EPiC Series in Computer Science

Volume 36, 2015, Pages 196–214

GCAI 2015. Global Conference on Artificial Intelligence

A Genetic Algorithm based Control Strategy for the Energy Management Problem in PHEVs*

Johanna Nellen, Benedikt Wolters, Lukas Netz, Sascha Geulen, and Erika Ábrahám

Department of Computer Science, RWTH Aachen University, Aachen, Germany

{johanna.nellen,sgeulen,abraham}@cs.rwth-aachen.de,
{benedikt.wolters,lukas.netz}@rwth-aachen.de

Abstract

Genetic algorithms have been applied to various optimization problems in the past. Our library GeneiAL implements a framework for genetic algorithms specially targeted to the area of hybrid electric vehicles. In a parallel hybrid electric vehicle (PHEV), an internal combustion engine and an electrical motor are coupled on the same axis in parallel. In the area of PHEVs, genetic algorithms have been extensively used for the optimization of parameter tuning of control strategies. We use GeneiAL to control the torque distribution between the engines directly. The objective function of this control strategy minimizes the weighted sum of functions that evaluate the fuel consumption, the battery state of charge, and drivability aspects over a prediction horizon of fixed finite length.
We analyze the influence of these weights and different configurations for the genetic algorithm on the computation time, the convergence, and the quality of the optimization result. For promising configurations, we compare the results of our control strategy with common control strategies.

1 Introduction

In recent years, the fuel consumption and emission of vehicles have come into focus of society, politics, and the automotive industry. As a result, pure electric vehicles have been placed on the market. However, for these vehicles, there is a trade-off between the possible driving range and the weight of the battery. *Hybrid electric vehicles* (*HEVs*) are equipped with an *internal combustion engine* (*ICE*) and an *electrical motor* (*EM*) and thus benefit from the advantages of both propulsion systems. The ICE allows a wider driving range and is efficient for high torque values. The EM produces no pollutant emission and is efficient for low torque and speed. Thus, the EM can support the combustion engine for a better efficiency.

Here, we consider *parallel hybrid electric vehicles* (*PHEVs*), i.e., the ICE and the EM are coupled on the same axis in parallel. Thus, the engines produce the requested torque for driving

*This work was partly supported by the German Research Foundation (DFG) as part of the DFG research project "OASys" (AB 461/2-1) and the Research Training Group "AlgoSyn" (GRK 1298).
Other people who contributed to this document include Martina Josevski and Dirk Abel (Institute of Automatic Control, RWTH Aachen University, Aachen, Germany).

either separately or in combination. Moreover, the ICE can be used to recharge the battery by generating more torque than requested. Alternatively, electric energy can be recuperated while braking, using the EM as a generator.

A *control strategy* has to distribute the torque requested by the driver between the engines within their specified physical limits. This problem is known as the *energy management problem* for HEVs [6]. In the past, heuristic control strategies (e.g., rule-based control strategies [11]) as well as control strategies based on optimal control (e.g., equivalent consumption minimization strategies [8, 6]) have been proposed. If an estimation on the future driving conditions is available, predictive control strategies can be applied. An overview of different kinds of control strategies is given in [9]. Optimal-control-based strategies use, e.g., dynamic programming [1] or evolutionary programming [10]. Note that an online control strategy has to provide a torque split within a limited computation time and that the control unit of the vehicle has limited processing power. This is the reason why we consider here only real-time capable control strategies, i.e., strategies that provide a new torque distribution every 0.02 seconds.

We introduce a control strategy based on *genetic algorithms* (GA) [5]. It considers multiple objectives: The overall goal is to minimize the fuel consumption. Secondly, our strategy maximizes the available electric energy while keeping the state of charge of the battery near a reference value to ensure a long battery lifetime. Finally, drivability aspects are considered during the optimization to increase the driving comfort (e.g., to reduce the noise emission of the internal combustion engine).

Genetic algorithms have already been applied in the context of HEVs. On the one hand, the sizing of powertrain components has been optimized [2]; on the other hand, the parameters of control strategies have been tuned using genetic algorithms [9]. In [14], a GA-based optimization for the scheduling of the electrical generator has been presented for a series hybrid solar vehicle. However, control strategies that are directly built on genetic algorithms are rare: In [15], a control strategy based on genetic algorithms has been introduced where the optimization is done over the complete driving cycle. Usually, even if the final destination is known in advance, it is difficult to determine a precise prediction for the route to the final destination. Furthermore, due to the computational overhead of an optimization over the whole driving cycle, such a strategy is not real-time capable and can only be used as a reference strategy. In contrast to that, we restrict the optimization to a limited prediction horizon and thus get a real-time capable control strategy.

We implemented a genetic algorithm library called GeneiAL [3] and used it to build a GA-based control strategy. Besides some standard genetic operators that are provided by GeneiAL, the library can easily be extended by customized genetic operators. For our control strategy, we use this feature to create smoothing variants of standard genetic operators to obtain better drivability. The GA-based control strategy optimizes the control by starting a genetic algorithm run for each time step.

We evaluated our control strategy and derived a real-time capable configuration for the genetic algorithm yielding good optimization results. Our control strategy uses a fitness function that computes a weighted sum of a set of objectives. Therefore, we evaluate the influence of various weights on different driving cycles. We show the trade-off between drivability and fuel consumption minimization. Drivability can be obtained by an additional objective in the fitness function or by a search space restriction. We examine the convergence of the genetic algorithm and the influence of the population size and the number of iterations on both the result and the runtime for a set of configurations.

The paper is organized as follows. In Section 2 the vehicle model, the energy management problem, and the notation used in the context of genetic algorithms are introduced. Our GA-

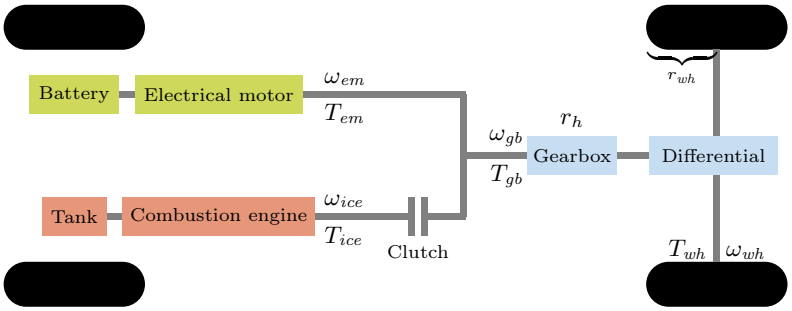

Figure 1: The powertrain configuration of a parallel hybrid electric vehicle.

based strategy is described in Section 3. Experimental results are presented in Section 4 and in Appendix A. We conclude the paper in Section 5. Lists of the acronyms, variables, and constants used in this paper are given in Appendix B.

2 Preliminaries

2.1 Vehicle Model

We consider a simplified model of a first generation *Toyota Prius*, but with a *parallel hybrid vehicle powertrain* (Figure 1). This parallel hybrid electric vehicle has an internal combustion engine (ICE) and an electrical motor (EM). Both engines are directly coupled to the same axis which is mounted to a manual transmission gearbox. The gearbox is connected to the wheels using a differential. So, the engines and the gearbox move with the same angular velocity $\omega_{ice} = \omega_{em} = \omega_{gb}$. We denote the current gear by h and model gear shifts as discrete changes. The gear ratio that corresponds to gear h is denoted by r_h and can be used to convert the angular velocity at the gearbox into the angular velocity at the wheels $\omega_{wh} = \omega_{gb}/r_h$. The correlation between the speed of the vehicle and the angular velocity at the wheels is given by $v = r_{wh}\omega_{wh}$, where r_{wh} is the wheel radius. If an acceleration/deceleration a is requested by the driver, the necessary torque T_{wh} at the wheels can be computed by

$$T_{wh} \;=\; r_{wh}\left(\frac{1}{2}\rho C_d A v^2 + (m+m_r)a + mgf_r\cos(\theta) + mg\sin(\theta)\right) \;,$$

where ρ is the density of air, C_d the air drag resistance, A the vehicle frontal area, m the mass of the vehicle, m_r the equivalent mass of the rotating parts of the vehicle, g the acceleration of gravity, f_r the rolling resistance, and θ the road slope. The torque T_{gb} that has to be generated by the engines can be obtained by

$$T_{gb} \;=\; \frac{T_{wh} + T_{br}}{\eta_{gb} r_h} \;=\; T_{ice} + T_{em} \;,$$

where T_{ice} is the torque at the ICE, T_{em} is the torque at the EM, T_{br} is the torque applied to the brakes, and η_{gb} is the mechanical transmission efficiency which we assume to be constant. We ignore the dynamics of the internal combustion engine and the electrical motor and assume immediate torque responses.

The instantaneous fuel consumption $\dot{m}_f$ is given by a function depending on the current angular velocity and the torque produced by the internal combustion engine. We approximate this function by interpolation on a discrete map.

