A New Type of Metamodel for Longitudinal Dynamics Optimization of Hybrid Electric Vehicles

Christopher Bacher[1], Günther R. Raidl[1], and Thorsten Krenek[2]

[1] Institute of Computer Graphics and Algorithms, TU Wien, Vienna, Austria
{bacher,raidl}@ac.tuwien.ac.at
[2] Institute for Powertrains and Automotive Technology, TU Wien, Vienna, Austria
thorsten.krenek@ifa.tuwien.ac.at

Abstract. Parameter optimization for Hybrid Electric Vehicles (HEVs) is very time consuming due to the necessity to evaluate a simulation model. This bottleneck is usually removed by using metamodels as surrogates for the simulation. We consider metamodels for longitudinal dynamics simulation, which simulate a vehicle following a given driving cycle. Typical "top-down" metamodels are parametrized by both the HEV model and the driving cycle. We propose a novel bottom-up metamodelling scheme only parametrized by the HEV model and discuss preliminary results.

Keywords: hybrid electric vehicle, optimization, metamodel, surrogate

1 Introduction

Hybrid Electric Vehicles (HEVs) and their related concepts—on the fringes just a few years ago—become increasingly common in automobiles today. Technically, HEVs combine aspects of conventional vehicles and electric vehicles. They possess an Internal Combustion Engine (ICE) using conventional fuel and one or more Electric Machines (EMs) with alternative energy storages. As energy storage systems batteries are most commonly used but also variants with fuel cells have been considered.

The reasons for the increasing popularity of HEVs are manifold. On one hand there are environmental considerations. HEVs are considered to be more environmentally friendly than conventional vehicles. Their ability to use EMs for propulsion leads to decreased pollutant emission. In the same breath increased fuel efficiency—and thereby cost reduction for the customer—is often named as a main benefit of HEVs, if not overshadowed by high initial costs.

On the other hand one finds that hybridization of vehicles is graded and can be found in many vehicles today. For example sports car manufacturers use EMs for boosting, i.e. faster acceleration in comparison to a common ICE.

In any use case the effective use of the hybrid components is critical for the performance and efficiency of a HEV. Proper sizes for the ICE, EMs, battery capacities and gear sets have to be chosen, as well as an effective operation strategy deciding which components are active under what circumstances.

One solution to this problem is to perform optimization using an accurate computer model for simulating the behaviour of the HEV under different circumstances. In the following we consider longitudinal dynamics simulations. This simulation type simulates the behaviour of the HEV on a driving cycle defining target velocities for each moment of the simulation.

The task of finding appropriate parameter settings for the according computer model poses a difficult continuous optimization problem. Manually establishing a good solution to this high-dimensional problem is neither effective nor efficient. Therefore metaheuristic algorithms are used for this purpose.

However the main drawback of their straight-forward use in this setting is the time-intensive evaluation of the objective function which involves the simulation of the chosen parameter configuration for the HEV. Simulation times for HEV models vary on their degree of detail and the length of the selected driving cycle, but can easily reach five to twenty minutes per test case in our experiments with high-detail models. A possible solution to this problem is the development of metamodels acting as surrogates for the simulation in the optimization process.

In the following sections we review related work, discuss the usage of metamodels in HEV optimization and outline their drawbacks. Afterwards we introduce a new metamodelling technique addressing these problems and provide results for preliminary experiments.

2 Related work

Optimization of HEVs has been discussed by many authors. Johnson et al. [4] optimize the parameters of a real-time operational strategy. The optimization is carried out on a surrogate based on a surface fitting model obtained from a Design-Of-Experiments for the original simulation to deal with long simulation times. Multi-objective optimization is treated in [3] and more recently by Rodemann et al. in [6]. In both [4] and [3] the simulation software ADVISOR is used, whereas [6] uses Matlab/Simulink as simulation tool. Both [3] and [6] do not have to deal with long simulation times, as in both cases faster, less detailed models are used. This is explicitly stated in [6] and from the authors' experience this also holds true for ADVISOR models as used in [3].

Optimization of simulation models with high computation times is treated in [2] and [5]. There different regression models based on neural networks are used as surrogates in metaheuristics. Both works use GT Suite as simulation software which is also used for the experiments in Section 5.

3 Metamodels and HEV Optimization

To optimize HEVs we rely on metaheuristic approaches like Genetic Algorithms and Particle Swarm Optimization as described in [2]. The metaheuristics use the

HEV model by setting its parameters and simulating it according to a prespecified driving cycle. A driving cycle specifies a target velocity for any point in time which the vehicle has to match.

In the authors' previous work on parameter optimization for HEVs it became necessary to develop surrogates for the simulation. As the objective function relies on performing a simulation for any tested parameter configuration and to neglect the high time requirements to perform the simulation, it is inefficient to use the model directly. A better possibility is to develop a faster but coarse version of the HEV model for optimization. Manually developing such a model and calibrating it to match the original model or even the HEV itself would incur substantial time and financial costs. Moreover if the optimization is run as part of the development of an actual vehicle, keeping the different models synchronized with the changes would likely be prohibitive.

