investigation [9]. In our case, the population is made by all possible combinations of power flow and gear number while the samples are only those combinations actually performed by dynamic programming in the current cluster.
The final aim of this analysis is make inferences about population using information stored in the sample data. The number of observations contained within a certain class is the frequency of that class. The frequency distribution is a graphical way (usually table or histogram) to show all the classes and their frequencies. The percentage of a class is obtained making the ratio between the frequency of the class and the total number of observations.

Modified mode or alternated frequency method

When there are multiple values occurring equally frequently, the function used in the Matlab environment (mode) returns the smallest of those values. In order to

try to avoid to loose some information, when there are two actions performed with the same frequency, they are chosen alternatively during even and odd cycles.
The trick of this modification creates chaos within the logic of the genetic algorithm and in this way it is possible reach convergence with lower error with respect to dynamic programming i.e. with a lower number of generations than the Frequency default method (4.1.5). However this result is strictly related to the current discretization.
The only way to assess which choice is the better one, is evaluating all the choices at the same time and selecting the rule associated to the choice that brings to a reduction in fuel consumption and in the pollutant emissions.
The results obtained with the mode function and modified mode function can be analyzed and compared (Figure 4.6). In this case the simulation is performed over the Nedc cycle with a compact vehicle; the tool is trained over a population of 1500 individuals and the termination condition is reached with a pre-defined number of generations that in this case is equal to 20.
Either functions show a decreasing trend:passing from one generation to the next, it can be seen that the score equals the previous one or improves.The balance between two scores is due to the fact that the best individual is always selected at the end of each generation.

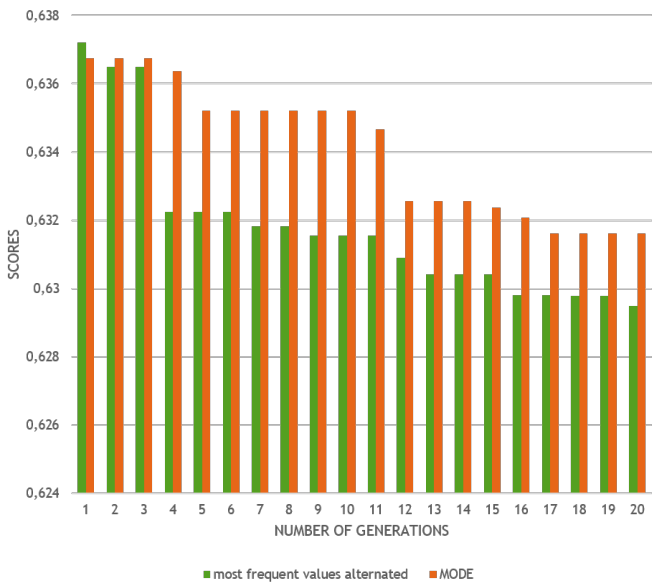


Figure 4.6: Comparison between the mode function and the modified mode function

Random choice

In the Matlab environment the random choice is performed using a function called "datasample". This function selects the rule randomly between all possible actions done in that specific cluster. This is done for all clusters and all individuals.

- Comparable results;
- sometimes a result can be greater than the previous one;
- Unpredictable results: the final score obtained could be better than that obtained with the mode function.

The results obtained with the mode function and the datasample function can be analyzed and compared (Figure 4.7). In this case the simulation is performed over the Nedc cycle with a compact vehicle; the tool is trained over a population of 1500 individuals and the termination condition is reached with a pre-defined number of generations that in this case is 20.
The datasample function shows a not-decreasing trend and its results can not be predicted at priori: at the end of the twenty generations we get a better score than the one obtained with the mode function, but this is only a case.

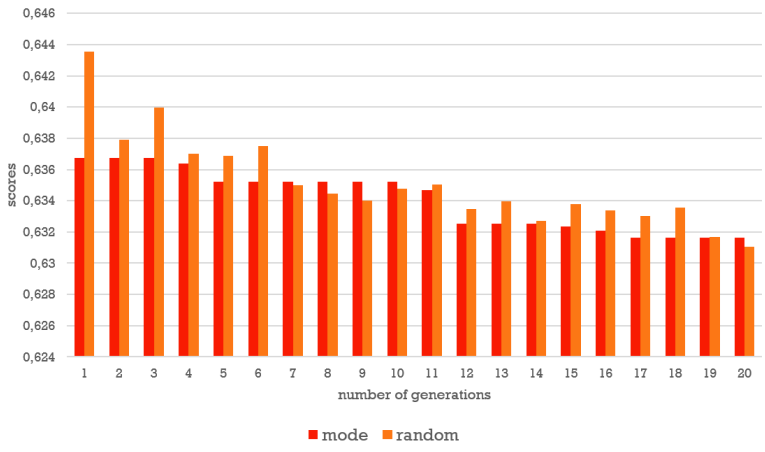


Figure 4.7: Comparison between the mode function and the datasample function

Conditional probability

Instead of selecting the most frequent action, the choice is performed considering the frequency of occurrence of each action. The reason behind this choice is try to not

loose information coming from dynamic programming by considering all those actions performed by dp within a given portion of space (even if that action is not the most frequent one). In the Matlab environment this is realized with the datasample function (by changing some options) that samples with probability proportional to the likelihood of each action. In fact the probability can be defined using the relative frequency. Definition: The probability of a particular outcome is the proportion of times that outcome would occur in a long run of repeated observations [9]. This kind of choice has been implemented in order to reproduce a theory that recalls the Bayesan one i.e considering the prior and the posterior probability of each possible choice in order to perform a choice based on the probability: in this way we obtain a set of results that could consider in a given cluster also the choices performed with a lower frequency than the most frequent one.

4.1.7 Sensitivity on the number of individuals

Different classes of individuals are analyzed. These classes are labeled as low, medium and high classes. In the first version of the code it was possible to change the size of the class at the beginning of the simulation: to each class different parameters were connected:

- size of vehicle's components;
- vehicles miles traveled;
- e-road penetration, e-road charge and e-road efficiency;
- fuel/battery and electric energy price;
- initial battery charge;
- number of individuals;
- number of generations.

The choice of the class results in different final score and so CO_2 ttw emissions mainly because by changing the class, the size of the component changes. Also the computation time was affected by the class' choice. This kind of implementation was adopted only for the compact vehicle and was useful to compare different classes with only only one command.
The choice of the number of individuals affects the computation time and capability of the genetic algorithm to improve. This means that a low class could result in

a fake convergence due to candidate solutions too much similar: in this case the capability of the algorithm is not properly tested. The choice of the number of generations affects the convergence: a genetic algorithm is thought to improve as the number of generations increases. The population selected consists of 1500 individuals, its size is selected to be two order of magnitude greater than the genome length.Several population size are analyzed, from 250 individuals up to 1500. For each class of individuals, ten iterations are performed.

4.2 ACORE

The acronym ACORE stands fore Adaptive-CORE. It has been developed to try to overcome the drawbacks of the Core tool in fact it does not require to know at priori the driving cycle. The procedure implemented aims to recognize the actual unknown driving mission associating it to known driving cycles, already trained and validated through the Core tool. The logic behind the algorithm can be summarized in a sequence of steps as depicted in Figure 4.8.

Figure 4.8: Acore flow chart

4.2.1 Definition of training driving cycles' set

At the beginning of the procedure, different training cycles are selected:

- Nedc, New european driving cycle.
- Ardc, Artemis Rural Driving Cycle
- Wltp, Worldwide Harmonized Light Vehicle Test Procedure
- Whvc, World harmonized vehicle cycle

- Etc, European transient cycle

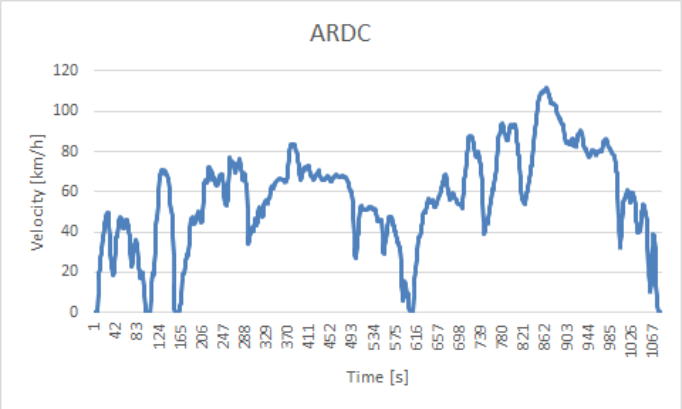


Figure 4.9: Ardc driving cycle

The Artemis project (Figure 4.9) is a statistical study performed in Europe and it mainly consists of 3 parts, urban, rural and motorway. These cycles are only used to understand the real driving conditions and so the vehicle's performance and not for pollutant certification.

The European transient cycle (ETC 4.10) is used for heavy-duty emission certifica-

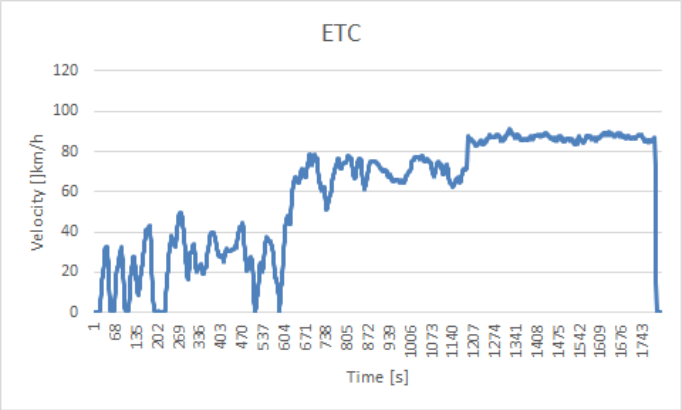


Figure 4.10: Etc driving cycle

tion. It is characterized by real road cycle measurements. It mainly consists of three parts, urban, rural and motorway. The first part is characterized by a maximum speed of 50 km/h, frequent starts, stops and idling. The average speed of the rural and of the motorway part is respectively about 72 km/h and 88 km/h.

