# Automatic Energy Management Controller Design for Hybrid Electric Vehicles

Tobias Rodemann
Honda Research Institute Europe
Carl-Legien-Strasse 30
63073 Offenbach/ Main, Germany
Email: tobias.rodemann@honda-ri.de

Lars Gräning
Honda Research Institute Europe
Carl-Legien-Strasse 30
63073 Offenbach/ Main, Germany
Email: lars.graening@honda-ri.de

Ken Nishikawa
Graduate School of Information
Science and Engineering,
Tokyo Institute of Technology,
O-okayama 2-12-1, Meguro,
Tokyo 152-8552, Japan
Email: nishikawa.k.af@m.titech.ac.jp

***Abstract*—Due to strict $CO_2$ emission limits, the optimal design of controllers for hybrid cars is an increasingly important topic for the automotive industry. Most current approaches to controller design rely solely on engineering knowledge. Utilizing technologies from computational intelligence is not yet common practice. In this work we evaluate how simple controllers can automatically be extracted from optimal control strategies computed by Dynamic Programming. We compare artificial neural network and decision tree based controllers in terms of performance (fuel consumption), stability, robustness, and interpretability, and we investigate the dependency on specific drive cycles used for generating optimal control. Our findings indicate that automatically derived controllers can result in a performance 1-2% below optimal fuel economy, but we observe a large variety in performance and in the controller structure for different drive cycles, thus, underlining the relevance of the correct choice of the drive cycles used for controller development. We also outline the impact of typical learning related issues like overfitting on the practical development process.**

## I. Introduction

The design of energy management controllers for hybrid cars is still a largely manual process. Based mostly on engineering knowledge, compact controllers are developed that determine the most efficient usage of the hybrid power sources in a car, e.g., fuel-powered combustion engine and battery-powered electric motors. In the academic community the design process is often supported by an initial determination of optimal control strategies specialized on individual drive cycles, i.e., assuming complete knowledge of the track to be driven, without any constraints on the controller architecture. Earlier [1] we have shown that both Dynamic Programming (DP) [2] and Evolutionary Algorithms (EAs) are suited to derive optimal control curves for an a-priori defined drive cycle. The optimal control curves are the basis to derive simple control rules, e.g., nested if-then-else statements or look-up-tables. This process is often done manually using visual data inspection and engineering knowledge, in a process that is probably approximated by a process akin to decision trees (see below). An alternative, more academic and basic approach, would be to use black-box learning methods like Artificial Neural Networks (ANNs).

Since driving patterns in different countries might vary, controllers will need to be adapted or even reimplemented from scratch for all major markets, thereby increasing the development costs. Here, we analyze if simple controllers automatically extracted from the optimal control results can produce a performance close to the optimal control and how the resulting control architecture depends on the selection of a specific drive cycle. Specifically, we investigate the performance loss when running a controller on a drive cycle it was not trained for. This is important when the same controller has to be used in a variety of countries with different driving characteristics.

We note that there is an extensive list of works focusing on optimizing a given controller, assuming a fixed controller architecture but some flexible control parameters (see reviews in [3], [4]). In such a setting even a many-objective optimization is feasible [5]. In the present work we investigate the preceding development step - the determination of the best controller architecture, for example in the form of a set of if-then-else rules, or the determination of the most relevant controller inputs. Obviously, the two steps are linked, since the chosen architecture limits (and guides) the parameter optimization of a controller.

We also need to mention that DP can be used to provide an optimal control given a driving profile, but due to an excessive computational demand and the difficulty in predicting driving profiles in real-world conditions, DP is so far not used online in the car during operation but mostly for assessing the upper performance limits and to derive real-world controller architecture as described in this work.

A more complete review on various strategies for hybrid car controller optimization can be found in [6].

## II. Simulation and Evaluation of Hybrid Electric Vehicles

As basis for our studies, we employed a simple parallel hybrid electric vehicle (HEV) model in Matlab as used in [7], conceptually described in [8], [9], [10]. The HEV model is simulating a vehicle where the combustion engine and the electric motor can work in parallel to provide the requested torque. The HEV model is depicted in Fig.1. It is composed of four subsystems: gearbox, internal combustion engine (ICE), electric motor (EM), and a battery. The modeled car has a total

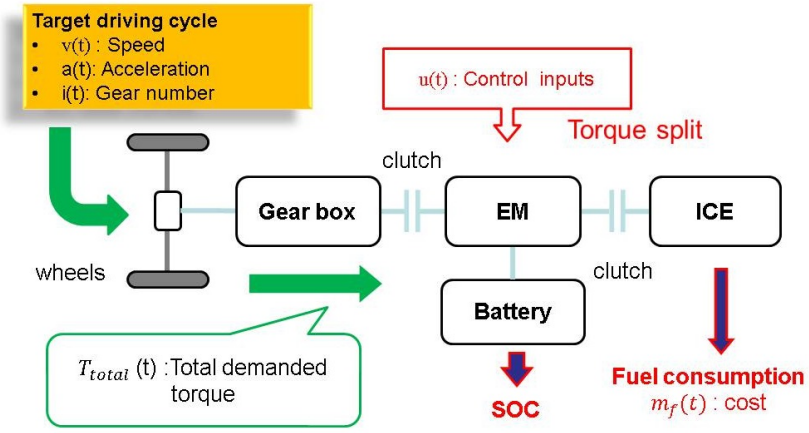



Fig. 1. Parallel hybrid electric vehicle configuration

weight of 1800 kg, a battery capacity of 6 Ah, a maximum electric motor torque of 160 Nm and a maximum ICE torque of 200 Nm. The HEV is simulated by a quasi-static discrete-time model [10]. The variation of the battery state-of-charge ($SOC$) and the total fuel mass consumption $J$ are computed by the HEV model as,