The *battery state of charge* (SoC) can be computed from the input power P_{em} of the electrical motor, the maximum cell capacity $Q_{batt,max}$, the battery current I and the open circuit voltage U_{oc}. The change of the SoC is given by

$$\dot{SoC} = -\frac{I}{Q_{batt,max}} = -\frac{P_{em}}{U_{oc}Q_{batt,max}} \ .$$

The input power of the electrical motor depends on the current angular velocity and torque of the motor. It is obtained by interpolation on a discrete map. We assume that the battery is given by an equivalent circuit where the battery cells are connected in series and we neglect internal losses. To guarantee a long battery lifetime, a control strategy has to keep the SoC near a reference value SoC_{ref}.

For a smooth and safe operation of the hybrid powertrain, the following limits are given for the angular velocities and torques of the engines as well as for the battery state of charge:

$$\begin{array}{rcccl} \omega_{ice,min} & \leq & \omega_{ice} & \leq & \omega_{ice,max} \\ \omega_{em,min} & \leq & \omega_{em} & \leq & \omega_{em,max} \\ 0 & \leq & T_{ice} & \leq & T_{ice,max}(\omega_{ice}) \\ T_{em,min}(\omega_{em}) & \leq & T_{em} & \leq & T_{em,max}(\omega_{em}) \\ SoC_{min} & \leq & SoC & \leq & SoC_{max} \end{array} \tag{1}$$

The bounds $T_{ice,max}$, $T_{em,min}$ and $T_{em,max}$ are functions of the respective angular velocities and are represented by static maps. Note that T_{ice} and T_{br} are always non-negative. However, T_{em} can have a negative value, in which case the battery is charged. Another way to charge the battery is recuperation of braking energy by opening the clutch at the ICE and using the EM as a generator.

2.2 Energy Management and Drivability

In PHEVs, both the torque of the internal combustion engine T_{ice} and the torque of the electrical motor T_{em} sum up to the torque at the gearbox T_{gb}. At each time step $t \in \{0, \ldots, T\}$, a control strategy gets an input speed $v(t)$, a road slope $\theta(t)$, and a gear $h(t)$. A speed function $v(t)$ together with a road slope function $\theta(t)$ and a gear function $h(t)$ is called a *driving cycle*. The control strategy computes for the given input a split $u(t) \in \mathbb{Q}_{\geq 0}$ that specifies the amount of torque that is produced by the internal combustion engine. The remaining torque is generated by the electrical motor.

$$\begin{array}{rcl} T_{ice}(t) & = & u(t) \cdot T_{gb}(t) \\ T_{em}(t) & = & (1 - u(t)) \cdot T_{gb}(t) \end{array}$$

In the following, we use $T^u_{ice}(t)$ and $T^u_{em}(t)$ when we explicitly refer to the split $u(t)$ under which the torques are determined. For a given driving cycle, the *quality* of a split function $u(t)$ is evaluated using an evaluation function $J(u(t), SoC(t), t)$. Common aspects that are considered are the fuel consumption, the battery state of charge, and the drivability of the split function. The aim in the *discretized* energy management problem is to find a cost-minimal split function $u^*(t)$ with value

$$J^* \ = \ \min_{u(\cdot)} \ \sum_{t=0}^{T} J(u(t), SoC(t), t) \ .$$

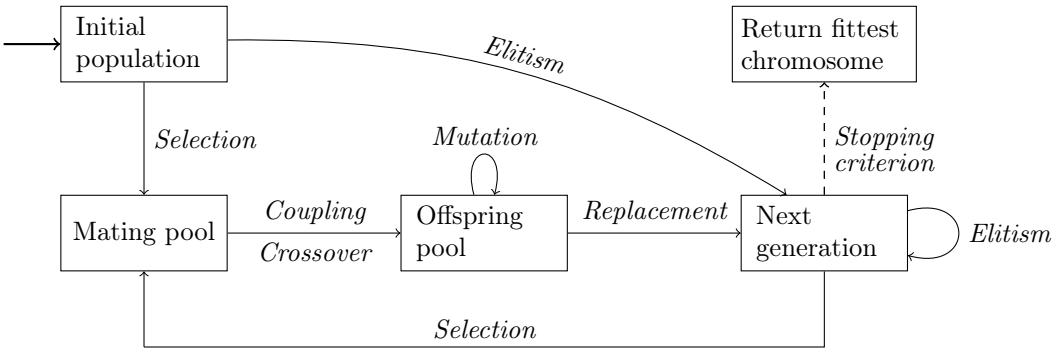

Figure 2: Schematic work flow of a genetic algorithm.

In general, the split of the optimal solution $u^*(t)$ for the energy management problem may fluctuate arbitrarily. Due to the noise emission of the ICE for varying torques, this results in bad *drivability* and reduces the driving comfort. To prevent this, on the one hand, an additional constraint can be added to the optimization that keeps the difference of consecutive splits in a predefined range. On the other hand, an evaluation function can be implemented that prefers consecutive splits with a small deviation.

2.3 Genetic Algorithms

Genetic algorithms [5, 7] belong to the class of stochastic local search algorithms [12]. Instead of performing a systematical search, genetic algorithms evolve a set of individual search paths. In general, this class of algorithms does not guarantee optimal solutions. However, for problems with a huge search space, they are fast in computing suitable solutions. In the area of the energy management problem for PHEVs, where a split has to be computed within a limited time frame, local search algorithms are a reasonable choice since they can terminate the computation any time and provide the best solution they have computed so far.

Genetic algorithms are inspired from evolution by natural selection. Figure 2 gives a schematic overview of a genetic algorithm. In each iteration (*generation*) of the optimization, a genetic algorithm uses a set of solutions (*population*). Initially, the set of solutions (*initial population*) is arbitrary, often randomly generated. Each solution is called a *chromosome* and can represent a single value (*gene*) or a sequence of values (*genes*). For each chromosome in the population, a value (*fitness*) is computed using an evaluation function (*fitness function*). The solutions with the highest values (*fittest chromosomes*) are selected for generating new solutions by adding them to the *mating pool*. A chromosome c_i is selected with a probability $f_i \,/\, \sum_{j=1}^{K} f_j$ (*fitness-proportional selection*), where f_i is the fitness of chromosome c_i and K is the population size.

New solutions are generated by *crossover* and *mutation*. For crossover at least two chromosomes from the mating pool are selected (*coupled*) and their solutions are combined. These new chromosomes (*children*) are put to the *offspring pool*. The size of the offspring pool is $\rho \cdot K$, where ρ is called *crossover rate*. Note that children can be far away from the original solutions (*parents*) in the search space. However, in the course of evolution, the solutions converge and the effect of the crossover becomes smaller (*decreasing diversity* in the population). Further randomness is added to the search by *mutation*, where the solutions of the offspring pool are

randomly changed with a probability μ (*mutation rate*).

The next generation is built from the chromosomes of the current generation, the offspring pool, and a new set of randomly generated solutions. Usually, the fittest chromosomes, the *elite*, survive in the next generation. Note that this *elitism* guarantees that the best solutions so far are used in the next generation and thus the fitness of the best chromosome cannot decrease. The other chromosomes of the current generation are *replaced* by chromosomes of the offspring pool either randomly or depending on the fitness. If the size of the offspring pool is not sufficient to replace the chromosomes, additional randomly generated chromosomes are added to the next generation.

When the *stopping criterion* has been reached, the optimization terminates and the genetic algorithm outputs the best solution, i.e., the fittest chromosome of the last generation. Different operators and methods for selection, crossover, mutation, elitism and replacement have been proposed in the literature [5].

3 Genetic Algorithm based Control Strategy

3.1 Genetic Algorithm Library

We implemented an extensible genetic algorithms library called GeneiAL, which is published under the MIT License [3]. Being the building block of our control strategy, the library was designed to encapsulate the work flow of a genetic algorithm. GeneiAL provides a framework to maximize the fitness value of chromosomes in a population over the course of generations using a custom fitness function for a user-defined optimization problem. The library offers the user high flexibility and extensibility while simultaneously featuring good scalability for the core functionality, e.g., through multi-threading support for evaluating the fitness function. For this purpose, GeneiAL supports user-defined chromosome types. Specifically, it is possible to optimize sequences of genes, i.e., the result is a sequence of optimized values.

The library facilitates the user to specify how initial chromosomes are generated: By default the initial population is filled with chromosomes that have random values within a specified interval. However, for chromosomes with a gene sequence, it is also possible to specify the maximal difference between two consecutive genes. Furthermore, instead of generating a random initial population, GeneiAL also supports the partial or full reuse of a previous population from a prior optimization run.

Additionally, GeneiAL allows the implementation of customized genetic operators. Apart from commonly known genetic operators (e.g., roulette-wheel selection, N-point crossover, and uniform mutation [5]), we implemented problem-specific genetic operations for crossover and mutation of chromosomes with a gene sequence (smoothed crossover, smoothed mutation).