The world harmonized vehicle cycle (WHVC, Figure 4.11) is a chassis dynamometer test. It consists of three parts (urban, rural and motorway). The urban part

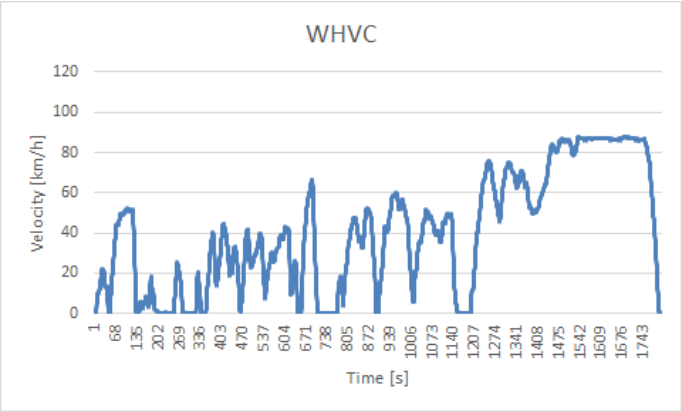


Figure 4.11: Whvc driving cycle

(900s) presents an average speed of 21.3 km/h and a maximum speed of 66.2 km/h. The rural part (481 s) presents an average speed of 43.6 km/h and a maximum speed of 75.9 km/h. The motorway part (419s) is characterized by an average speed of 76.7 km/h and a maximum speed of 87.8 km/h.
The Nedc cycle (Figure 4.12) was designed in 1980s and it was used as reference cycle for vehicles homologation in Europe. It consists of four urban parts called Ece and an extra-urban part called Eudc. It was criticized by experts because it did not represent real life driving conditions: it is characterized by soft accelerations, there are a lot of constant speed parts and idle events. The Nedc cycle has been

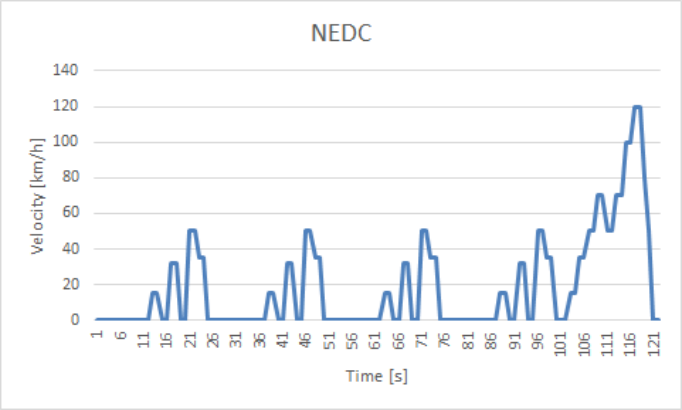


Figure 4.12: Nedc driving cycle

substituted by the Wltp cycle(Figure 4.13) that applies from September 2017. The Wltp test is performed on a chassis dynamometer. It is divided into 3 classes, depending on the power to mass ratio of the tested vehicle. The Europe zone belongs to the Class3, with a PMR $\geq$ 34. This cycle is characterized by four zones: one

representative of urban driving, one of sub-urban driving, one of extra-urbn driving and a highway zone.
The main differences between the Wltp and the Nedc cycle are summarized in Table 4.2.

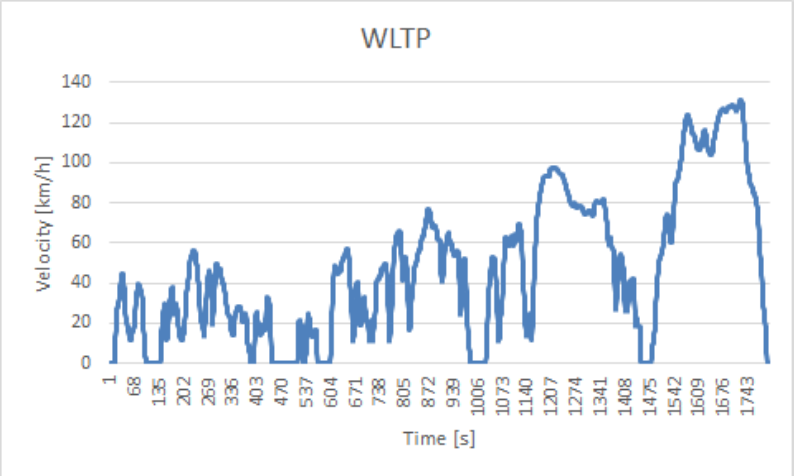


Figure 4.13: Wltp driving cycle

	NEDC	**WLTP**
Test cycle	Single test cycle	Dynamic cycle (more representative of the actual driving conditions)
Cycle time	20 minutes	30 minutes
Cycle distance	11 Km	23.25 Km
Driving phases	2 phases, 66% urban and 34% non-urban	4 phases, 52% urban and 48% non-urban
Average speed	34 km/h	46.5 km/h
Gear shift	Vehicles have fixed gear shift points	Different gear shift points

Table 4.2: NEDC-WLTP comparison [19], [20]

These cycles are divided into segments of equal size. The size is selected at the

beginning of the algorithm and stored within a variable called timestep. For each segment, several parameters are evaluated and used to make the comparison between cycles [12] [13]:

- mean, maximum and minimum speed;
- mean, maximum and minimum acceleration;
- number of positive and negative accelerations;
- idle times;

In the Matlab environment each cycle is stored within matrix. Each row corresponds to a segment. For each segment the parameters defined before are evaluated and stored within a structure where each field is linked to a different parameter. In turn, each parameter is linked to each cycle. This matrioska matrix can be summarized in Equation 4.8.

$$\text{MATRIX} = \begin{cases} \text{velocity} \begin{cases} \text{Mean} \begin{cases} \text{Nedc} \\ \text{Ardc} \\ \text{Wltp} \\ \text{Whvc} \\ \text{Etc} \end{cases} \\ \text{Maximum} \{ ... \\ \text{Minimum} \{ ... \\ \text{Idle times} \{ ... \end{cases} \\ \text{acceleration} \begin{cases} \text{Mean} \{ ... \\ \text{Maximum} \{ ... \\ \text{Minimum} \{ ... \\ \text{Number of positive acceleration} \{ ... \\ \text{Number of negative acceleration} \{ ... \end{cases} \end{cases} \tag{4.8}$$

For example, considering the mean velocity of the Nedc cycle, it results to be a matrix with one row and a number of columns equal to the number of segments: in the first position it is stored the mean value of the first velocity segment.
These values are compared with the unknown driving mission whose cycle data are acquired with a clock of one second. Starting from the time instant equal to the step size, the cycle recognition is done considering a number of instants equal to the

step size and prior to the actual time instant. The parameters defined before are evaluated on this window and compared with the segments of all driving cycles.
The actual cycle parameters are compared with the parameters of all the training driving cycles' segments.

$$d_{param} = v_{param,cyc,i} - v_{param,actcyc} \text{ where } i = 0, ..., N_{Segm,Cyc} \tag{4.9}$$

For each cycle, these values are stored within matrix whose number of rows is equal to the number of parameters previously defined while the number of columns are equal to the number of segments of the specific training driving mission considered. So the first column represent the first segment while the in the first row it is stored the difference between the mean speed of the cycle's first segment and the mean speed of the segment of the actual cycle; in the second row the difference between the maximum speed of the cycle's first segment and the maximum speed of the actual cycle segment and so on.
All the parameters are assumed to be equal in weight. For each matrix, it is evaluated the mean value of each column and it is selected the minimum one. At this point it is obtained a vector containing a number of values equal to the training cycle: these values are compared and the minimum one identifies the training cycle closer to the actual one.
Two different approaches for the division of cycles into segments are analyzed and compared, namely A Solution (Segment approach) and B solution(Sliding window approach).
In the A solution, each cycle is divided into segment whose length is equal to the timestep: for example, if the stepsize is equal to 11 seconds, the segment one covers the portion that goes from the first instant to the 12th ones, the second segment from the 13th to the 24. Considering the time duration of each cycle,the number of segments in which each cycle is divided is obtained by dividing the time duration of the cycle with the timestep:

$$N_{segm} = \frac{D_{t,cyc}}{timestep} \tag{4.10}$$

The number of segment obtained with the *Segment approach* is reported in Table 4.3.

The total number of segments resulting from solution A with a timestep of 11 seconds is 714. In the B solution, each cycle is divided in segments using a sliding window that proceeds throughout the cycle: the segment one goes from the 1st time instant up to the 12th, the second segments from the 2nd time instant up to the 13. Considering the time duration of each cycle,the number of segments in which each cycle is divided is obtained by subtracting the time step to the time duration of the cycle:

$$N_{segments} = D_{t,cycle} - timestep \tag{4.11}$$

	N_{segm}
Nedc	127
Ardc	98
Wltp	163
Whvc	163
Whvc	163

Table 4.3: Definition of A Solution's number of segments

The number of segment obtained with the *Sliding window approach* is reported in Table 4.4. The total number of segments resulting from solution A with a timestep

	N_{segm}
Nedc	1169
Ardc	1071
Wltp	1789
Whvc	1789
Whvc	1789

Table 4.4: Definition of B Solution's number of segments

of 11 seconds is 7607.

4.2.2 Current driving cycle evaluation

The current driving cycle is evaluated starting from the time instant equal to the time step, considering a historical window equal to the time step. On this window, the parameters defined before are evaluated and compared with all the parameters of all the training driving cycles.
The two approaches are compared by testing them in the recognition of the training cycles. The selected cycle are the Ardc, Etc, Wltp. The training set consists of five driving cycles, Nedc, Ardc, Wltp, Whvc, Etc. The simulation times are reported in Table 4.6.

	Solution A	Solution B
Etc	82s	503s
Wltp	82s	493s
Ardc	49s	273s

Table 4.5: Simulation time comparison

It can be noted that:

- solution B the most effective in all three cases;
- the simulation time is less than the cycle time duration.

Considering the success rates obtained with the sliding window approach, from now on it is the only solution considered.
Figures 4.14 , 4.15, 4.16 represent the results obtained with the cycle recognition algorithm in three cases i.e in the recognition of the Etc driving cycle, Wltp and Ardc.