$$SOC_{t+1} = f(SOC_t, u_t, v_t, a_t, i_t) + SOC_t \quad (1)$$

$$J = \Sigma_{t=0}^{T-1} \Delta m_{fuel}(u_t, v_t, a_t, i_t) \cdot T_s \quad (2)$$

where $t$=0, 1, ..., $T$ represents discrete time steps, $SOC_t$ the battery state of charge, $u_t$ the torque split input, $v_t$ the vehicle speed, $a_t$ the vehicle acceleration, $i_t$ the gear number and $\Delta m_{fuel}(t)$ the fuel mass consumption at $t$. The time step is $T_s = 1[s]$ in this study. When $v_t$, $a_t$, and $i_t$ are known from a target driving cycle and a gear control strategy, the HEV model (1), (2) can be modified by including $t$ into the model arguments:

$$SOC_{t+1} = f_t(SOC_t, u_t, t) + SOC_t, \quad (3)$$

$$J = \Sigma_{t=0}^{T-1} \Delta m_{fuel,t}(u_t, t) \cdot T_s. \quad (4)$$

A final penalty $m_{elec}(SOC_T)$ is computed depending on the difference between the final battery level $SOC_T$ and the target value $SOC_{tar}$. The objective is to find the best torque split input $u_t$ at $t = 0, 1, ...T$, which minimizes the equivalent fuel consumption and satisfies all constraints:

$$\min_{u_t \in U_t} \Sigma_{t=0}^{T-1} (\Delta m_{fuel,t}(u_t, t)) + m_{elec}(SOC_T) \quad , \quad (5)$$

s.t. $SOC_0 = SOC_{init}$ and $SOC_t \in [SOC_{min},\ SOC_{max}]$, where $SOC_{init}$ is the initial state of charge. The $SOC_{min}$ and $SOC_{max}$ are the minimum and maximum limitation of the SOC.

With respect to the HEV model used in this work, there is a significant difference between $u(t)$=1.0 and $u(t) \in [0.0, 1.0)$ at time $t$. The model will decouple the ICE if and only if $u(t)$=1.0. In general, we have to follow two control strategies, one to decide for decoupling the ICE (binary decision) and one to control the continuous torque split value $u(t)$. For brevity we will mostly show the results for the continuous torque split control in the present work.

The performance of a hybrid electric vehicle is judged based on standardized drive cycles, which are defined as a time series that represents the speed of a vehicle at each time step. Different countries implement different drive cycles for the evaluation of HEVs in terms of fuel consumption and emissions, wherein each drive cycle targets to represent typical driving situations of a particular country. In this study, we consider driving situations based on drive cycles from six regions: *Japan, Europe, USA1, USA2, India, World*, shown in table I.

Reduction of fuel consumption and $CO_2$ emissions is one of the most important targets for the global automotive industry. For example, the European Union (EU) defined a regulation limit regarding $CO_2$ emissions from vehicles and penalty payments are imposed if the average $CO_2$ emissions of a manufacturer's fleet exceeds a limit value, that is going to decrease over time [11]. Fuel consumption and $CO_2$ emissions are measured on specified drive cycles, making these profiles highly relevant in practice. There are substantial fines for exceeding the $CO_2$ emission limits for the fleet average: up to 95 Euro per car per g /km of $CO_2$ above the limit value. This translates into approximately 100 Euro / car fines for every percent of additional $CO_2$ emission on the standardized drive cycles. As we will later see, the difference between controller architectures is often just a few percent, but this might translate into substantial costs if emission limits are violated.

## III. Learning Optimal Energy Management Control

The overall process for learning an optimally performing energy management controller for HEVs is depicted in Fig. 2. Based on one or more drive cycles and the HEV model, first, optimal control strategies are derived (using Dynamic Programming or Evolution Strategies [1]), resulting in an optimal torque split $u(t)$ at each time step $t$ along the drive cycle. The resulting optimal torque split data together with