The optimization evolves until a stopping criterion is reached. GeneiAL provides the following basic stopping criteria: The *fitness value criterion* is triggered when the best fitness value of the current generation is greater or equal a specified value. Alternatively, the computation can be terminated when a maximal number G_{max} of generations has been produced. The *fixed-point criterion* compares the best fitness value of the current generation with the best fitness value in previous generations within a specified window size. The computation is stopped when the difference between the fitness values is smaller than a given threshold value. GeneiAL allows to create advanced user-defined stopping criteria as well as combining (AND, OR, XOR, negation) multiple stopping criteria.

A set of diagnostic tools enables analyzing the genetic algorithm's behavior, i.e. to measure its runtime and to analyze the behavior of single genetic operators. Additionally, GeneiAL can

be easily extended by user-specified observers with customized pre- and post-processing logic, e.g., before a new generation is evolved.

3.2 Control Strategy

Our control strategy is a predictive control strategy based on optimal control. In contrast to control strategies based on dynamic programming (DP) [1], our strategy stores a feasible solution in each optimization step. Thus, it can provide a solution even if the optimization is stopped early.

In each time step, the strategy gets predictions on the speed of the vehicle, the slope of the road, and the gear used by the driver for all time steps within the prediction horizon, i.e., for the current time step and the following T_p time steps. It optimizes a split sequence of length $T_p + 1$ and returns only the first value of this sequence as the split for the current time step.

The chromosomes represent split sequences with a split for each time step within the prediction horizon. We use split values $u(i)$ from the set $\{0, 0.01, \ldots, 1\}$ for $i \in \{t, \ldots, t + T_p\}$.

$u(t)$	$u(t+1)$	$u(t+2)$	$\cdots$	$u(t+T_p)$

The population is a set $\mathcal{C}$ of K chromosomes, where K is fixed for all generations. The initial population is generated uniformly at random, i.e., each chromosome is a sequence of randomly generated splits $c_k = u^k(t) \circ u^k(t+1) \circ \ldots \circ u^k(t+T_p)$ for $k \in \mathcal{C}$. Alternatively, we can generate the initial population for a time step $t + 1$ by reusing the optimized population of time step t. For this, the split sequence of each chromosome is shifted by one and a random split value is appended for time step $t + T_p + 1$, i.e. $c'_k = u^k(t+1) \circ u^k(t+2) \circ \ldots \circ u^k(t+T_p+1)$, where $u^k(t + T_p + 1)$ is randomly generated.

The fitness function is used to evaluate the chromosomes. In the area of HEVs, multiple objectives have to be considered: the fuel consumption, the battery state of charge (SoC) at the end of the prediction horizon, the difference between SoC and SoC_{ref}, and the difference between consecutive splits. We implemented the following fitness functions that compute values within the interval $[0, 1]$ for a prediction horizon of length T_p+1. Each of these fitness functions computes a high fitness value for good chromosomes and a low value for bad ones.

- *Fuel consumption minimization e_f:* We estimate a lower bound ($\dot{m}_{f,min}$) and an upper bound ($\dot{m}_{f,max}$) for the instantaneous fuel consumption within the prediction horizon $[t, t + T_p]$ and set

$$e_f := 1 \; - \; \left(\frac{1}{T_p + 1} \; \cdot \; \sum_{\tau=t}^{t+T_p} \dot{m}_f(\omega_{ice}(\tau), T^u_{ice}(\tau)) \; - \; \dot{m}_{f,min} \right) \; / \; (\dot{m}_{f,max} \; - \; \dot{m}_{f,min}) \; .$$

  Note that this equation scales the average fuel consumption to the interval $[0, 1]$ and that GeneiAL maximizes the fitness function, whereas the fuel consumption has to be minimized.

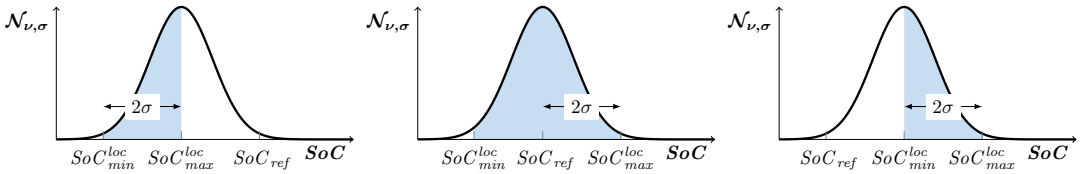

Figure 3: Shifted and scaled normal distribution depending on SoC^{loc}_{min}, SoC^{loc}_{max}, and SoC_{ref}. Left: $SoC^{loc}_{max} < SoC_{ref}$, center: $SoC^{loc}_{min} \leq SoC_{ref} \leq SoC^{loc}_{max}$, right: $SoC^{loc}_{min} > SoC_{ref}$. The area of the interval $[SoC^{loc}_{min}, SoC^{loc}_{max}]$ is filled blue.

- *SoC level maximization e_{sl}:* We estimate a lower bound (SoC^{loc}_{min}) and an upper bound (SoC^{loc}_{max}) for the battery state of charge at time step $t + T_p$ and set

$$e_{sl} := \Big(SoC(t+T_p) \;-\; SoC^{loc}_{min}\Big) \;/\; \Big(SoC^{loc}_{max} \;-\; SoC^{loc}_{min}\Big) \;.$$

- *SoC deviation minimization e_{sd}:* The idea behind this evaluation function is to keep the battery state of charge near SoC_{ref} using a normal distribution function (Figure 3). Let $\mathcal{N}_{\nu,\sigma}$ be the normal distribution function with expected value ν and standard deviation σ. We set

$$e_{sd} := \mathcal{N}_{\nu,\sigma}(SoC(t+T_p)) \;/\; \mathcal{N}_{\nu,\sigma}(\nu) \;,$$

  where ν and σ depend on the estimates of SoC^{loc}_{min}, SoC^{loc}_{max}, and on the reference value SoC_{ref}. With $SoC_{diff} := \max\left(|SoC^{loc}_{min} - SoC_{ref}|, |SoC^{loc}_{max} - SoC_{ref}|\right)$, we use

$$(\nu,\sigma) := \begin{cases} (SoC^{loc}_{max},\; (SoC^{loc}_{max} - SoC^{loc}_{min}) \;/\; 2) & \text{for } SoC^{loc}_{max} < SoC_{ref} \\ (SoC_{ref},\; SoC_{diff} \;/\; 2) & \text{for } SoC^{loc}_{min} \leq SoC_{ref} \leq SoC^{loc}_{max} \\ (SoC^{loc}_{min},\; (SoC^{loc}_{max} - SoC^{loc}_{min}) \;/\; 2) & \text{for } SoC^{loc}_{min} > SoC_{ref} \end{cases} .$$

  Thereby, we shift the normal distribution function such that the battery state of charge $SoC \in [SoC^{loc}_{min}, SoC^{loc}_{max}]$ that minimizes $|SoC - SoC_{ref}|$ is mapped to the highest value. Furthermore, the normal distribution function is scaled such that the battery state of charge $SoC \in [SoC^{loc}_{min}, SoC^{loc}_{max}]$ that maximizes $|SoC - SoC_{ref}|$ is mapped to a value near 0.

- *Split difference minimization e_{ud}:* Let $[u_{min}, u_{max}]$ be the allowed split range. With $u(-1) = 0$, we set

$$e_{ud} := 1 \;-\; \left(\frac{1}{T_p+1} \;\cdot\; \sum_{\tau=t}^{t+T_p} (u(\tau) - u(\tau-1))^2\right) \;/\; (u_{max} - u_{min})^2 \;.$$

- *Split difference transgression e_{ut}:* Let Δu_{max} be the maximal allowed difference between consecutive splits and $[u_{min}, u_{max}]$ be the allowed split range. We set

$$e_{ut} := 1 \;-\; \left(\frac{1}{T_p+1} \;\cdot\; \sum_{\tau=t}^{t+T_p} i_\tau \cdot (\Delta u_{max} - |u(\tau) - u(\tau-1)|)^2\right) \;/\; (\Delta u_{max} - (u_{max} - u_{min}))^2 \;,$$

  where $i_\tau := \begin{cases} 0 & \text{if } |u(\tau) - u(\tau-1)| < \Delta u_{max} \\ 1 & \text{else} \end{cases}$.

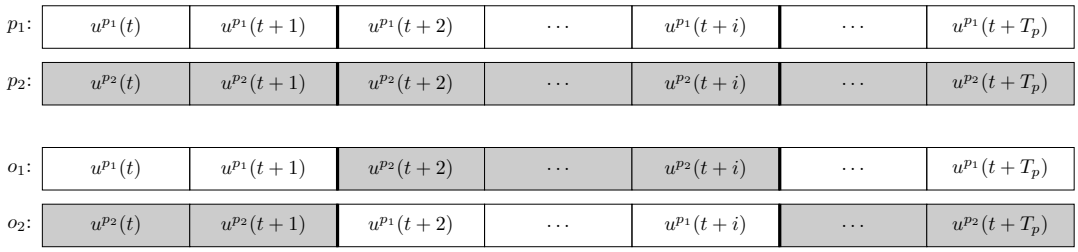

Figure 4: 2-point crossover for the points $(t+1, t+i)$

The fitness function is a weighted sum of these evaluation functions, i.e., $w_f e_f + w_{sl} e_{sl} + w_{sd} e_{sd} + w_{ud} e_{ud} + w_{ut} e_{ut}$. The *fitness configuration* of a GA-based control strategy is the tuple $(w_f, w_{sl}, w_{sd}, w_{ud}, w_{ut})$. Note that the weights are positive numbers that sum up to one. So, the fitness value is in the interval $[0, 1]$.