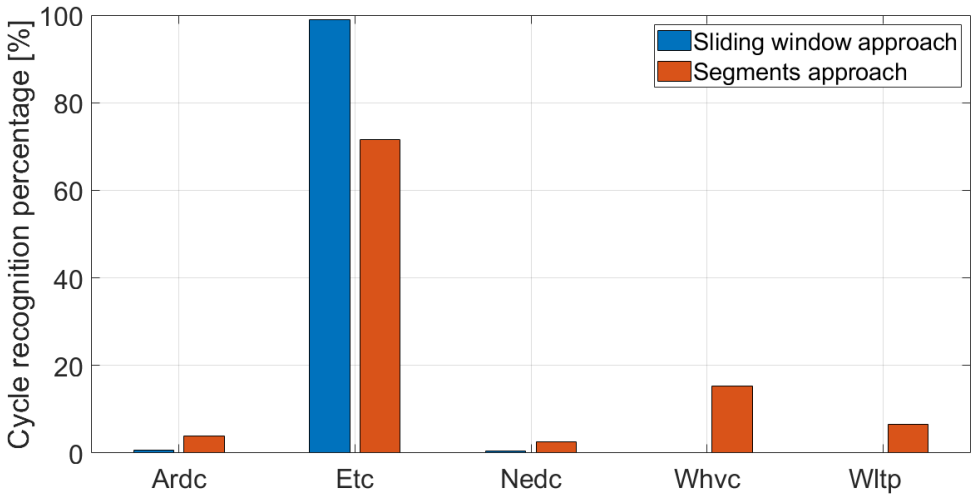


Figure 4.14: Etc cycle recognition

The Wltp cycle recognition, Figure 4.16, results to be successful at 91,2% while the remaining 9 % is splitted between Ardc (0.6 %) and Nedc (8.2%)cycles. As can be observed in Figure 4.17, the algorithm selects the Nedc cycle in a specific time window. By comparing the Nedc and the Wltp velocity profiles in that specific time window, it can be noted that both cycles have null velocity profiles (Figures 4.18 and 4.19). For the Wltp and Etc cycles, the success rate is over 90 percent, while in the Ardc case, 100 percent is obtained. It can be seen that by enlarging the time window within which the cycle parameters are evaluated, the ability of the algorithm to recognize one of the training cycles improves (Figure 4.20). In Figure 4.21 the results obtained by enlarging the window size in the Wltp cycle recognition are showed. By enlarging the window size, the algorithm efficiency improves.

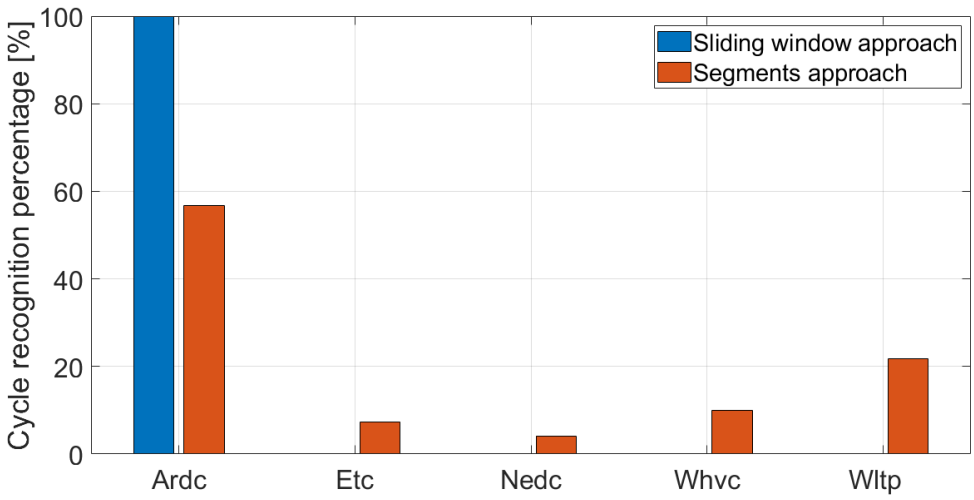


Figure 4.15: Ardc cycle recognition

step size	11 s	20 s
Etc	503s	451s
Wltp	493s	420s

Table 4.6: Simulation time comparison

4.2.3 Rule assignment

The five training cycles of the training set are validated on different trainings on the core tool. The results in terms of CO_2 tank to wheel are analyzed and for each cycle of the training set, it is selected the training of core at which it answers better.
In the Matlab environment this is realized by means of matrix where each row represents the results in terms of CO_2 ttw obtained during the validation phase of the Core tool and each column represents one of the training cycle used for the pattern recognition.
The Core tool has been validated over single trainings performed on the same set of driving cycles used for the driving cycle recognition. Because these training do not result to be feasible for the entire set, by means of a feasibility check all unfeasible results are discarded.
The rules and the relative discretization of the selected training of core are applied to the current time instant of the unknown driving cycle.