TABLE I
DRIVE CYCLES

| name | Distance [km] | Total time [s] | Average speed [km/h] | Region | Type |
|---|---|---|---|---|---|
| JC08 | 8.17 | 1204 | 34.7 km/h | Japan | city |
| 10-15 mode | 4.16 | 661 | 33.1 km/h | Japan | city |
| UDC | 1.02 | 195 | 27.7 km/h | Europe | city |
| EUDC | 6.96 | 400 | 69.7 km/h | Europe | city |
| Artemis(urban) | 4.87 | 920 | 24.7 km/h | Europe | city |
| Artemis(road) | 17.27 | 1081 | 59.3 km/h | Europe | rural road |
| Artemis(motorway) | 29.55 | 1067 | 101.1 km/h | Europe | highway |
| UDDS | 11.99 | 1369 | 38.9 km/h | USA(1) [a] | city |
| HWFET | 16.51 | 765 | 78.2 km/h | USA(1) [a] | highway |
| IM240 | 3.15 | 240 | 49.6 km/h | USA(1) [a] | - |
| SC03 | 5.76 | 596 | 42.8 km/h | USA(1) [a] | - |
| LA92 | 15.80 | 1435 | 47.3 km/h | USA(2) [b] | - |
| India(city) | 17.49 | 2690 | 26.0 km/h | India [c] | city |
| India(high way) | 11.65 | 882 | 47.7 km/h | India [c] | highway |
| WLTC(low) | 3.09 | 589 | 25.3 km/h | World | city |
| WLTC(medium) | 4.76 | 433 | 44.5 km/h | World | city |
| WLTC(high) | 7.16 | 455 | 60.7 km/h | World | rural road |
| WLTC(extra) | 8.25 | 323 | 94.0 km/h | World | highway |

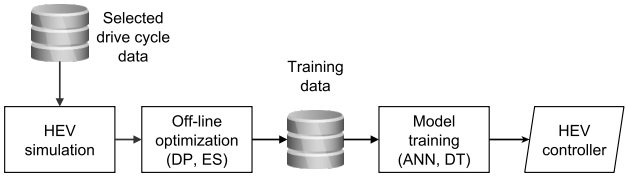



Fig. 2. Overview of the process for automatic HEV controller learning

input data like current speed, SOC or requested torque forms the training data for learning a controller, which optimally manages the two energy sources, namely ICE and electric motor. This controller would also be simple and fast enough to be used in a real car. In this work we compared two different controller models, one based on the classical artificial neural network architecture and one based on decision trees. The former method is a very basic and well-understood black-box learning approach while the latter has some similarities to the manual development process and the resulting controller structures. Both approaches have well-known limitations and flaws like overfitting, limited interpretability (for NNs) or limited modeling capacity for DTs. This study will analyze how much these issues influence the performance of the controller in this specific application case.

The resulting controller is implemented into the HEV model to decide online about the torque split, given the data of the predefined input signals for different driving conditions.

### A. Optimal Control Strategy

As stated above, finding an optimal control strategy for a drive cycle of limited length is equivalent (under the assumption of discretized time) to finding the optimal sequence of $u(t_k), k \in [1, T]$ with $T$ as the length of the drive cycle. In [1] we have investigated two different methods to compute the optimal control. In the following we summarize the most relevant findings.

Dynamic Programming (DP) [2] is commonly used to find the optimal control by means of an optimal torque split [12], [10]. The DP can deal with highly non-linear systems and derive the optimal control under time-variant complex constraints, however suffers from an exponential increase in computational costs with respect to the number of state and target variables. In the basic application [7], that we are also considering in this work, DP outperformed EAs both in terms of solution quality (fuel consumption) and run-time

Alternatively, Evolutionary Algorithms (EAs) can be adopted to search for an optimal control strategy. EAs are generic population-based meta-heuristic algorithms following concepts from natural evolution. In [1] we have shown that for a less constrained controller optimization task (the controller was allowed to control the car's speed in addition to the torque split), an EA based approach can outperform DP in both computation time and solution quality, especially for larger number of state and target variables.

In this study, we used DP to calculate the optimal control prior to the extraction of the control rules, since only a single target variable $u(t)$ and a single state variable $SOC(t)$ are under consideration.

### B. Neural Network based Controller

Inspired from biological neural networks, artificial neural networks (ANNs) are established universal function approximators often used in machine learning applications due to their good generalization capabilities. Commonly, ANNs are trained in batch mode using offline available data. In this manner, we adopt ANNs for learning the potentially complex nonlinear relationship between the input signals to the energy management controller and the torque split $u(t)$. In this study, a multi-layer feed forward neural network, a fast and simple neural network model, was used. Figure 3 depicts the multi-layer feed forward neural network structure used, which is composed of 3 input units, 4 hidden units, and one output

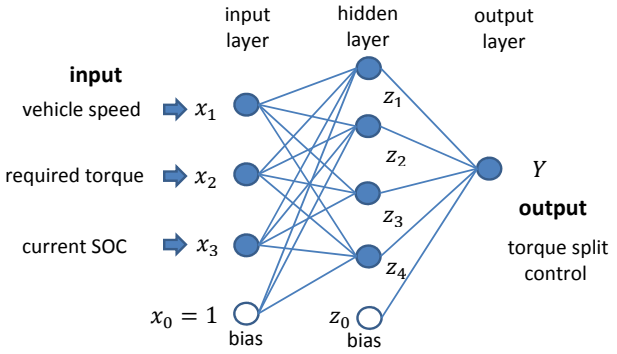



Fig. 3. Configuration of a neural network

unit. We used the local adaptive learning scheme "Rprop" [13] to optimize the weights of the neural networks. It is a first-order learning method and typically convergences faster than standard back prop. The nets were trained for 10000 epochs using Rprop. Independent of the performance of the trained ANN, the predictive model based on the neural network is "black box". It might be difficult to get any insight from the network structure and to predict how the network behaves especially when operating outside the known data range.