The genetic operators (selection, coupling, crossover, mutation, elitism, replacement) are used to build a new generation. For selection, we chose *roulette wheel selection* [5] that selects chromosomes based on the idea of spinning a roulette wheel. The space that a chromosome occupies on the wheel corresponds to its probability, i.e., we use a fitness-proportional selection.

The coupling of the chromosomes in the mating pool is done randomly. This means that two chromosomes are chosen uniformly at random for mating.

For crossover, we use different strategies that generate two children for each chromosome couple. *N-point crossover* swaps between the parents' genes at N randomly selected points (Figure 4). *Smoothed crossover* is a variant of N-point crossover, where the split differences between neighboring genes of the offspring are kept within $\pm\Delta u_{max}$ for better drivability.

The offspring is mutated using *uniform mutation*, i.e., a gene is selected uniformly at random for mutation. A set of k randomly selected genes of a chosen chromosome are mutated. This means that the split is replaced by a random value. *Smoothed mutation* is a variant of uniform mutation. After a uniform mutation, the ε-neighborhood of each of the k mutated genes is increased/decreased in the same direction than the mutated gene for better drivability. Thereby, we assure that the split differences of consecutive genes are kept within $\pm\Delta u_{max}$.

For the new generation, an elite of the chromosomes with the highest fitness is selected and these chromosomes are added to the next generation. Furthermore, we replace the worst chromosomes by new offspring from the offspring pool. If the number of offspring is smaller than the number of chromosomes that should be replaced, we add randomly generated chromosomes to the next generation.

As stopping criterion, the algorithm terminates after G_{max} generations in order to ensure a fixed computation time.

The GA-configuration of a GA-based control strategy is a tuple $C_{GA} = (K, \rho, \mu, G_{max})$ that contains the population size, the crossover rate, the mutation rate, and the maximal number of generations. In our implementation, the number of elite chromosomes is given by $(1-\rho)K$.

4 Experimental Results

We tested our GA-based control strategy using 2-point-crossover and uniform mutation and compared it with other control strategies using a `MATLAB/Simulink` model with the vehicle dynamics from Section 2.1. For this purpose, we chose the following commonly used driving cycles as benchmarks: the NEDC (*New European Driving Cycle*) is the standard driving cycle used in Europe and is composed of a city and a highway part. The city program FTP-75 and the highway program HWFET of the *EPA Federal Test Procedure* are the standard driving cycles used in the US.

For real-time capability, the current torque T_{gb} is distributed between the engines every 0.02 seconds based on a given torque split. However, we allow that the torque split computed by a control strategy can stay fixed for at most one second. The GA-based strategy is real-time capable, e.g., for $C_{GA} = (50, 0.75, 0.1, 100)$. In our benchmarks, we use a prediction horizon of length $T_p = 20$ seconds for all predictive control strategies. For this length, a navigation system can provide a sufficiently good approximation on the future driving conditions.

Table 1 gives the fuel consumption in gram and the battery state of charge (SoC) at the end of the driving cycle for a set of common control strategies. All strategies that are presented in this section use $SoC_{ref} = 0.6$ and keep the state of charge of the battery in the interval $[0.5, 0.7]$.

	NEDC		FTP-75		HWFET	
Strategy	Fuel cons.	SoC	Fuel cons.	SoC	Fuel cons.	SoC
ICE	**428.101**	0.7000	**727.606**	0.7000	**379.841**	0.7000
EM	392.686	0.6246	593.519	0.5336	360.903	0.5805
RDP	387.081	0.6246	591.783	0.5614	350.621	0.5807
ADP	388.726	0.6245	589.266	0.5336	352.513	0.5805
A-ECMS	390.239	0.6246	592.168	0.5442	356.356	0.5806
T-ECMS	391.641	0.6246	592.514	0.5335	358.858	0.5805
GA_{best}	**380.123**	0.6263	**581.001**	0.5398	**347.776**	0.5789

Table 1: Fuel consumption in gram and SoC at the end of the driving cycle for various control strategies. GA_{best} uses the GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$ and the fitness configuration $(0.5, 0.0, 0.5, 0.0, 0.0)$ (NEDC, HWFET) or $(0.5, 0.5, 0.0, 0.0, 0.0)$ (FTP-75). The lowest and highest fuel consumption values are typed in black and blue bold print, respectively.

The strategy ICE drives purely with the internal combustion engine, unless $\omega_{ice} < \omega_{ice,min}$. Analogously, the strategy EM uses the electrical motor whenever this is feasible according to (1). In particular, if $SoC < SoC_{min}$, EM uses the internal combustion engine. Table 1 shows that the highest fuel consumption is obtained by the strategy ICE followed by the EM strategy.

However, the control strategies based on optimal control (RDP, ADP, A-ECMS, T-ECMS, GA) that optimize when to use the electrical motor achieve better results. The control strategies RDP and ADP [4] are based on *receding dynamic programming* [1], where the fuel consumption is minimized using an objective function that calculates the weighted sum of the fuel mass flow and the deviation of the battery state of charge from SoC_{ref} for a prediction horizon. Additionally, ADP uses a cost-to-go approximation in the objective function to estimate the costs that are needed for the rest of the driving cycle. The *equivalent consumption minimization strategy* (ECMS) [6] minimizes the equivalent fuel consumption, i.e., the sum of the current fuel consumption and the input power of the electrical motor times a time-dependent equivalence factor. In *adaptive ECMS* (A-ECMS) [6], the equivalence factor is adapted using, e.g., a PI-

controller. The *telemetry ECMS* (T-ECMS) [13] adapts the equivalence factor over time using an estimation for the energy needed for the rest of the driving cycle (*energy horizon*). T-ECMS uses two equivalence factors, one for charging and one for discharging. These factors are weighted using the estimated energy horizon. Note that both ECMS does not need a priori knowledge of the driving cycle.

Finally, the results of the best GA-configuration for the respective driving cycles are presented in Table 1. The fuel consumption of our GA-based strategy is up to 1.8% lower than the results for the other control strategies on all driving cycles.

Fitness evaluation and drivability. In this section, we present the results of our GA-based control strategy for different fitness weights. First, we searched for a configuration of fitness weights with optimal results for various driving cycles. For this benchmark (Table 2), we chose the real-time capable GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$ and used *fitness configurations* $(w_f, w_{sl}, w_{sd}, w_{ud}, w_{ut})$ that neglect the drivability ($w_{ud} = w_{ut} = 0$).

Fitness weights					NEDC		FTP-75		HWFET	
w_f	w_{sl}	w_{sd}	w_{ud}	w_{ut}	Fuel cons.	SoC	Fuel cons.	SoC	Fuel cons.	SoC
0.0	0.0	1.0	0.0	0.0	397.632	0.7000	598.475	0.6325	362.076	0.6768
0.0	0.2	0.8	0.0	0.0	397.722	0.7000	598.507	0.6327	362.080	0.6770
0.0	0.4	0.6	0.0	0.0	397.848	0.7000	599.146	0.6348	362.171	0.6777
0.0	0.6	0.4	0.0	0.0	423.424	0.7000	711.675	0.7000	373.724	0.7000
0.0	0.8	0.2	0.0	0.0	**424.079**	0.7000	713.216	0.7000	374.075	0.7000
0.0	1.0	0.0	0.0	0.0	424.075	0.7000	**713.247**	0.7000	374.041	0.7000
0.2	0.0	0.8	0.0	0.0	397.058	0.7000	597.775	0.6316	361.880	0.6760
0.2	0.2	0.6	0.0	0.0	397.550	0.7000	597.730	0.6323	361.929	0.6770
0.2	0.4	0.4	0.0	0.0	397.585	0.7000	597.956	0.6330	361.980	0.6775
0.2	0.6	0.2	0.0	0.0	423.832	0.7000	708.784	0.7000	374.160	0.7000
0.2	0.8	0.0	0.0	0.0	424.009	0.7000	711.952	0.7000	374.158	0.7000
0.4	0.0	0.6	0.0	0.0	391.080	0.6820	597.439	0.6307	354.816	0.6410
0.4	0.2	0.4	0.0	0.0	393.618	0.6943	597.439	0.6313	358.780	0.6608
0.4	0.4	0.2	0.0	0.0	397.530	0.7000	597.226	0.6321	361.718	0.6772
0.4	0.6	0.0	0.0	0.0	423.632	0.7000	705.355	0.7000	**374.402**	0.7000
0.6	0.0	0.4	0.0	0.0	386.894	0.6260	584.393	0.5331	353.313	0.5789
0.6	0.2	0.2	0.0	0.0	386.462	0.6260	583.841	0.5330	351.940	0.5789
0.6	0.4	0.0	0.0	0.0	384.365	0.6260	582.048	0.5330	350.275	0.5789
0.8	0.0	0.2	0.0	0.0	386.914	0.6260	584.897	0.5330	354.080	0.5789
0.8	0.2	0.0	0.0	0.0	384.917	0.6260	582.834	0.5330	352.463	0.5789
1.0	0.0	0.0	0.0	0.0	384.806	0.6260	583.413	0.5330	353.265	0.5789
0.5	0.0	0.5	0.0	0.0	**380.123**	0.6263	595.223	0.6205	**347.776**	0.5789
0.5	0.5	0.0	0.0	0.0	382.325	0.6260	**581.001**	0.5398	349.560	0.5789

Table 2: Fuel consumption in gram and SoC at the end of the driving cycle for different fitness weights and the real-time capable GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$. The lowest and highest fuel consumption values are typed in black and blue bold print, respectively.