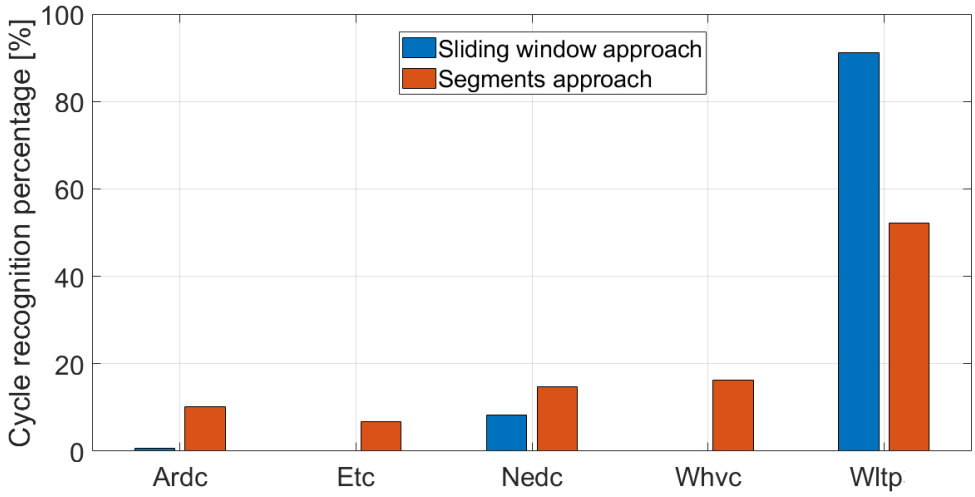


Figure 4.16: Wltp cycle recognition

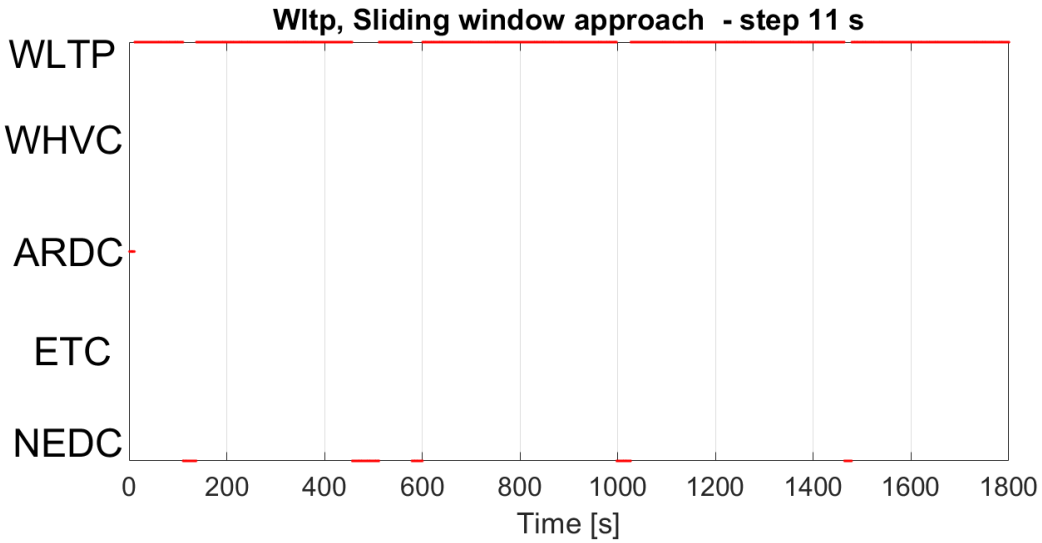


Figure 4.17: Wltp cycle recognition

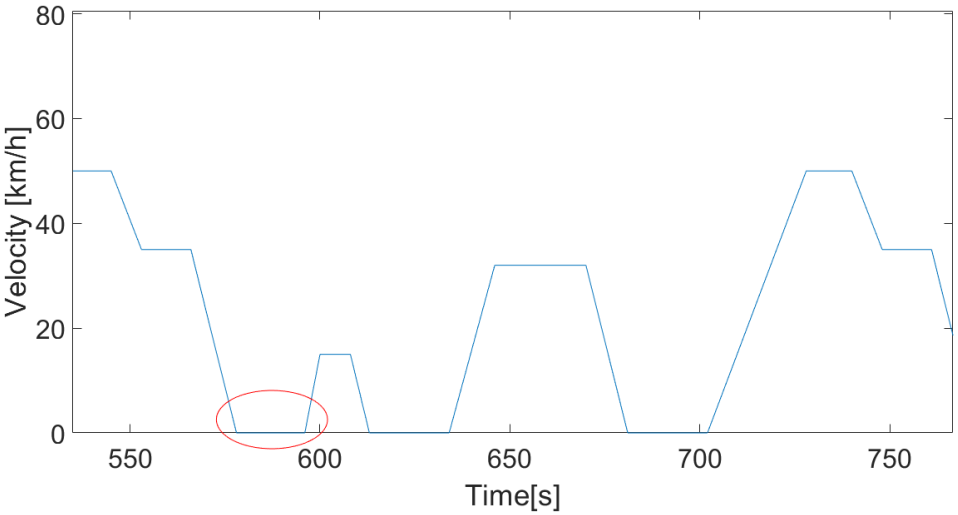


Figure 4.18: Zoom on the Nedc cycle

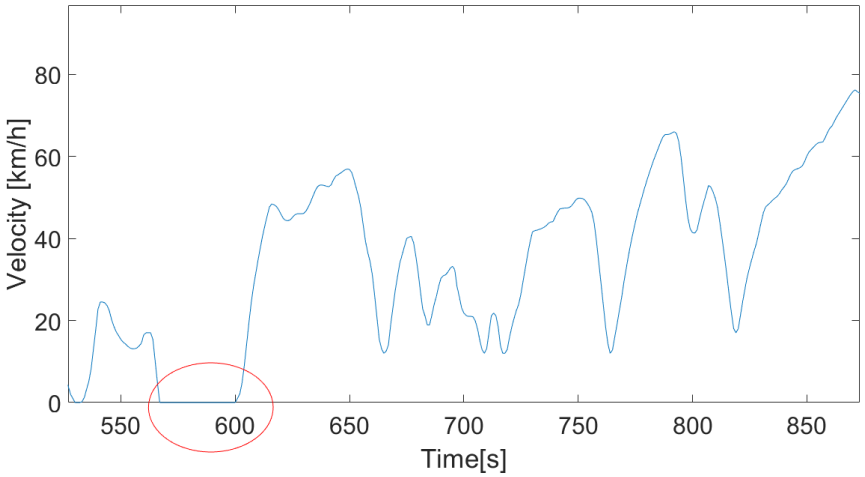


Figure 4.19: Zoom on the Wltp cycle

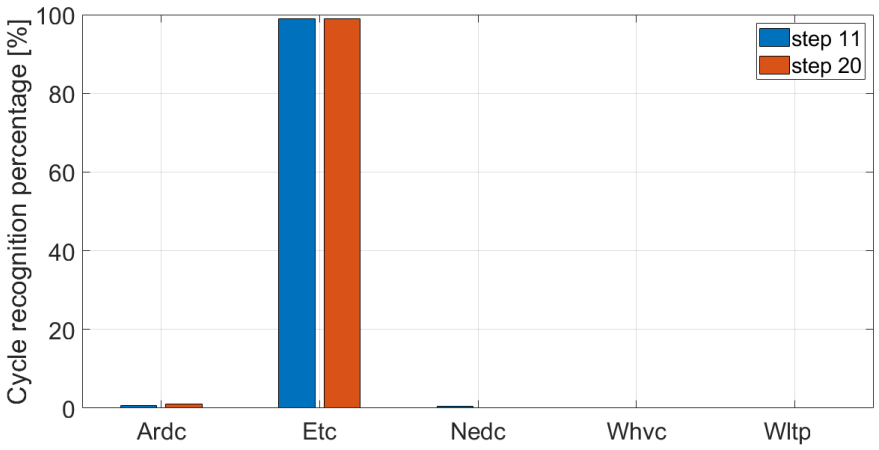


Figure 4.20: Comparison between different time steps applied on Etc cycle

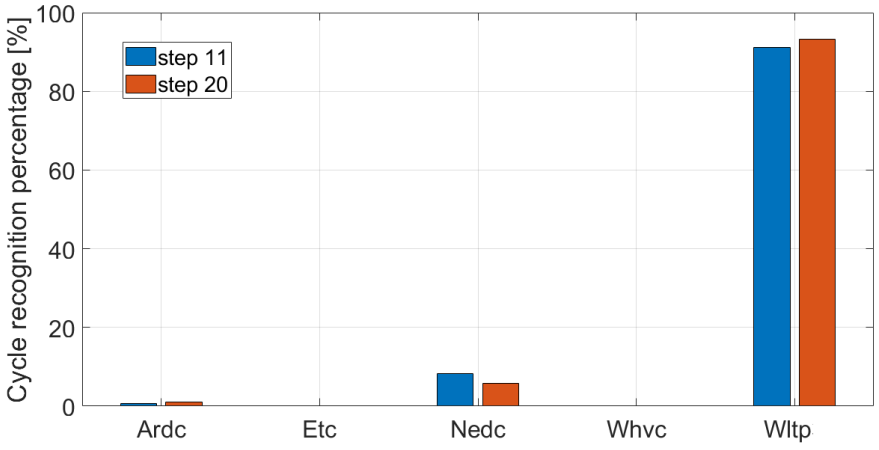


Figure 4.21: Comparison between different time steps applied on Wltp cycle

Chapter 5

Results

5.1 Core results

Genetic algorithm performance is affected by several parameters such as the population size, the crossover and mutation factors.

5.1.1 Population size

The performance of the CORE tool using different population sizes has been investigated. Three classes of individuals have been defined. The first class, labeled as L class, consists of 500 individuals and ten generations. The second class, labeled as M class, consists of 1500 individuals and 30 generations. The third class, labeled as H class, consists of 2500 and 30 individuals. The analyses have been performed using the default parameters.
The clustering optimization algorithms employs velocity, velocity variation and SOC as input variables. The discretization used is [8 9 3]. The crossover factor imposed is equal to 0.7 while the portion of modified genome is equal to 0.1. The offspring has been generated using the parent selection method: the genes of the selected parents are mixed in a random way by flipping a coin.
The rule extraction algorithm uses the Default frequency method: once the mesh is created, for each cluster the most frequent action performed by the dynamic programming is selected and becomes the unique action.
The simulations have been performed over the Whvc driving cycle and the vehicle selected is an heavy duty. The results are showed in Figure 5.1. The scores in terms of kg of CO_2 tank to wheel have been compared.

The low class results to be the most inefficient one: the convergence is reached not because the genetic algorithm actually improves but only because there are too similar solutions (too similar individuals).
As suggested by literature, the population dimension is chosen to be two order of

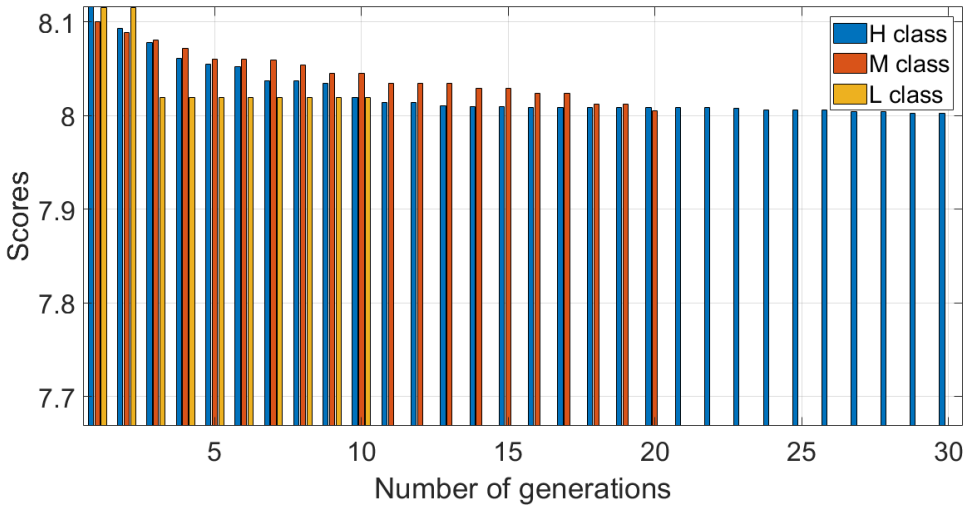


Figure 5.1: Class comparison

magnitude greater than the length of the genome . The M class results to be a good compromise in terms of simulation time and ga improvements.
The H class shows results very close to the medium class but it has been discarded because of simulation time requirements. The optimal solution is represented by the results in terms of kg of CO_2 ttw obtained by the benchmark optimizer (dynamic programming). As can be seen from Figure 5.2 the L class shows a linear descending trend reaching convergence after few number of generations. The M class shows a descending trend obtaining better results than the L class; passing from one generation to the successive one, it is observed an improvement of the GA.

5.1.2 Rule extraction methods comparison on M class

The rule extraction algorithm has been tested by using three different methods as explained in 4.1.6 . In the *Frequency default method* for each cluster, the most frequent action selected by dynamic programming is chosen, becoming the unique action for that cluster.
In the *Alternated frequency method*, the frequency of occurrence of dp actions is evaluated for each cluster. The action selected is always the most frequent one but in this case, if two actions are performed with the same frequency, they are alternatively chosen during even and odd generations. On the contrary, in the *Alternated frequency method*, if two actions are performed with the same frequency, the selected action is always the one with the lowest index.
In the *Probabilistic method* the unique action for each cluster is computed considering

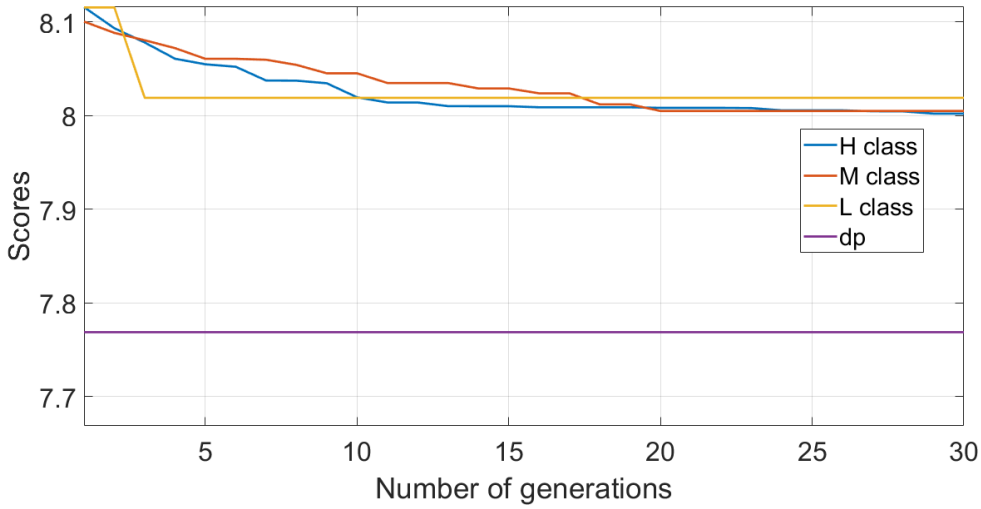


Figure 5.2: Class comparison

the likelihood of occurrence of the actions performed by the benchmark optimizer. The main parameters of the GA are maintained constant: the crossover factor is set equal to 0.7, the portion of the genome mutated is equal to 0.1, the offspring is generated by applying the Parent selection approach. The results obtained on the M class are showed in Figures 5.3, 5.4. As can be noted, the *Probabilistic method*