Therefore automatically deduced regression models are created as a replacement to the simulation. The regression models used in previous work are different variants of neural networks which are trained on results from a sampling phase during optimization. The inputs of the networks are all varied parameters of the HEV model and its output is a prediction of the according objective value. The results in [2] show that the use of these surrogates is effective and an altered optimization scheme in [5] seems to even improve the overall performance.

Nevertheless the use of neural networks as metamodels comes with a substantial drawback. The fitted metamodel is only valid for a single driving cycle. The time-expensive collection of training data hampers the usability of the metamodel as it has to be created anew for each driving cycle.

Though optimizing the HEV for a variety of driving cycles is deemed. Otherwise the final optimized parameter configuration might overfit the prespecified driving cycle resulting in poor performance in real usage scenarios. A better approach would be to optimize the vehicle over a weighted set of driving cycles which represent typical use cases of potential drivers. To this end we propose a new metamodelling scheme discussed in the next section.

4 Bottom-Up Metamodels for HEVs

To deal with the above-mentioned reusability issue we diverge from the "top-down" perspective of the regression models for HEVs simulation, where metamodels represent the simulation as an atomic unit. This viewpoint on regression lacks generalization across driving cycles.

In [1] several possibilities to enhance the prediction performance of the neural networks are explored. Notably two concepts—the Partial Simulation and the Time-Progressive Learning Ensemble (TPLE) approaches—try to split the prediction of the simulation into several parts. The Partial Simulation approach splits the driving cycle, uses the original model to simulate the first part of the cycle and passes the resulting fuel consumption along with the parameter configuration of the HEV to a neural network to predict the final objective value. TPLE on the other hand splits the driving cycle into several parts and trains separate regression models to predict the fuel consumptions for each part. Starting with the first part, the prediction result is iteratively passed to the next slice until a final objective value is obtained. Both variants show good prediction behaviour in experiments. Our bottom-up approach relies on similar decompositions and composes the final result from a chain of predictions for parts of a driving cycle.

4.1 Formal description of Bottom-Up Metamodels

Formally let D be the driving cycle of length $|D|$ to be predicted and $\mathcal{S}$ be the set of all driving scenarios, and $\mathcal{S}_T \subset \mathcal{S}$ a set of driving scenarios chosen for training. Driving scenarios are short linear driving cycles acting as building blocks which we use to reconstruct the whole driving cycle D. A scenario s is defined by the triple $(v_\mathrm{b}, v_\mathrm{e}, a)$ where $v_\mathrm{b}, v_\mathrm{e}$ specify the start and end velocity of the scenario and a the respective acceleration. Observe that $\mathcal{S}_T$ can be chosen independently from D. Selection strategies for $\mathcal{S}_T$ will be discussed below.

Let $\mathcal{P}$ be the discretized search space of the optimization problem and $\mathbf{p} \in \mathcal{P}$ be a candidate solution. The continuous search space is discretized due to performance reasons and due to insignificant changes in the objective function. This matter is discussed in more detail in [2].

Let $\mathcal{P}_\mathcal{A}$ be $\mathcal{P}$'s dimensions restricted to the set of operational parameters. Operational parameters are parameters which affect the HEVs operation strategy. The operation strategy $\mathcal{A} : \mathcal{P}_\mathcal{A} \times \mathcal{M} \times \mathcal{I} \rightarrow \mathcal{M}$ determines the next operation mode $m \in \mathcal{M}$ of the HEV. A mode m defines the active components and their use during driving, e.g. pure electric, range extension, power split or recuperation. Modes for an exemplary vehicle are described in Section 5. Further $\mathcal{I}$ is the set of input signals to $\mathcal{A}$ on which a the mode switch depends, like State of Charge (SOC) of the battery, axle torque, or power demand. Observe that $\mathcal{A}$ forms a state machine with rule-based state changes. $\mathcal{I}$ itself is a subset of the metamodels outputs, i.e. $\mathcal{I} \subseteq \mathcal{O}$, as the state of $\mathcal{I}$ has to be predicted throughout the evaluation to determine the current operation mode.

Let $\mathcal{P}_s$ be the set of structural parameters. Structural parameters affect the simulation results of a driving scenario if the operation mode is fixed, e.g. gear ratios or engine sizing.