In Table 5 in Appendix A, the results for a broader range of fitness weights are presented. The quality of the optimization (fuel consumption) depends on the fitness configuration. For bad configurations, the GA-based strategy performs only slightly better than the ICE strategy; for good configurations, the fuel consumption is within the range of the best control strategies.

Good results are achieved for $w_f \in [0.5, 1.0]$. The fitness configuration $(0.5, 0.5, 0.0, 0.0, 0.0)$ yields good results on all driving cycles.

In a second approach (Table 3), we considered the drivability evaluation in the fitness function (i.e., $w_{ud} > 0$ or $w_{ut} > 0$). First, we neglected the battery state of charge in the optimization, i.e., we set $w_{sl} = w_{sd} = 0$ (upper part of Table 3). However, we considered also the optimization of the fuel consumption, the battery state of charge, and the drivability simultaneously (lower part of Table 3).

We use $\sum|.| := \sum_{t=0}^{T} |u(t) - u(t-1)|$ with $u(-1) := 0$ to measure the drivability of the computed split sequences. The best drivability (low $\sum|.|$) is achieved for high w_{ud} values. However, we observe that additional drivability constraints impose a fuel consumption that is only up to 1.7% higher (fitness configuration $(0.0, 0.0, 0.0, 1.0, 0.0)$ on HWFET) than for the fitness configuration $(1.0, 0.0, 0.0, 0.0, 0.0)$ that neglects drivability. Table 6 in Appendix A gives the results for further fitness configurations.

Fitness weights					NEDC			FTP-75			HWFET		
w_f	w_{sl}	w_{sd}	w_{ud}	w_{ut}	Fuel cons.	SoC	$\sum\|.\|$	Fuel cons.	SoC	$\sum\|.\|$	Fuel cons.	SoC	$\sum\|.\|$
0.0	0.0	0.0	0.0	1.0	386.822	0.6259	111.23	589.103	0.5585	149.91	356.827	0.5789	92.64
0.0	0.0	0.0	0.2	0.8	389.248	0.6259	53.93	587.981	0.5366	73.09	359.213	0.5789	45.51
0.0	0.0	0.0	0.4	0.6	389.291	0.6259	52.31	588.118	0.5369	72.50	359.264	0.5789	44.06
0.0	0.0	0.0	0.6	0.4	389.298	0.6259	51.55	588.075	0.5365	71.04	359.266	0.5789	43.54
0.0	0.0	0.0	0.8	0.2	389.317	0.6259	51.31	587.937	0.5358	70.17	359.288	0.5789	43.16
0.0	0.0	0.0	1.0	0.0	389.325	0.6259	51.13	587.916	0.5357	**69.70**	359.271	0.5790	**42.98**
0.2	0.0	0.0	0.0	0.8	387.969	0.6260	71.42	586.399	0.5330	121.31	357.231	0.5789	91.44
0.2	0.0	0.0	0.2	0.6	389.050	0.6259	54.38	587.686	0.5330	83.00	358.585	0.5789	71.68
0.2	0.0	0.0	0.4	0.4	389.223	0.6260	50.53	587.848	0.5330	77.53	358.785	0.5789	63.59
0.2	0.0	0.0	0.6	0.2	389.300	0.6259	50.66	587.853	0.5330	76.23	358.950	0.5789	59.55
0.2	0.0	0.0	0.8	0.0	389.361	0.6260	**48.85**	587.958	0.5330	74.11	359.032	0.5789	57.23
0.4	0.0	0.0	0.0	0.6	387.847	0.6260	69.67	586.394	0.5330	122.72	357.121	0.5789	96.17
0.4	0.0	0.0	0.2	0.4	388.772	0.6260	57.94	587.560	0.5330	90.06	358.265	0.5789	79.69
0.4	0.0	0.0	0.4	0.2	388.998	0.6259	55.09	587.719	0.5330	84.64	358.511	0.5789	72.74
0.4	0.0	0.0	0.6	0.0	389.134	0.6260	52.98	587.746	0.5330	83.94	358.600	0.5789	70.47
0.6	0.0	0.0	0.0	0.4	387.719	0.6260	72.51	586.156	0.5330	126.73	356.879	0.5789	102.95
0.6	0.0	0.0	0.2	0.2	388.545	0.6260	59.99	587.340	0.5330	97.64	357.843	0.5789	91.72
0.6	0.0	0.0	0.4	0.0	388.717	0.6260	58.88	587.438	0.5330	96.18	358.001	0.5789	88.45
0.8	0.0	0.0	0.0	0.2	387.146	0.6259	74.71	585.949	0.5330	136.32	356.095	0.5789	116.63
0.8	0.0	0.0	0.2	0.0	387.960	0.6260	65.64	586.937	0.5330	109.79	357.015	0.5789	107.37
1.0	0.0	0.0	0.0	0.0	384.845	0.6260	**116.64**	583.357	0.5330	**213.93**	353.186	0.5789	**168.29**
0.05	0.05	0.00	0.90	0.00	389.268	0.6259	53.50	587.579	0.5332	73.97	359.145	0.5789	52.66
0.05	0.00	0.05	0.90	0.00	389.228	0.6259	54.26	587.588	0.5338	76.29	359.025	0.5789	52.11
0.05	0.00	0.05	0.80	0.10	389.207	0.6259	52.37	587.481	0.5333	77.73	358.992	0.5789	53.30

Table 3: Fuel consumption in gram, SoC at the end of the driving cycle, and the sum of the split differences of consecutive time steps ($\sum|.|$) for different fitness weights and the GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$. The lowest and highest values for $\sum|.|$ are typed in black and blue bold print, respectively.

In Table 4, we restricted the search space to drivable split sequences, i.e., two consecutive splits differ by at most Δu_{max}. Small Δu_{max} values improve drivability whereas the fuel consumption is only slightly worse than the results in Table 3.

Real-time capability. We examined the convergence behavior of our strategy for $K \in \{30, 50, 100, 200, 500, 1000\}$. We used the fitness configuration $(1.0, 0.0, 0.0, 0.0, 0.0)$, $\rho = 0.75$, $\mu = 0.1$, an elite of size $(1 - \rho)K$, and we terminated the optimization after 100 generations.

Figure 5 shows the increase of the fitness values over 100 generations of a genetic algorithm run for different population sizes. The population size influences the maximal fitness value of the initial population and the convergence rate, which influences the quality of the optimization

	NEDC			FTP-75			HWFET		
Δu_{max}	Fuel cons.	*SoC*	$\sum \lvert . \rvert$	Fuel cons.	*SoC*	$\sum \lvert . \rvert$	Fuel cons.	*SoC*	$\sum \lvert . \rvert$
0.1	388.471	0.6259	**35.19**	588.044	0.5330	**64.68**	358.312	0.5792	**74.11**
0.2	386.700	0.6259	46.06	586.411	0.5330	90.35	352.408	0.5791	99.52
0.3	386.155	0.6259	50.45	585.729	0.5330	109.33	351.386	0.5791	107.31
0.4	386.030	0.6259	55.59	585.263	0.5330	125.73	351.748	0.5791	116.15
0.5	385.980	0.6260	63.01	584.897	0.5330	138.25	352.386	0.5791	127.76
0.6	385.753	0.6259	67.13	584.682	0.5330	153.52	352.851	0.5791	138.66
0.7	385.520	0.6259	76.05	584.370	0.5330	168.72	353.286	0.5791	148.26
0.8	385.385	0.6260	84.87	584.272	0.5330	184.91	353.571	0.5791	161.16
0.9	385.289	0.6260	93.57	584.088	0.5330	199.62	353.777	0.5791	169.61
1.0	385.400	0.6260	**109.26**	584.038	0.5330	**210.10**	353.985	0.5791	**177.49**

Table 4: Fuel consumption in gram, *SoC* and $(\sum \lvert . \rvert)$ at the end of the driving cycle. The fitness configuration $(1, 0, 0, 0, 0)$, the GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$, and the *smoothing variants* of *N-point crossover* and *uniform mutation* have been used. The lowest and highest values for $\sum \lvert . \rvert$ are typed in black and blue bold print, respectively.