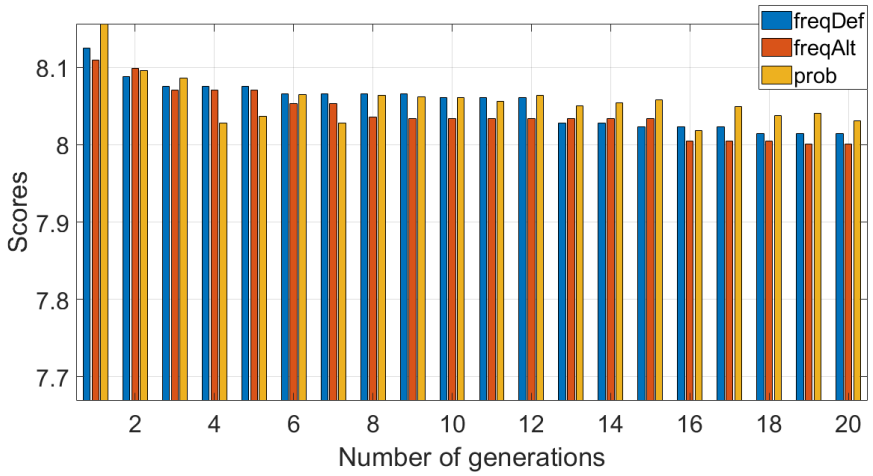


Figure 5.3: Choices comparison on medium class

shows a not-descending trend; passing from one generation to the next, higher scores can be obtained. This method has been discarded because of its unpredictable results and because the selected control strategy might be not optimal in terms of Soc

constraints, obtaining in this way unfeasible results.

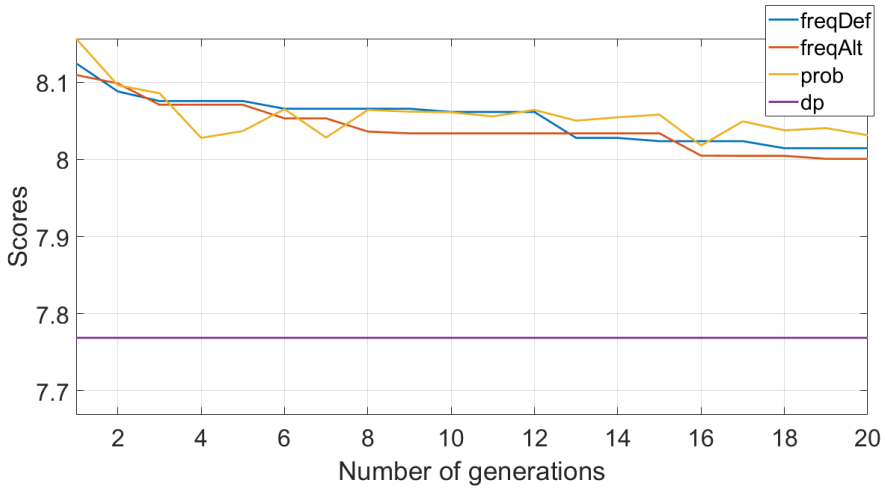


Figure 5.4: Choices comparison on medium class

The optimal solution is represented by the results in terms of kg of CO_2 tank to wheel obtained by the benchmark optimizer (dynamic programming). As can be noted from Figure 5.4, the *Frequency alternated method* is the most efficient. It shows a descending trend and it reaches better final scores than the *Frequency default method*. However, the scores depend on the grid size information: these results are obtained starting from three different initial populations, randomly generated. The relative error with respect to the dynamic programming is equal to 3.5 %.

5.1.3 Sensitivity analyses

In order to better investigate the performance of the rule extraction method, several sensitivity analyses have been performed, changing the number of individuals and generations. Each simulation has been repeated for 10 iterations, keeping constant the rule extraction method, the number of generation and the number of individuals. The crossover factor is fixed to 0.7 while the portion of the modified genome is equal to 0.1. The offspring is generated by using the Parent selection method. The selection has been made by using the roulette wheel selection.

250 individuals and 160 generations

The first trial has been performed with a population of 250 individuals that evolves for 160 generations. The results are showed in Figures 5.5, 5.6 where the score trend of ten iterations as a function of the number of generations is depicted. As

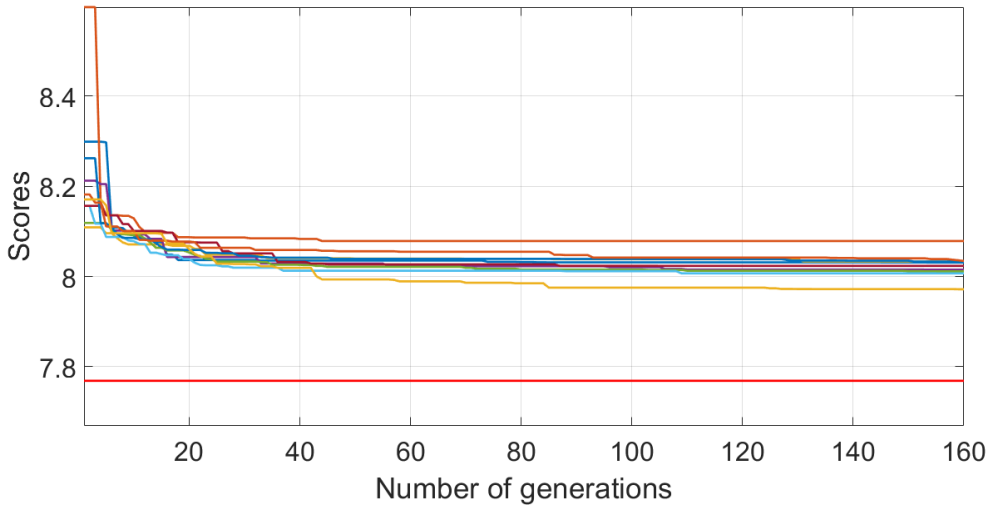


Figure 5.5: 250 individuals, 160 generations, Frequency default method

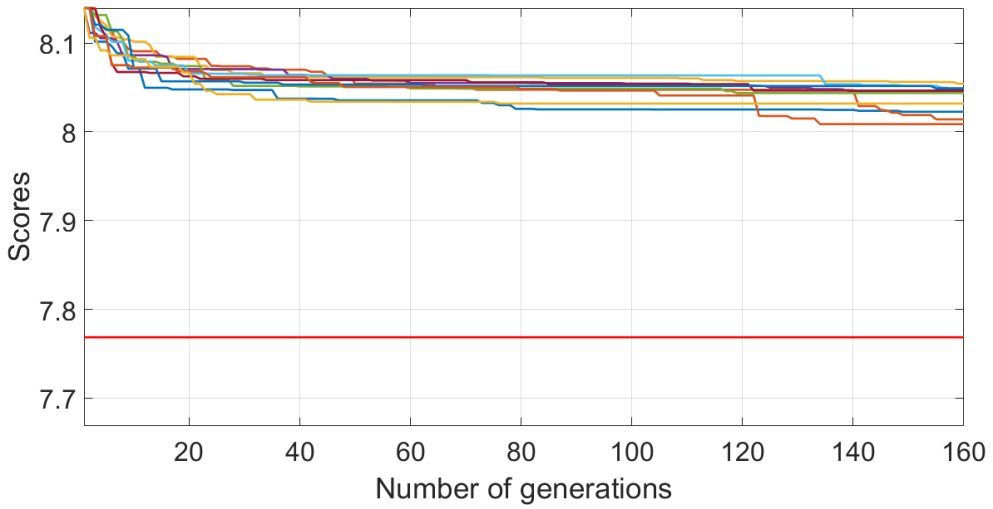


Figure 5.6: 250 individuals, 160 generations, Alternated Frequency method

can be seen the dynamic programming result (represented by the red line) is not reached by neither of the two methods. With the frequency default method, one of the trials score falls under 8 threshold but the others reach convergence after few

generations. The alternated frequency method shows a always decreasing trend in all the trials and there are lower trend part at horizontal tangent than those present in the frequency default method.