Two separate ANNs computed the control of torque split factor and the engine coupling decision. Therefore, we trained two types of the network controller, for the torque split factor $u$, "$u$-net", and for the decision of engine decoupling, "coupling-net". In the output layer, the $u$-net returns the sum of inputs from the hidden layer while the coupling-net uses the sigmoid function as activation function. The HEV decouples the ICE when the output of the coupling-net is larger than 0.5. The $u$-net and coupling-net were trained for 10000 and 5000 epochs using Rprop respectively.

### C. Decision Tree based Controller

The second approach for an automatic controller design is based on decision trees (DTs) [14]. DTs are supervised learning methods and are commonly used for data mining. DTs classify the training data into several generalized classes by splitting the training data into subsets based on the reference values of the input variable. This splitting procedure is represented by a tree-like graph. An example of a decision tree is shown in Fig.4.

The tree consists of attribute nodes, branches, and decision nodes. The attribute nodes represent tests based on the input variables, whereas the branches represent the outcome of the test and the decision nodes denote the predicted target output values. Algorithms for constructing decision trees work by splitting a training data set into subsets at each node using splitting criteria such as Gini impurity or information gain [15], [14] to measure the homogeneity of a split. Finally, pruning techniques are applied to reduce the complexity of the trained DT to avoid over-fitting. DTs were used as binary trees in this study. The merit of decision trees is that a tree can represent the predictive model visually and makes it easy to understand and to interpret. The resulting tree can directly be translated into a set of nested if-then-else rules. Gini's diversity index and root mean square are used as splitting criteria to predict the coupling decision and torque split value respectively. Note that the coupling-tree is a classification tree while the $u$-tree is a regression tree. When $u = 1$, the engine is disconnected from the power train and the engine drag torque is removed from the required torque $T_{dem}$. Therefore, it makes a large difference for the reduction of fuel consumption whether the torque split factor $u$ is exactly one or is close to one. The coupling-tree considers only whether the ICE should be connected to the power train or not. The $u$-tree is trained by the data taken when the ICE was connected to the power train ($-1 \leq u < 1$) and computes continuous values $u$. The $u$-tree and coupling-tree were pruned to 10 and 5 levels respectively to avoid over-fitting.

## IV. Experimental Results

For the purpose of evaluating DTs and NNs, and comparing controllers from different driving situations, we generated controllers using DTs and NNs trained by different drive cycles. We generated 50 sample drive cycles each, which were modifications of 6 types of original drive cycles. The length of each sample driving cycle was around 3600 seconds. The DP calculates the optimum torque split control for 5 different $SOC_{init}[\%] = 35$, 45, 55, 65, and 75 so that we would get various types of data in $SOC$ space. Target state $SOC_{tar}$ was 55% in all sample drive cycles. Total number of training data is about 100,000 each. DTs and NNs were trained separately for each training data set.

First of all, we compared the optimal control from the DP and a "no HEV" baseline strategy that means the EM is not used in driving ($u(t) = 0$ for all $t$), which is one of the worst possible controls. These controllers were simulated on 50 test drive cycles newly generated from each original drive cycle pattern. The average of equivalent fuel consumptions (eFCs) [g/km] on the test driving cycles are shown in table II. This result shows that the optimal control realizes approximately 20%-40% fuel reduction from the performance of the baseline control. Standard deviation for example for the relative consumption between DP and "no HEV" is between 1.3% (Japan) and 6.8% (EU), showing a certain variation in fuel consumption for different randomly assembled drive cycles, which could only be avoided by a very careful design of the drive cycles. However, general trends are remarkably stable over drive cycle variations.

TABLE II
eFC [g/km] from the "no HEV" strategy and DP, values in brackets are percentages relative "no HEV", averaged over 50 random variations of the basic drive cycles.

| | Test Driving cycle | | | | | |
|---|---|---|---|---|---|---|
| | Japan | Europe | USA1 | USA2 | India | WLTC |
| DP | 30.3(63) | 41.5(79) | 33.5(74) | 39.6(73) | 29.1(59) | 36.8(76) |
| No HEV | 47.7 | 52.2 | 45.3 | 54.4 | 49.7 | 48.5 |

We evaluated fuel reduction performance of controllers generated from DTs and NNs. When control of DTs or NNs