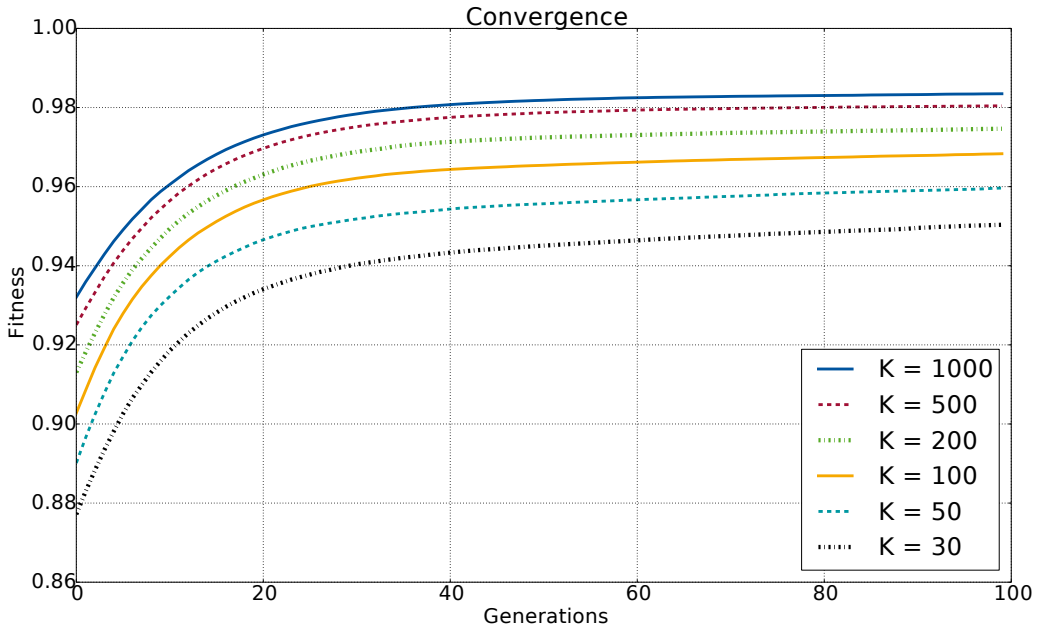

Figure 5: Fitness values of the best chromosome in the population over the generations of a genetic algorithm run with $\rho = 0.75$ and $\mu = 0.1$ for different population sizes.

result. However, as a compromise between the quality of the optimization result and the real-time capability of our control strategy, we use a population size of $K = 50$.

5 Conclusion

Our GA-based control strategy shows promising results for the energy management problem of hybrid electric vehicles. The fuel consumption of our strategy is comparable to the results of common control strategies. On the NEDC, the GA-based strategy performs about 1.8%

better than the best considered reference strategy. Furthermore, adding an evaluation of the drivability to the fitness function or restricting the search space to drivable split sequences has little effect on the fuel consumption at the end of a driving cycle. However, the driving comfort can be improved by both. Nevertheless, the choice of the fitness configuration is important to achieve a low fuel consumption. By adjusting the population size, we get a real-time capable control strategy. As future work, we plan to extend the GA-based control strategy such that the first x torque splits of a chromosome are used instead of only the first one. By this, we can spend more time on the optimization.

References

[1] Hoseinali Borhan, Chen Zhang, Ardalan Vahidi, Anthony M. Phillips, Ming L. Kuang, and Stefano Di Cairano. Nonlinear Model Predictive Control for Power-split Hybrid Electric Vehicles. In *Proceedings of the 49th IEEE Conference on Decision and Control (CDC)*, pages 4890–4895. IEEE, 2010.

[2] Li-Cun Fang and Shi-Yin Qin. Concurrent Optimization for Parameters of Powertrain and Control System of Hybrid Electric Vehicle Based on Multi-Objective Genetic Algorithms. In *Proceedings of the SICE-ICASE International Joint Conference*, pages 2424–2429. IEEE, 2006.

[3] Genetic algorithm library GeneiAL. http://geneial.org.

[4] Sascha Geulen, Martina Josevski, Johanna Nellen, Janosch Fuchs, Lukas Netz, Benedikt Wolters, Dirk Abel, Erika Ábrahám, and Walter Unger. Learning-based Control Strategies for Hybrid Electric Vehicles. In *Proceedings of the 2015 IEEE Multi-Conference on Systems and Control (MSC)*, 2015. To appear.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, first edition, 1989.

[6] Lino Guzzella and Antonio Sciarretta. *Vehicle Propulsion Systems: Introduction to Modeling and Optimization.* Springer, third edition, 2013.

[7] Randy L. Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms.* Wiley-Interscience, second edition, 2004.

[8] Gino Paganelli, Sebastien Delprat, Thierry-Marie Guerra, Janette Rimaux, and Jean-Jacques Santin. Equivalent Consumption Minimization Strategy for Parallel Hybrid Powertrains. In *Proceedings of the 55th IEEE Vehicular Technology Conference*, pages 2076–2081. IEEE, 2002.

[9] Aishwarya Panday and Hari Om Bansal. A Review of Optimal Energy Management Strategies for Hybrid Electric Vehicle. *International Journal of Vehicular Technology*, 2014(160510), 2014.

[10] Antonio Piccolo, Lucio Ippolito, Vincenzo Galdi, and Alfredo Vaccaro. Optimisation of Energy Flow Management in Hybrid Electric Vehicles via Genetic Algorithms. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 434–439. IEEE, 2001.

[11] Pierluigi Pisu and Giorgio Rizzoni. A Comparative Study Of Supervisory Control Strategies for Hybrid Electric Vehicles. *IEEE Transactions on Control Systems Technology*, 15(3):506–518, 2007.

[12] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, third edition, 2010.

[13] Antonio Sciarretta, Michael Back, and Lino Guzzella. Optimal Control of Parallel Hybrid Electric Vehicles. *IEEE Transactions on Control System Technology*, 12(3):352–363, 2004.

[14] Marco Sorrentino, Ivan Arsie, Raffaele Di Martino, and Gianfranco Rizzo. On the Use of Genetic Algorithm to Optimize the On-board Energy Management of a Hybrid Solar Vehicle. *Oil&Gas Science and Technology - Rev. IFP*, 65(1):133–143, 2010.

[15] Randombage Sudath Wimalendra, Lanka Udawatta, Eran Edirisinghe, and Suranga Karunarathna. Determination of Maximum Possible Fuel Economy of HEV for Known Drive

Cycle: Genetic Algorithm Based Approach. In *Proceedings of the 4th International Conference on Information and Automation for Sustainability (ICIAFS)*, pages 289–294. IEEE, 2008.