700 individuals and 50 generations

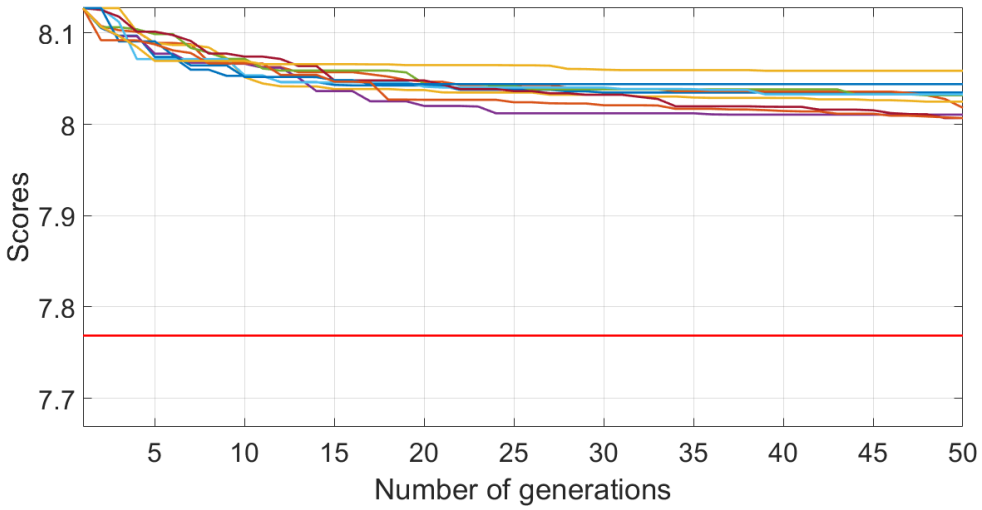


Figure 5.7: 700 individuals, 50 generations, Frequency default method

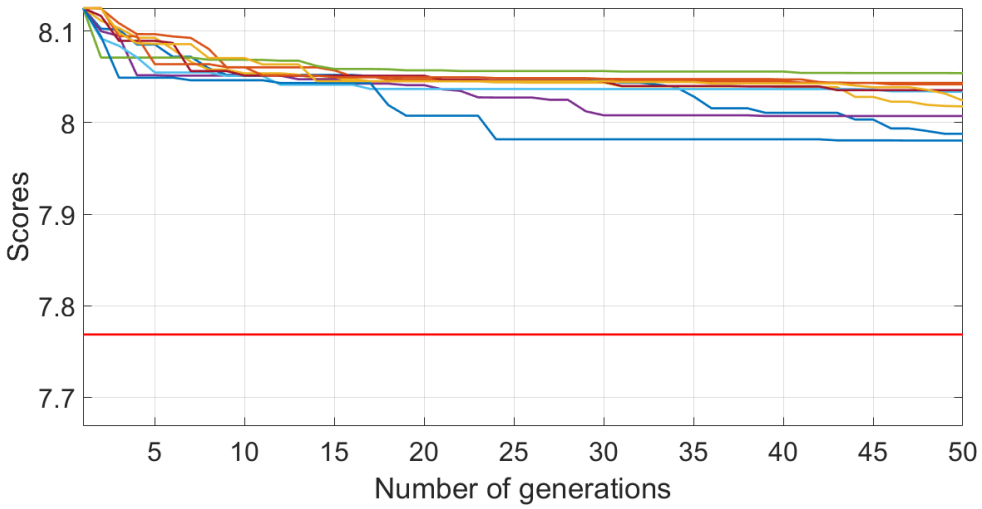


Figure 5.8: 700 individuals, 50 generations, Alternated frequency method

The alternated frequency method (Figures 5.7, 5.8) results to be more efficient than the frequency default method: it reaches lower final score in one of the trials

while the score trend is always descending. The horizontal tangent parts are more extended with the first method.

1500 individuals and 30 generations

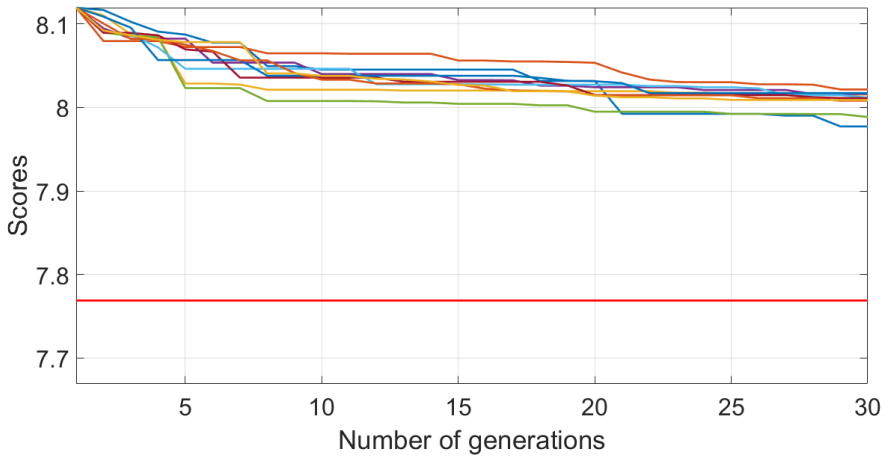


Figure 5.9: 1500 individuals, 30generations, Frequency default method

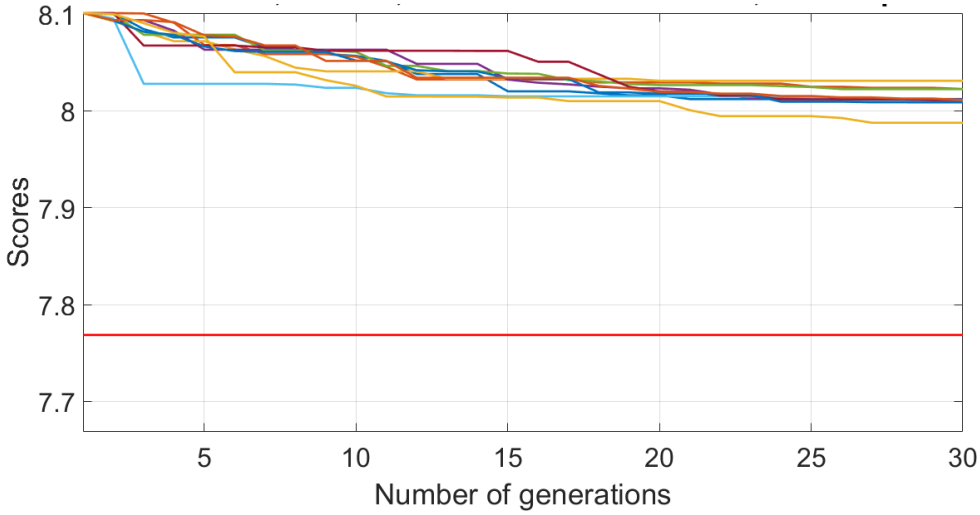


Figure 5.10: 1500 individuals, 30generations, Alternated Frequency method

5.1.4 Different discretization

A different discretization has been investigated over the Whvc driving cycle with the M class population size. The sensitivity has been made changing the rule extraction

method. The alternated frequency method results to be the most efficient, showing

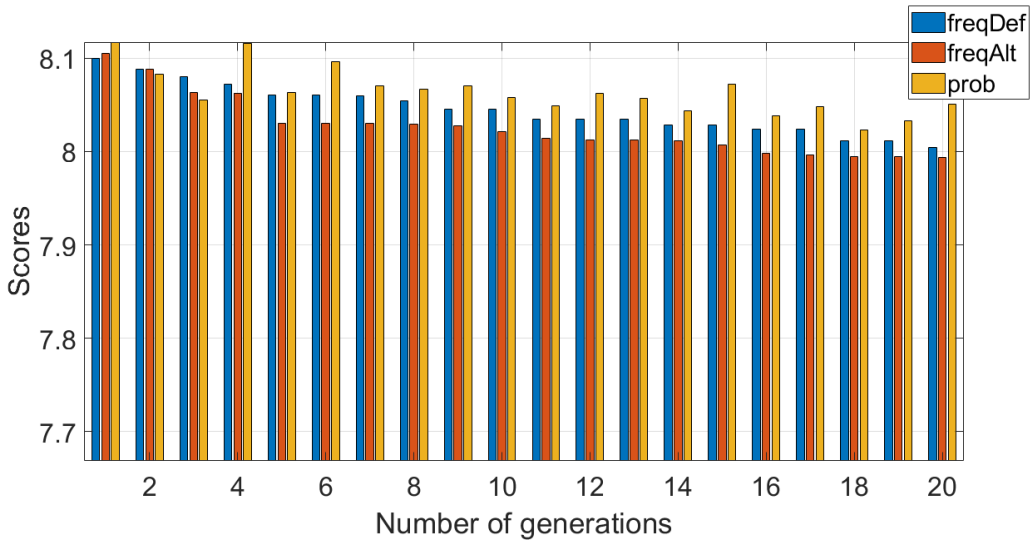


Figure 5.11: Different discretization - M class

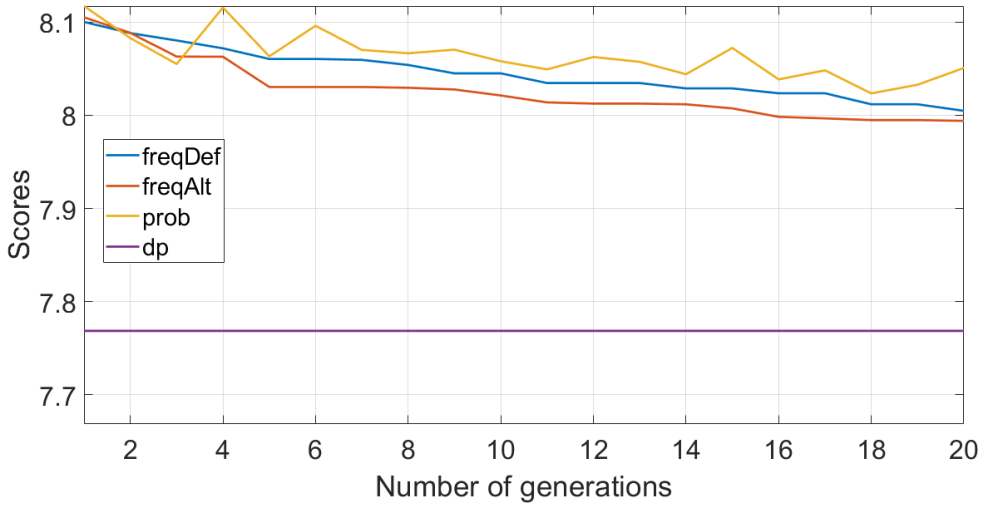


Figure 5.12: Different discretization [19 21 3]- M class

a always descending trend. Also the frequency default method shows a decreasing trend but it reaches a higher final score than the alternated frequency method. The probabilistic approach shows a not decreasing trend as assessed in 4.1.6. As can be seen, the obtained final score is very close to the score obtained with the previous discretization.

5.1.5 Sensitivity on crossover factor

The crossover factor determines the number of individuals affected by crossover. In the graph depicted in Figure 5.13, the results obtained imposing different crossover factors are showed. As can be seen, the smaller the crossover factor, the faster convergence is reached because the population is made up of very similar solutions. The simulations are performed over a population of 1500 individuals that evolves for 30 generations. The best result is obtained using a crossover factor of 0.7 even if it is very closer to the result obtained with a factor equal to 0.8. The simulations are performed over the Nedc driving cycle and the selected vehicle is an heavy duty.