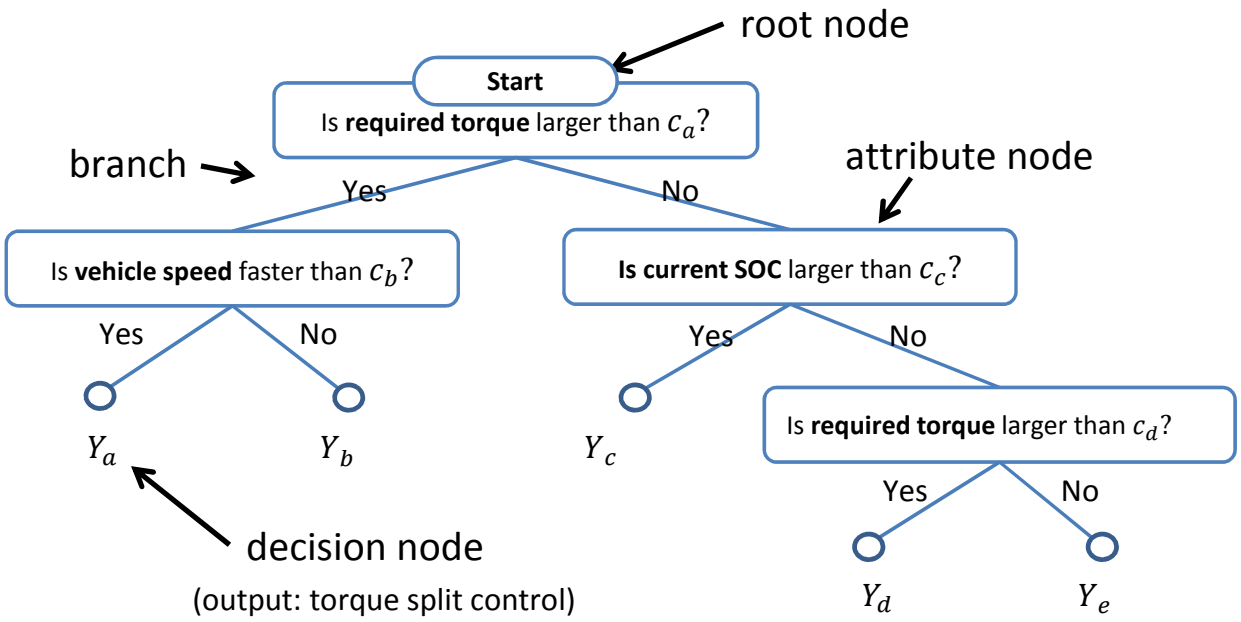



Fig. 4. An example of a decision tree: $c_a, c_b, c_c$, and $c_d$ are reference values regarding three inputs, vehicle speed, required torque, and current SOC. $Y_a, Y_b, Y_c, Y_d$, and $Y_e$ are predicted torque split factors.

exceeds limitations, the control was replaced by feasible maximum/minimum values to satisfy the constraint using predefined safety filters. Fig. 5 shows the average of equivalent fuel consumption (eFC) [g/km] on the test drive cycles from the DT and the NN controller relative to the values from the DP averaged over 50 different random variations of the basic drive cycles.

When training data was equal to test data, the performance of DTs and the NNs is not very different. It indicates that both methods learned the training data correctly. Also fuel consumption is just 1-2% above the value from the DP. However, when training drive cycle and test drive cycle are not the same, performance of trained controllers gets worse for particular data sets such as USA2 and India because $SOC$ reaches minimum or maximum limits. Here, fuel consumption values more than 10% above DP levels are possible. Very good generalization is achieved for the WLTC protocol as training data which gives good results on all test drive cycles.

To find detailed characteristics of controllers, we compared control policies extracted by the DTs and the NNs. The models of the DTs and the NNs have three input variable, $v, T_{couple}$, and $SOC$, and one output $u$. Figure 6 shows as an example the control policy maps from the DTs and the NNs trained from drive cycle sets for Japan. The policy from the DP at the indicated SOC level (as 3rd input) is shown on the left side. The control policies from the DTs and the NNs are shown in the middle and on the right side. Ranges colored by dark brown in dotted lines indicate areas where the HEV should employ only the EM to provide all required power and the ICE should be decoupled from power train. The figure of control policies shows five basic types of decisions (in some other tests, also invalid outputs were generated by NNs):

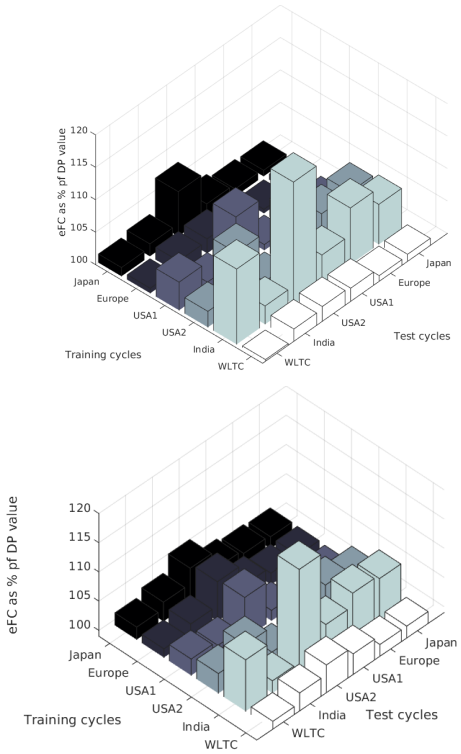



Fig. 5. eFC for DTs (*top*) and NNs (*bottom*) relative to the value for DP with different training and test data, averaged over 50 randomly generated test cycles. Standard deviations over the 50 test cycles were on average 1.7 % for DT and 1.3 % for NN.