A Appendix

Fitness weights					NEDC		FTP-75		HWFET	
w_f	w_{sl}	w_{sd}	w_{ud}	w_{ut}	Fuel cons.	SoC	Fuel cons.	SoC	Fuel cons.	SoC
0.0	0.0	1.0	0.0	0.0	397.632	0.7000	598.475	0.6325	362.076	0.6768
0.0	0.1	0.9	0.0	0.0	397.602	0.7000	598.501	0.6326	362.082	0.6770
0.0	0.2	0.8	0.0	0.0	397.722	0.7000	598.507	0.6327	362.080	0.6770
0.0	0.3	0.7	0.0	0.0	397.778	0.7000	598.563	0.6328	362.180	0.6775
0.0	0.4	0.6	0.0	0.0	397.848	0.7000	599.146	0.6348	362.171	0.6777
0.0	0.5	0.5	0.0	0.0	397.913	0.7000	602.638	0.6503	362.287	0.6779
0.0	0.6	0.4	0.0	0.0	423.424	0.7000	711.675	0.7000	373.724	0.7000
0.0	0.7	0.3	0.0	0.0	423.960	0.7000	713.061	0.7000	374.179	0.7000
0.0	0.8	0.2	0.0	0.0	424.079	0.7000	713.216	0.7000	374.075	0.7000
0.0	0.9	0.1	0.0	0.0	**424.095**	0.7000	**713.536**	0.7000	374.127	0.7000
0.0	1.0	0.0	0.0	0.0	424.075	0.7000	713.247	0.7000	374.041	0.7000
0.1	0.0	0.9	0.0	0.0	397.503	0.7000	597.966	0.6321	361.922	0.6765
0.1	0.1	0.8	0.0	0.0	397.510	0.7000	597.935	0.6325	361.967	0.6769
0.1	0.2	0.7	0.0	0.0	397.617	0.7000	597.938	0.6326	362.019	0.6772
0.1	0.3	0.6	0.0	0.0	397.674	0.7000	597.966	0.6327	362.045	0.6774
0.1	0.4	0.5	0.0	0.0	397.693	0.7000	598.630	0.6350	362.076	0.6776
0.1	0.5	0.4	0.0	0.0	397.575	0.7000	601.398	0.6492	362.139	0.6780
0.1	0.6	0.3	0.0	0.0	423.336	0.7000	710.972	0.7000	374.164	0.7000
0.1	0.7	0.2	0.0	0.0	423.973	0.7000	712.094	0.7000	374.085	0.7000
0.1	0.8	0.1	0.0	0.0	424.021	0.7000	712.537	0.7000	374.066	0.7000
0.1	0.9	0.0	0.0	0.0	424.004	0.7000	712.784	0.7000	374.079	0.7000
0.2	0.0	0.8	0.0	0.0	397.058	0.7000	597.775	0.6316	361.880	0.6760
0.2	0.2	0.6	0.0	0.0	397.550	0.7000	597.730	0.6323	361.929	0.6770
0.2	0.3	0.5	0.0	0.0	397.524	0.7000	597.769	0.6326	361.987	0.6774
0.2	0.4	0.4	0.0	0.0	397.585	0.7000	597.956	0.6330	361.980	0.6775
0.2	0.5	0.3	0.0	0.0	397.431	0.7000	600.889	0.6480	362.034	0.6780
0.2	0.6	0.2	0.0	0.0	423.832	0.7000	708.784	0.7000	374.160	0.7000
0.2	0.7	0.1	0.0	0.0	424.005	0.7000	711.451	0.7000	374.138	0.7000
0.2	0.8	0.0	0.0	0.0	424.009	0.7000	711.952	0.7000	374.158	0.7000
0.3	0.0	0.7	0.0	0.0	395.194	0.6991	597.731	0.6314	360.014	0.6655
0.3	0.2	0.5	0.0	0.0	397.432	0.7000	597.556	0.6317	361.778	0.6765
0.3	0.3	0.4	0.0	0.0	397.567	0.7000	597.554	0.6322	361.847	0.6771
0.3	0.4	0.3	0.0	0.0	397.522	0.7000	597.525	0.6327	361.904	0.6776
0.3	0.5	0.2	0.0	0.0	397.268	0.7000	600.564	0.6477	361.952	0.6781
0.3	0.6	0.1	0.0	0.0	423.819	0.7000	707.124	0.7000	**374.420**	0.7000
0.3	0.7	0.0	0.0	0.0	423.920	0.7000	710.770	0.7000	374.246	0.7000
0.4	0.0	0.6	0.0	0.0	391.080	0.6820	597.439	0.6307	354.816	0.6410
0.4	0.1	0.5	0.0	0.0	392.559	0.6892	597.411	0.6309	356.519	0.6495
0.4	0.2	0.4	0.0	0.0	393.618	0.6943	597.439	0.6313	358.780	0.6608
0.4	0.3	0.3	0.0	0.0	396.531	0.7000	597.308	0.6314	361.552	0.6757
0.4	0.4	0.2	0.0	0.0	397.530	0.7000	597.226	0.6321	361.718	0.6772
0.4	0.5	0.1	0.0	0.0	397.015	0.7000	599.433	0.6439	361.883	0.6784
0.4	0.6	0.0	0.0	0.0	423.632	0.7000	705.355	0.7000	374.402	0.7000
0.5	0.0	0.5	0.0	0.0	**380.123**	0.6263	595.223	0.6205	**347.776**	0.5789
0.5	0.1	0.4	0.0	0.0	380.227	0.6260	594.459	0.6173	348.343	0.5789
0.5	0.2	0.3	0.0	0.0	380.691	0.6260	593.793	0.6149	348.588	0.5789
0.5	0.3	0.2	0.0	0.0	381.075	0.6260	591.353	0.6035	348.982	0.5789
0.5	0.4	0.1	0.0	0.0	381.596	0.6260	588.705	0.5836	349.515	0.5789
0.5	0.5	0.0	0.0	0.0	382.325	0.6260	**581.001**	0.5398	349.560	0.5789
0.6	0.0	0.4	0.0	0.0	386.894	0.6260	584.393	0.5331	353.313	0.5789
0.6	0.2	0.2	0.0	0.0	386.462	0.6260	583.841	0.5330	351.940	0.5789
0.6	0.3	0.1	0.0	0.0	385.770	0.6260	583.494	0.5330	351.131	0.5789
0.6	0.4	0.0	0.0	0.0	384.365	0.6260	582.048	0.5330	350.275	0.5789
0.7	0.0	0.3	0.0	0.0	386.958	0.6260	584.762	0.5330	353.671	0.5789
0.7	0.2	0.1	0.0	0.0	386.092	0.6260	584.013	0.5330	352.182	0.5789
0.7	0.3	0.0	0.0	0.0	384.492	0.6260	582.259	0.5330	350.888	0.5789
0.8	0.0	0.2	0.0	0.0	386.914	0.6260	584.897	0.5330	354.080	0.5789
0.8	0.1	0.1	0.0	0.0	386.399	0.6260	584.785	0.5330	354.148	0.5789
0.8	0.2	0.0	0.0	0.0	384.917	0.6260	582.834	0.5330	352.463	0.5789
0.9	0.0	0.1	0.0	0.0	386.501	0.6260	584.914	0.5330	354.107	0.5789
0.9	0.1	0.0	0.0	0.0	384.963	0.6260	583.341	0.5330	353.280	0.5789
1.0	0.0	0.0	0.0	0.0	384.806	0.6260	583.413	0.5330	353.265	0.5789

Table 5: Fuel consumption in gram and SoC at the end of the driving cycle for different fitness weights and the GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$. The lowest and highest fuel consumption values are typed in black and blue bold print, respectively.