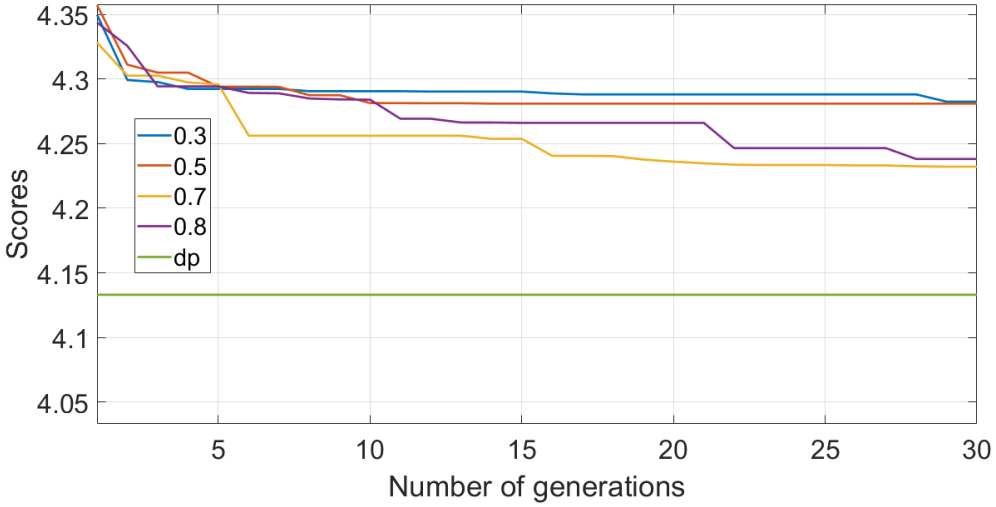


Figure 5.13: Crossover factors comparison

5.2 Acore results

In Figures 5.14, Clust 8 cycle recognition the results obtained with the Acore tool are depicted. The driving cycle analyzed belongs to the Clust family i.e driving cycles representing real driving conditions. Each cycle is divided into segments using the Sliding window approach and it is compared with five cycle, Nedc, Ardc, Wltp, Whvc and Etc. Figures 5.16 and 5.17 show a zoom of the Clust 2 and Etc driving cycles. As can be noted in the figure 5.15, in the portion in between 400 s and 425 s, the algorithm selects the Etc driving cycle. Comparing this results with the velocity profile in Figures 5.16 and 5.17 ranges approximately around 40 km/h for both cycles. Figures 5.20 and 5.21 show a zoom of the Clust 8 and Nedc driving

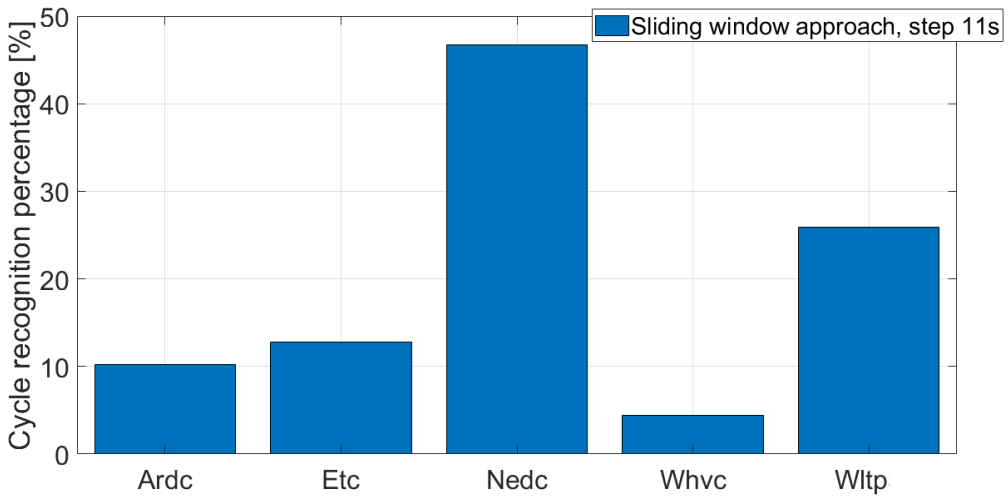


Figure 5.14: Clust 2 cycle recognition

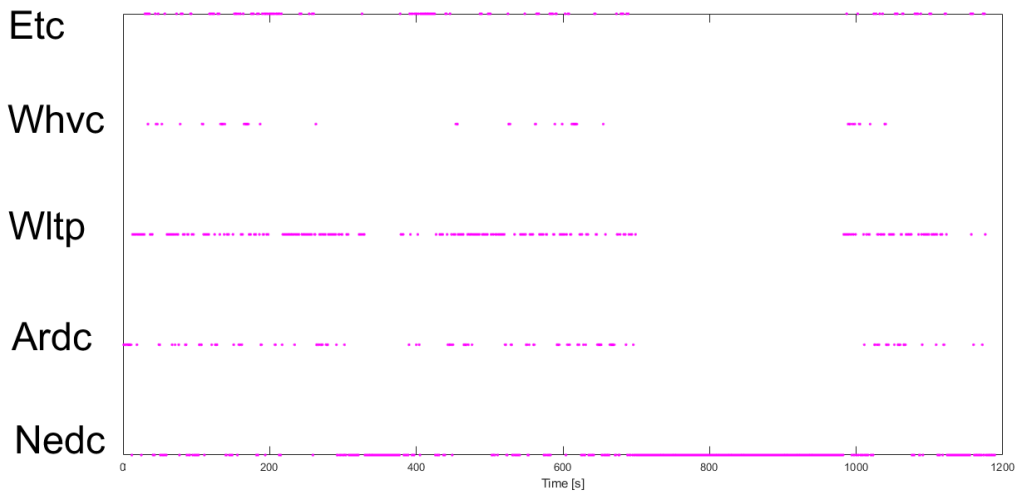


Figure 5.15: Clust 2 cycle recognition

cycles. As can be noted in Figure 5.19, in the portion in between 1000 s and 1180 s, the algorithm selects Nedc driving cycle. Comparing this results with the velocity profile in Figures 5.20 and 5.21 ranges approximately around 70 and 85 km/h for both cycles.