- $u = 1$, $T_{couple} > 0$ (dark brown): pure electric driving,
- $u = 1$, $T_{couple} < 0$ (dark brown): full regenerative braking,
- $u < 0$ (blue): recharging battery by the ICE,
- $u \approx 0$ (green): driving using only the ICE,
- $0 < u < 1$, $T_{couple} < 0$ (red or yellow): partly regenerative braking,

The control map of the DT shows that the DT could extract the basic structure of the DP data, including the striped pattern for higher torque at $SOC = 55\%$ which is due to gear shift effects. The NN in this example did not extract the partly regenerative braking area because of too few training data. The NN also smoothly interpolated the gear shift pattern. For some drive cycles we observed that NNs produced invalid outputs even if the training data does not contain any invalid outputs.

We also compared the control rules extracted by DTs from different drive cycles. Figure 7 shows examples for extracted control trees using three different drive cycles. For most data sets trees had the criterion of torque as the first (top) node to separate the decision for regenerative braking. For the remainder, trees could be classified into three types depending on the type of dominant input that is placed on the second highest level. We found all three inputs as second level decisions nodes and a large variety further down the decision tree, meaning that the basic structure of the controller differs substantially depending on the type of drive cycle used. It is therefore unlikely that a controller can be adapted to a specific drive cycle by simply changing decision nodes or attribute node thresholds within a fixed architecture.

Is is also interesting to note that the top three decision layers use different inputs: for Japan it is torque, speed and SOC, but for Europe and the US only torque and speed are relevant. Based on EU or US data, the developer might be tempted to ignore SOC as controller input, limiting the potential for adapting this very controller to Japanese conditions.

## V. Discussion and Outlook

We have shown that DTs and NNs can learn on-line torque split control from the results of the offline optimization using DP. Control policy maps from the DTs and the NNs showed different characteristics, but resulted in similar performance levels. The training with different drive cycles than those used for testing can result in significant performance losses. Comparing DTs and NNs we found that DTs are better capable of modeling discontinuities in the torque split because DTs implement crisp decision boundaries. In contrast, NNs attempt to approximate the training data using continuous base functions. DTs are also better to derive control rules from only a few samples, which are often ignored by NNs. However, DTs might not fit well if more complex decision boundaries have to be modeled. An additional benefit of DTs is that they are much easier to interpret and to adapt like for incorporating engineering constraints than NNs. For example, we learned from the analysis of DTs that controllers from different regions could be structurally different, providing evidence for the need of a region specific controller adaptation.

The presented work is basically a study to assess the current situation and the potential for nature-inspired methods like EAs and NNs to improve the process of controller development in various directions. A future target is the integration of methods for optimal control and automatic rule extraction for arbitrary drive cycles and a better assessment of the quality of extracted rules in terms of performance and stability, including robustness to unseen input parameter configurations. A more automated process for controller design is necessary due to the potentially large performance differences between controllers trained for different drive cycles. If controllers can be quickly adapted to the specific driving conditions, development costs could be reduced substantially. But we have also seen that an automatically derived controller might provide invalid controls. A potential remedy could be a cascade or modular architecture of controllers, with individual parts that are flexible and can automatically be learned and some parts being fixed, e.g., to certify parts that would guarantee a safe operation of the car or that guarantee a minimum driving efficiency. How a controller can be modularized and learned in such a way is an open research question.

In our previous work [1] we also discovered that DP will probably not be able to provide optimal control curves for more complex scenarios. We are therefore planning to derive optimal control using Evolutionary Algorithms, which would also open up possibilities to include multiple additional objectives like emissions, drivability, battery lifetime or others into the controller design process similar to the work described in [5].

## Acknowledgments

Ken Nishikawa acknowledges the financial support from Honda Research Institute Europe.

## References

[1] T. Rodemann and K. Nishikawa, "Can evolutionary algorithms beat dynamic programming for hybrid car control?" in *EvoApplications*, ser. LNCS 9597, G. Squillero and P. Burelli, Eds., vol. 1. Springer, 2016, pp. 1–14.

[2] R. Bellman, "Dynamic programming and stochastic control processes," *Information and control*, vol. 1, no. 3, pp. 228–239, 1958.

[3] E. Silvas, T. Hofman, and M. Steinbuch, "Review of optimal design strategies for hybrid electric vehicles," in *IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling*, vol. 3, no. 1, 2012, pp. 57–74.