Fitness weights					NEDC			FTP-75			HWFET		
w_f	w_{sl}	w_{sd}	w_{ud}	w_{ut}	Fuel cons.	SoC	$\sum \lvert . \rvert$	Fuel cons.	SoC	$\sum \lvert . \rvert$	Fuel cons.	SoC	$\sum \lvert . \rvert$
0.0	0.0	0.0	0.1	0.9	389.202	0.6259	50.89	588.098	0.5373	70.71	359.170	0.5789	42.86
0.0	0.0	0.0	0.2	0.8	389.245	0.6259	50.08	588.042	0.5368	67.92	359.214	0.5789	42.37
0.0	0.0	0.0	0.3	0.7	389.283	0.6259	49.00	588.075	0.5368	67.12	359.222	0.5789	41.40
0.0	0.0	0.0	0.4	0.6	389.294	0.6259	48.58	588.015	0.5364	66.68	359.277	0.5789	40.61
0.0	0.0	0.0	0.5	0.5	389.296	0.6259	47.43	588.122	0.5367	66.06	359.259	0.5789	40.58
0.0	0.0	0.0	0.6	0.4	389.299	0.6259	47.87	588.071	0.5364	65.53	359.265	0.5789	40.34
0.0	0.0	0.0	0.7	0.3	389.324	0.6260	47.80	588.092	0.5364	65.60	359.264	0.5789	40.76
0.0	0.0	0.0	0.8	0.2	389.319	0.6259	47.65	587.988	0.5361	65.48	359.296	0.5789	40.29
0.0	0.0	0.0	0.9	0.1	389.303	0.6259	47.07	588.086	0.5363	**64.27**	359.293	0.5789	**40.24**
0.1	0.0	0.0	0.0	0.9	388.022	0.6259	66.00	586.463	0.5330	109.28	357.307	0.5789	82.76
0.1	0.0	0.0	0.1	0.8	389.061	0.6259	51.32	587.709	0.5330	78.36	358.573	0.5789	65.41
0.1	0.0	0.0	0.2	0.7	389.187	0.6260	48.02	587.806	0.5330	74.71	358.816	0.5789	59.02
0.1	0.0	0.0	0.3	0.6	389.285	0.6259	47.70	587.864	0.5330	71.01	358.928	0.5789	55.63
0.1	0.0	0.0	0.4	0.5	389.336	0.6259	46.72	587.914	0.5330	68.96	359.016	0.5789	53.18
0.1	0.0	0.0	0.5	0.4	389.339	0.6260	46.54	587.931	0.5330	68.10	359.066	0.5789	51.10
0.1	0.0	0.0	0.6	0.3	389.386	0.6260	45.65	587.935	0.5330	67.00	359.148	0.5789	49.33
0.1	0.0	0.0	0.7	0.2	389.395	0.6259	45.95	587.955	0.5330	67.51	359.195	0.5789	47.79
0.1	0.0	0.0	0.8	0.1	389.409	0.6260	45.64	587.912	0.5330	66.88	359.194	0.5789	47.49
0.1	0.0	0.0	0.9	0.0	389.407	0.6260	45.61	587.927	0.5330	65.32	359.219	0.5789	47.29
0.2	0.0	0.0	0.0	0.8	387.964	0.6260	66.32	586.394	0.5330	111.69	357.233	0.5789	84.82
0.2	0.0	0.0	0.1	0.7	388.862	0.6260	52.86	587.596	0.5330	79.50	358.407	0.5789	70.83
0.2	0.0	0.0	0.2	0.6	389.053	0.6260	50.50	587.697	0.5330	77.70	358.579	0.5789	66.48
0.2	0.0	0.0	0.3	0.5	389.192	0.6260	48.16	587.805	0.5330	74.88	358.683	0.5789	63.11
0.2	0.0	0.0	0.4	0.4	389.222	0.6260	46.92	587.846	0.5330	71.91	358.780	0.5789	58.98
0.2	0.0	0.0	0.5	0.3	389.245	0.6259	46.58	587.853	0.5330	71.56	358.903	0.5789	57.45
0.2	0.0	0.0	0.6	0.2	389.303	0.6259	47.04	587.864	0.5330	70.93	358.947	0.5789	55.67
0.2	0.0	0.0	0.8	0.0	389.361	0.6260	**45.36**	587.941	0.5330	68.62	359.035	0.5789	52.93
0.3	0.0	0.0	0.0	0.7	387.915	0.6259	65.65	586.404	0.5330	112.00	357.190	0.5789	85.71
0.3	0.0	0.0	0.1	0.6	388.695	0.6260	54.04	587.558	0.5330	83.64	358.241	0.5789	75.00
0.3	0.0	0.0	0.2	0.5	388.960	0.6260	52.08	587.664	0.5330	80.11	358.401	0.5789	71.63
0.3	0.0	0.0	0.3	0.4	389.074	0.6259	50.04	587.739	0.5330	76.94	358.537	0.5789	67.78
0.3	0.0	0.0	0.4	0.3	389.124	0.6260	48.78	587.772	0.5330	76.76	358.646	0.5789	64.23
0.3	0.0	0.0	0.5	0.2	389.203	0.6260	48.79	587.851	0.5330	73.89	358.666	0.5789	62.98
0.3	0.0	0.0	0.7	0.0	389.255	0.6260	47.25	587.863	0.5330	72.80	358.830	0.5789	59.14
0.4	0.0	0.0	0.0	0.6	387.839	0.6260	64.69	586.393	0.5330	114.49	357.106	0.5789	89.87
0.4	0.0	0.0	0.1	0.5	388.560	0.6260	56.37	587.460	0.5330	88.13	358.105	0.5789	77.07
0.4	0.0	0.0	0.2	0.4	388.776	0.6260	53.80	587.593	0.5330	82.92	358.249	0.5789	74.00
0.4	0.0	0.0	0.3	0.3	388.936	0.6260	51.48	587.661	0.5330	82.13	358.379	0.5789	71.49
0.4	0.0	0.0	0.4	0.2	388.999	0.6259	51.16	587.699	0.5330	78.77	358.498	0.5789	67.62
0.4	0.0	0.0	0.5	0.1	389.047	0.6260	49.91	587.715	0.5330	78.16	358.548	0.5789	67.66
0.4	0.0	0.0	0.6	0.0	389.140	0.6259	49.20	587.751	0.5330	78.33	358.598	0.5789	65.91
0.5	0.0	0.0	0.0	0.5	387.807	0.6259	66.77	586.308	0.5330	115.22	357.013	0.5789	90.68
0.5	0.0	0.0	0.1	0.4	388.416	0.6259	56.60	587.374	0.5330	90.79	357.952	0.5789	84.21
0.5	0.0	0.0	0.2	0.3	388.672	0.6259	54.75	587.478	0.5330	87.42	358.078	0.5789	78.95
0.5	0.0	0.0	0.3	0.2	388.833	0.6260	53.78	587.512	0.5330	84.97	358.210	0.5789	76.78
0.5	0.0	0.0	0.4	0.1	388.945	0.6260	52.13	587.591	0.5330	84.09	358.264	0.5789	75.49
0.5	0.0	0.0	0.5	0.0	388.949	0.6259	51.92	587.587	0.5330	83.42	358.310	0.5789	72.33
0.6	0.0	0.0	0.0	0.4	387.718	0.6260	67.33	586.136	0.5330	117.69	356.878	0.5789	95.80
0.6	0.0	0.0	0.1	0.3	388.282	0.6260	57.48	587.250	0.5330	93.53	357.718	0.5789	87.96
0.6	0.0	0.0	0.2	0.2	388.551	0.6260	55.71	587.328	0.5330	91.77	357.866	0.5789	84.35
0.6	0.0	0.0	0.4	0.0	388.721	0.6260	54.67	587.417	0.5330	89.16	358.006	0.5789	81.57
0.7	0.0	0.0	0.0	0.3	387.467	0.6260	67.99	586.081	0.5330	122.30	356.574	0.5789	101.55
0.7	0.0	0.0	0.1	0.2	388.060	0.6260	59.81	587.105	0.5330	97.50	357.329	0.5789	94.22
0.7	0.0	0.0	0.3	0.0	388.384	0.6260	58.78	587.194	0.5330	93.46	357.505	0.5789	91.58
0.8	0.0	0.0	0.0	0.2	387.138	0.6259	69.37	585.938	0.5330	125.40	356.116	0.5789	108.92
0.8	0.0	0.0	0.1	0.1	387.756	0.6260	62.32	586.937	0.5330	101.95	356.990	0.5789	100.50
0.8	0.0	0.0	0.2	0.0	387.971	0.6260	60.95	586.956	0.5330	101.28	357.032	0.5789	99.35
0.9	0.0	0.0	0.0	0.1	386.840	0.6260	73.54	585.669	0.5330	132.71	355.722	0.5789	116.67
0.9	0.0	0.0	0.1	0.0	387.282	0.6260	67.19	586.622	0.5330	109.64	356.235	0.5789	107.99
1.0	0.0	0.0	0.0	0.0	384.848	0.6260	**108.31**	583.384	0.5330	**199.01**	353.271	0.5789	**156.03**

Table 6: Fuel consumption in gram, SoC, and $\sum \lvert . \rvert$ at the end of the driving cycle for different fitness weights and the GA-configuration $C_{GA} = (50, 0.75, 0.1, 100)$. The lowest and highest values for $\sum \lvert . \rvert$ are typed in black and blue bold print, respectively.

B Glossary

Acronym	Description
HEV	Hybrid electric vehicle
PHEV	Parallel hybrid electric vehicle
ICE	Internal combustion engine
EM	Electrical motor
SoC	Battery state of charge
GA	Genetic algorithm
GeneiAL	Genetic algorithm library
NEDC	New European Driving Cycle
EPA	United States Environmental Protection Agency
FTP	Federal Test Procedure
FTP-75	City program of the EPA FTP
HWFET	Highway program of the EPA FTP
DP	Dynamic programming
RDP	Receding dynamic programming
ADP	RDP with cost-to-go approximation
ECMS	Equivalent consumption minimization strategy
A-ECMS	Adaptive ECMS
T-ECMS	Telemetry ECMS

Table 7: List of acronyms used in this paper.

Variable/Constant	Description
$\omega_{ice}, \omega_{em}, \omega_{wh}$	Angular velocity at ICE/EM/wheels
T_{wh}, T_{gb}	Torque at the wheels/gear box
T_{ice}, T_{em}, T_{br}	Torque applied to the ICE/EM/brakes
h	Current gear
r_h	Gear ratio of gear h
η_{gb}	Mechanical transmission efficiency
v	Speed of the vehicle
a	Acceleration of the vehicle
r_{wh}	Wheel radius
m	Mass of the vehicle
m_r	Equivalent mass of the rotating parts of the vehicle
A	Frontal area of the vehicle
ρ	Density of air
C_d	Air drag resistance
g	Acceleration of gravity
f_r	Rolling resistance
θ	Road slope
$\dot{m}_f$	Instantaneous fuel consumption
$\dot{m}_{f,min}$, $\dot{m}_{f,max}$	Lower and upper limit of the instantaneous fuel consumption over a given prediction horizon
P_{em}	Input power of the electrical motor
$Q_{batt,max}$	Maximum cell capacity
I	Battery current
U_{oc}	Open circuit voltage
SoC	Battery state of charge
SoC_{ref}	Reference value for the SoC
SoC_{min}, SoC_{max}	Global lower and upper limit of the SoC
SoC^{loc}_{min}, SoC^{loc}_{max}	Lower and upper limit of the SoC for the end of the prediction horizon
t	Current time step
T	Total duration of the chosen driving cycle
T_p	Duration of prediction horizon
$u(t)$	Torque split at time t
u_{min}, u_{max}	Lower and upper limit for the torque split u
Δu_{max}	Maximal allowed difference between consecutive splits
$J(u(t), SoC(t), t)$	Evaluation function
K	Population size
P_0, P_i	Initial population, population in generation i
c_k	k-th chromosome of a given population
G_{max}	Maximal number of generations
ρ	Crossover rate
μ	Mutation rate
w_f, e_f	Weight and fitness function for minimizing the fuel consumption
w_{sl}, e_{sl}	Weight and fitness function for maximizing the SoC level
w_{sd}, e_{sd}	Weight and fitness function for minimizing the SoC deviation from SoC_{ref}
w_{ud}, e_{ud}	Weight and fitness function for minimizing the difference of consecutive splits
w_{ut}, e_{ut}	Weight and fitness function for minimizing the transgression of Δu_{max}
$\mathcal{N}_{\nu,\sigma}$	Normal distribution with expected value ν and standard deviation σ

Table 8: List of variables and constants used in this paper.