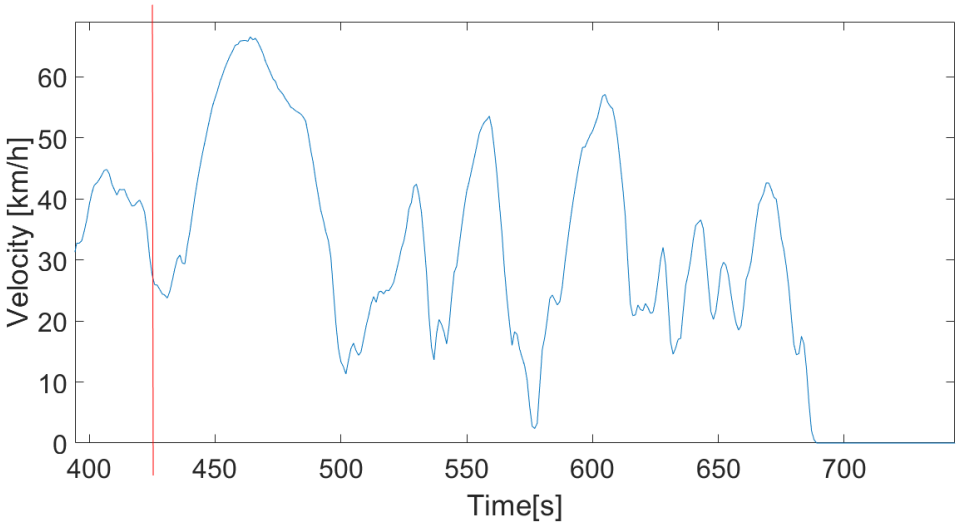


Figure 5.16: Clust 2 zoom

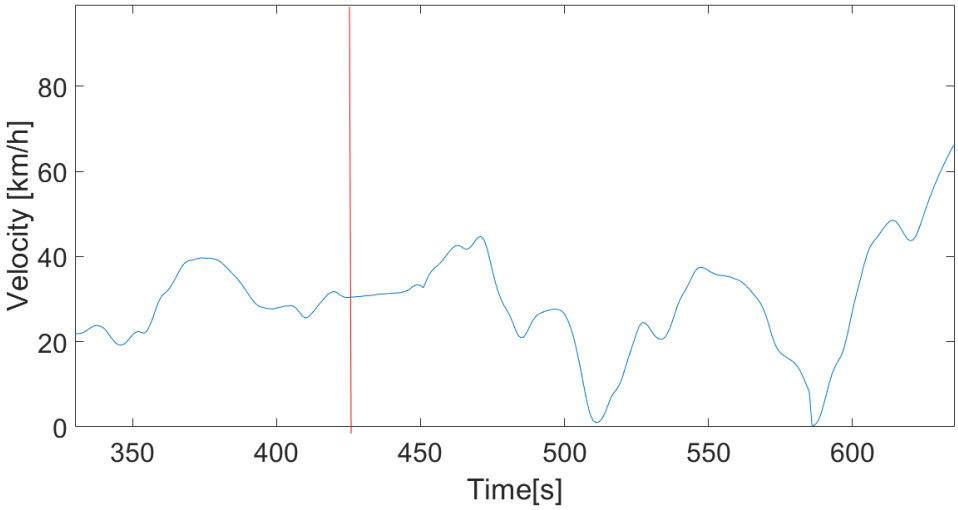


Figure 5.17: Etc zoom

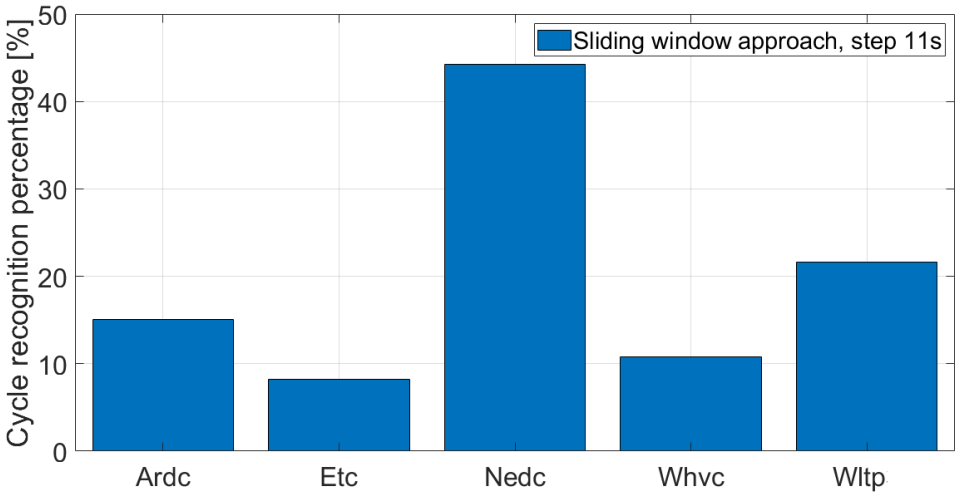


Figure 5.18: Clust 8 cycle recognition

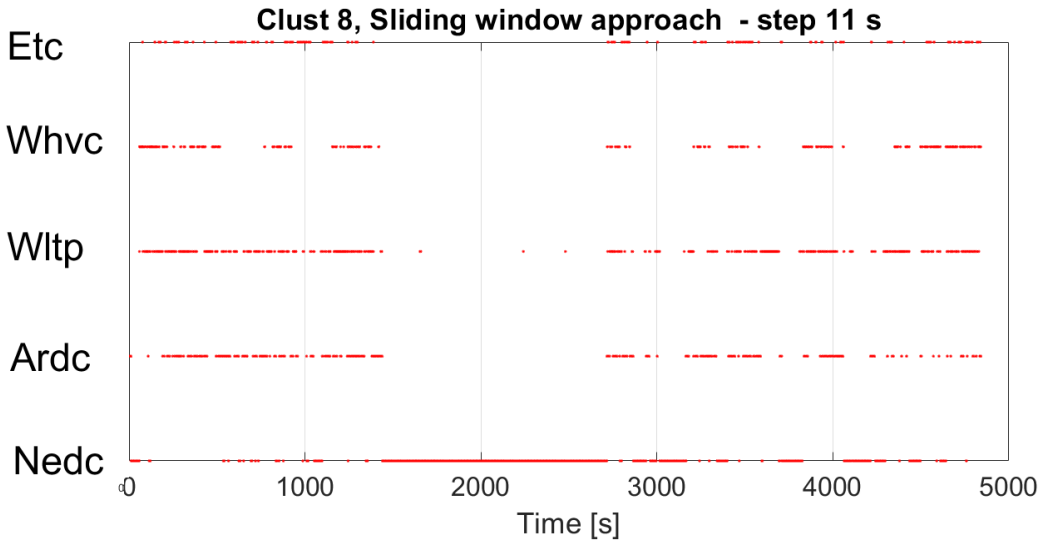


Figure 5.19: Clust 8 cycle recognition

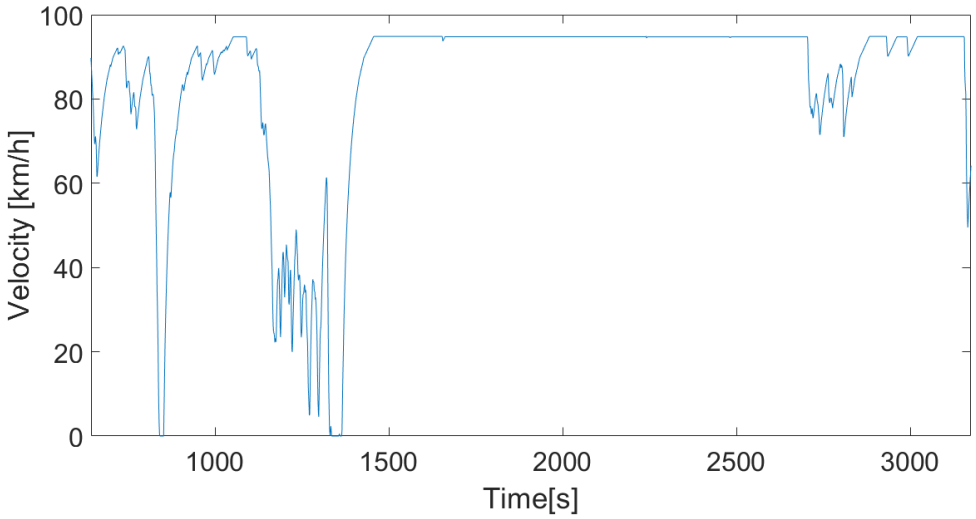


Figure 5.20: Clust 8 zoom

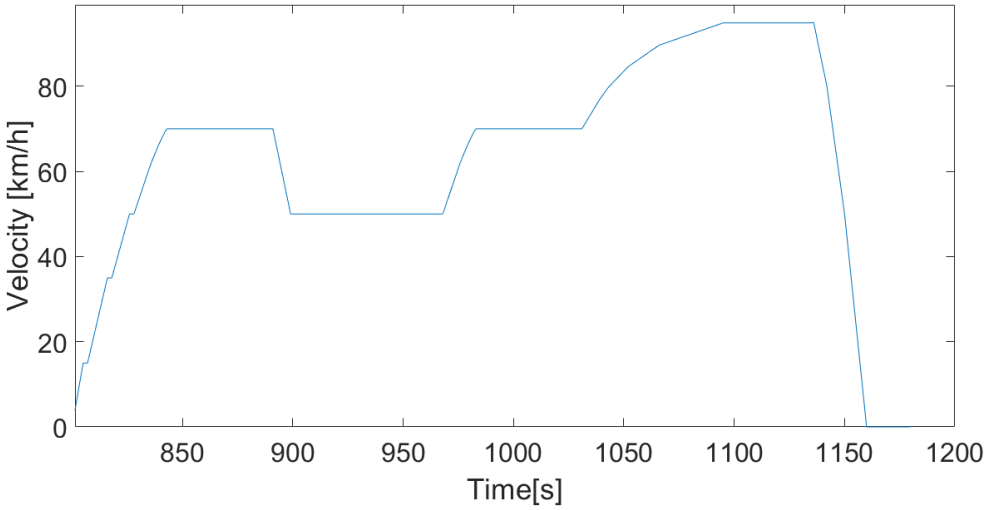


Figure 5.21: Zoom on Nedc cycle

Chapter 6

Conclusions and future works

6.1 Conclusions

This study is focused on the development of an adaptive real-time control strategy for (plug- in) hybrid electric vehicles. The simulated vehicles are equipped with a compression ignition engine. In the first part of this study, a real time optimizer, namely Core [4], has been analyzed and used to develop a new rule based control strategy implementable on-board, Acore. This controller aims to select in real time the optimal control strategy in terms of gear number and powerflow. It compares an unknown driving mission with a set of known driving cycles for which optimal set of rules is previously identified. At each instant, once the cycle recognition is made, the set of rule is applied to the unknown driving cycle. It has been tested over several driving cycles such as Nedc, Ardc, Wltp, Etc.
The core tool is used to compute a 3d map discretized using the input variables (velocity, velocity variation, Soc), generating in this way a mesh of cluster. The rule, in terms of combination of powerflow and gear number, has to be selected among the alternatives proposed by the dynamic programming: considering that each cluster occupies a portion of space, in that portion the dynamic programming could perform several actions. It has been investigated the way in which the actions performed by the global optimizer are selected.
The main findings of this activity can be summarized as follows:
Rule extraction Two approaches have been implemented in order to explore different solutions and try to improve the performance of the Core tool. The results obtained with the Alternated frequency method show not-negligible improvements on the Core tool performance.
Termination condition
The internal solver (GA) used within the Clustering optimization tool has been

analyzed. Different sensitivity analyses have been performed in order to better understand which parameters actually affect the Ga performance. In order to decrease the simulation time, a new termination condition has been proposed.

Acore tool

A new real time optimizer has been developed in order to overcome the Core tool shortcomings. It can be applied in real-time without a priori knowledge of the driving mission.

6.2 Future works

Genetic algorithm

Deep sensitivity analyses of the offspring generation have to be performed, applying new methods as the Stud Ga to try to improve the performance of the genetic algorithm.

Acore

It would be fundamental to investigate an alternative set of parameters for the driving cycle recognition.

Bibliography

[1] R. Bellman, *Dynamic programming*,Princeton University Press, 1957.

[2] Wei Liu, *Hybrid electric vehicle system modeling and control*,Wiley, 2017.

[3] Iqbal Husain, *Electric and hybrid vehicles design fundamentals*,2011

[4] M.Venditti, *Innovative Models and Algorithms for the Optimization of Layout and Control Strategy of Complex Diesel HEVs* , 2015.

[5] R. Finesso, E. Spessa, M.Venditti, *An unsupervised Machine Learning Tecnique for the Definition of a rule based control stategy in a complex HEV* SAE International, 2016.

[6] S.Gotshall, B. Rylander, *Optimal population size and genetic algorithm*

[7] Sun Yuan Kung, *Kernel methods and machine learning*, Cambridge University Press, 2014

[8] P.R.Kumar, *Dynamic programming*

[9] J. Isotalo, *Basics of statistics*

[10] S. N. Ghoreishi, A. Clausen, B.N. Joergensen, *Termination criteria in Evolutionary Algorithms: A Survey*

[11] A. Maruyama, R. Kono, Y. Sato, T. Ishizu, M. Koseki, Y. Muranaka, *Automotive lithium-ion batteries*

[12] S. Jeon, S. Jo, Y. Park, J. Lee, *Multi- mode driving control of a parallel hybrid electric vehicle using driving pattern recognition*, Journal of Dynamic Systems, Measurements, and Control, 2002.

[13] H. He, C. Sun, X. Zhang, *A method for identification of driving patterns in Hybrid electric vehicle based on a LVQ neural network* , Energies, 2012.

[14] C.M Bishop, *Pattern recognition and machine learning*,2006

[15] *https* : *//www.tutorialspoint.com/genetic − algorithms*

[16] *https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php*

[17] *https://www.mv.helsinki.fi/home/jmisotal/BoS.pdf*

[18] *http://www.eolss.net/sample-chapters/c18/E6-43-18-05.pdf*

[19] *http://wltpfacts.eu/from-nedc-to-wltp-change*

[20] *http://www.car-engineer.com/the-different-driving-cycles*