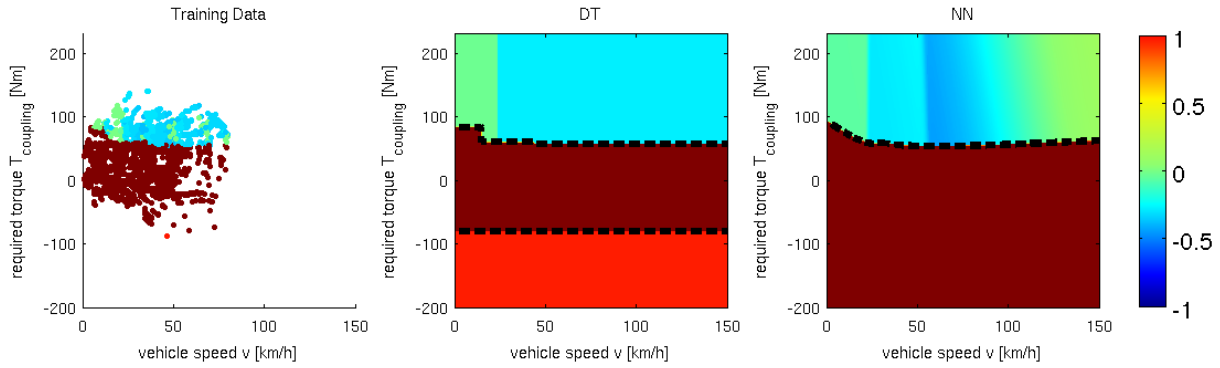
[4] F. R. Salmasi, "Control strategies for hybrid electric vehicles: Evolution, classification, comparison, and future trends," *Vehicular Technology, IEEE Transactions on*, vol. 56, no. 5, pp. 2393–2404, 2007.

[5] T. Rodemann, K. Narukawa, M. Fischer, and M. Awada, "Many-objective optimization of a hybrid car controller," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, A. M. Mora and G. Squillero, Eds., vol. 9028. Springer International Publishing, 2015, pp. 593–603.

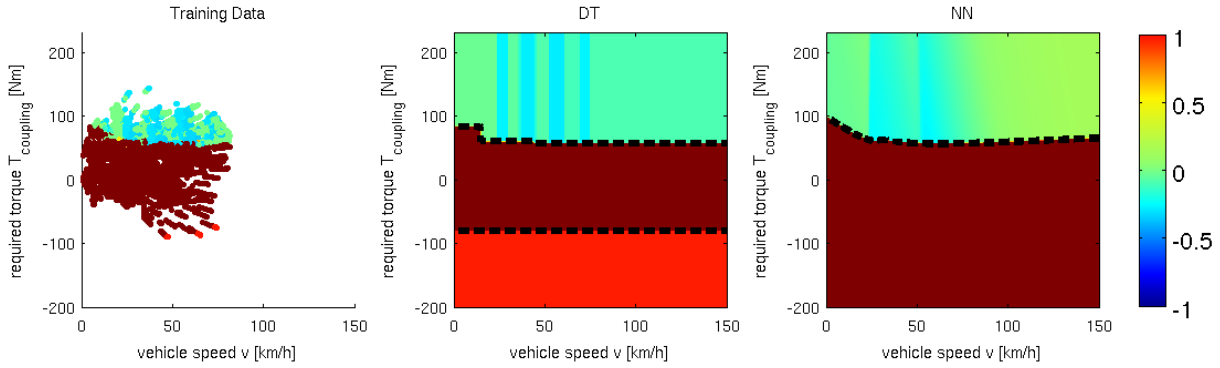
[6] E. Silvas, T. Hofman, N. Murgovski, P. Etman, and M. Steinbuch, "Review of optimization strategies for system-level design in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2016.

[7] O. Sundström and L. Guzzella, "A generic dynamic programming Matlab function," in *Control Applications, (CCA) Intelligent Control, (ISIC), IEEE*, 2009, pp. 1625–1630.

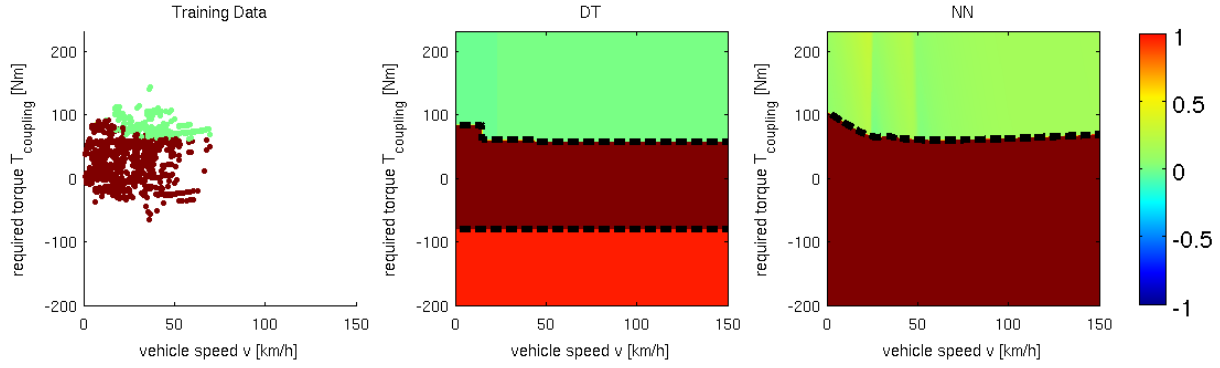
[8] L. Guzzella and C. Onder, *Introduction to Modeling and Control of Internal Combustion Engine Systems*. Springer, 2004.




(a) at minimum SOC ($SOC = 30\%$)

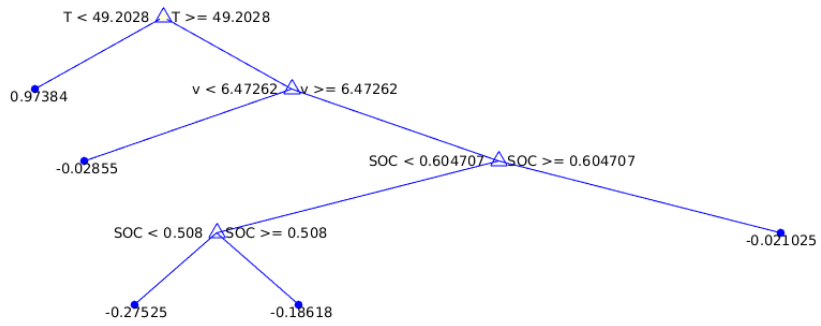



(b) at middle SOC ($SOC = 55\%$)

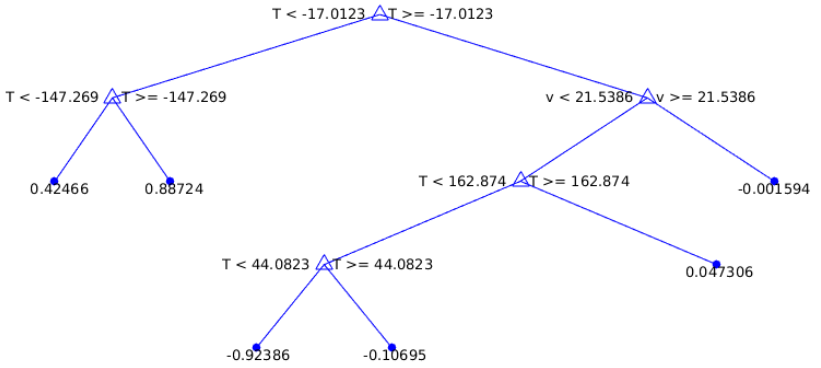



(c) at maximum SOC ($SOC = 80\%$)

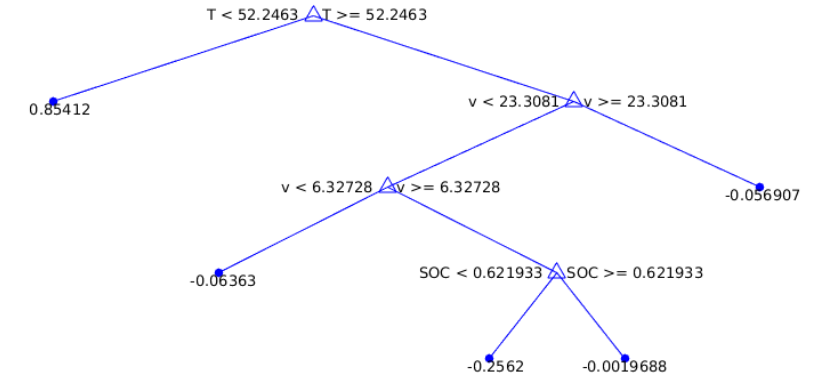
Fig. 6. Control policy maps of the DT and the NN from the Japanese driving cycles: Ranges colored by dark brown in dotted lines indicate areas where the ICE should be decoupled from power train. The three rows present control policy maps for three different current SOC level. Note the substantial structural differences between especially NNs and DTs.




(a) Japan




(b) Europe




(c) US

Fig. 7. Decision 'u' tree from Japanese (top), European (center), US (bottom) drive cycles. Note that the decoupling tree is not shown. Trees structurally differ already at the second level from the top. Note that the details of the trees have been pruned to increase visibility.


[9] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems: Introduction to Modeling and Optimization*. Springer Berlin Heidelberg, 2005. [Online]. Available: http://books.google.de/books?id=zMEsWs9S7u8C

[10] O. Sundström, L. Guzzella, and P. Soltic, "Optimal hybridization in two parallel hybrid electric vehicles using dynamic programming," *Proceedings of the 17th IFAC world congress*, vol. 17, no. 1, pp. 4642–4647, May 2013.

[11] "Regulation (EU) No 333/2014 of the European Parliament and of the Council of 11 March 2014 amending Regulation (EC) No 443/2009 to define the modalities for reaching the 2020 target to reduce $CO_2$ emissions from new passenger cars," pp. 15–21, April 2014, official Journal of the European Union, L103, p.15-21.

[12] F. Millo, L. Rolando, R. Fuso, and F. Mallamo, "Real CO2 emissions benefits and end users operating costs of a plug-in hybrid electric vehicle," *Applied Energy*, vol. 114, pp. 563–571, 2014.

[13] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *Neural Networks, 1993., IEEE International Conference on*, 1993, pp. 586–591 vol.1.

[14] L. Rokach and O. Maimon, *Data Mining with Decision Trees: Theory and Applications*. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2008.

[15] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.