Finally let $\phi^\mathcal{A}_{\mathcal{S}_T} : \mathcal{P} \times \mathcal{D} \rightarrow \mathcal{O}$ be the bottom-up metamodel parametrized by the driving scenarios and the operation strategy to predict values of the output signals $\mathbf{o} \in \mathcal{O}$ for a given candidate solution $\mathbf{p}^* \in \mathcal{P}$ and a driving cycle $D \in \mathcal{D}$. The metamodel $\phi^\mathcal{A}_{\mathcal{S}_T}$ is trained by creating scenario models $\varphi^m_{\mathbf{p}_s,\mathbf{j},s} : \mathcal{P}_s \times \mathcal{J} \times \mathcal{S}_T \rightarrow \varDelta\mathcal{O}$ by recording the results of simulating all combinations of $s \in \mathcal{S}_T$, $m \in \mathcal{M}$, a subset of $\mathbf{p}_s \in \mathcal{P}_s$, and a subset of $\mathbf{j} \in \mathcal{J}$. The inputs $\mathbf{j} \in \mathcal{J}$, $\mathcal{J} \subseteq \mathcal{O}$ are additional discretized signals affecting the simulation results, e.g. SOC or power demand. The subsets of $\mathcal{P}_s$ and $\mathcal{J}$ can be obtained by a properly chosen Design-of-Experiments (DoE), e.g. full factorial if the domains are small.

The scenario models $\varphi^m_{\mathbf{p}_s,\mathbf{j},s}$ can then be used to predict the output vector $\mathbf{o}$ for given $\mathbf{p}^*_s$, $\mathbf{j}^*$ (not necessarily contained in the DoE), and a scenario $s^* \in \mathcal{S}$. The results of the single scenario models are then combined by a machine learning method $f_{\mathbf{p}^*_s,s^*} : \{\varDelta\mathcal{O}\} \rightarrow \varDelta\mathcal{O}$ such as k-Nearest-Neighbour or neural networks. The output of the bottom-up metamodel $\phi^{\mathcal{A}}_{\mathcal{S}_T}$ is obtained by the following recursion.

$$\phi^{\mathcal{A}}_{\mathcal{S}_T}(\mathbf{p}^*, D) = \mathbf{o}_{|D|-1} \tag{1}$$

$$\mathbf{o}_t = \mathbf{o}_{t-1} \oplus f_{\mathbf{p}^*, s_t}\left(\{\varphi^{m_t}_{\mathbf{p}_s, \mathbf{j}, s}(\mathbf{p}^*, \mathbf{o}_{t-1}, s_t) \mid \forall \mathbf{p}_s, \mathbf{j} \in \mathrm{DoE}(\mathcal{P}_s, \mathcal{J}), s \in \mathcal{S}_T\}\right) \tag{2}$$

$$m_t = \mathcal{A}\left(\mathbf{p}^*, m_{t-1}, \mathbf{o}_{t-1}\right) \tag{3}$$

$$s_t = (v_{\mathrm{b}}(D,t), v_{\mathrm{e}}(D, t+1), a(D,t)) \tag{4}$$

Basically the metamodel determines for each point in time t the active mode m_t, uses $f_{\mathbf{p}^*_s, s^*}$ and the scenario models to predict the change in the output signals and joins the results using $\oplus$ with the results from the previous time step. Depending on the predicted signal $\oplus$ might be a simple replacement or an addition. Of course sensible values for m_0 and $\mathbf{o}_0$ have to be chosen. Observe that depending on the implementation of $f_{\mathbf{p}^*_s, s^*}$ it is probably not necessary to evaluate all $\varphi^{m_t}_{\mathbf{p}_s, \mathrm{inf}, s}$ but only a small subset thereof.

4.2 Obtaining training scenarios

In the following we discuss the selection of training scenarios $\mathcal{S}_T$. From the explanations above it is clear that the choice of $\mathcal{S}_T$ is highly relevant to the performance of the metamodel. It is to be expected that a more fine grained set of scenarios produces more accurate results. Nevertheless if $\mathcal{S}_T$ is chosen too large then the simulation overhead can be significantly worse if compared to obtaining training data for top-down metamodels. Moreover the simulation time for the driving scenarios is without direct use to the optimization process as D is not evaluated. Therefore there are two major ways to choose $\mathcal{S}_T$.

1. Choose $\mathcal{S}_T$ s.t. a wide range of possible driving cycles can be approximated in future use, e.g. by equidistantly sampling the scenario space or some other DoE variant.
2. If the set of driving cycles $\mathcal{D}$ to be predicted is known beforehand, e.g. major standardized driving cycles used for official emission rating (US06, NEDC, WLTC3), then it is reasonable to trim $\mathcal{S}_T$ to yield higher accuracy on $\mathcal{D}$.

We propose a compression heuristic for the second case. Let ε_a, ε_v be the minimal considered change in acceleration/velocity.

1. Split each cycle D into scenarios of duration Δt and categorize them in transient (acceleration, breaking) scenarios $\mathcal{S}_A$ and constant speed scenarios $\mathcal{S}_C$ (if $a \leq \varepsilon_a$).
2. Sort the constant velocity scenarios by velocity and collect all scenarios from s_0 to s_i greedily until

$$v(s_{i+1}) - (v(s_i) + \varepsilon_v) > \varepsilon_v \tag{5}$$

   Then create a scenario with constant speed equal to the average speed of the the collected scenarios, add it to the training set and proceed with $s_0 = s_i$.

3. The following is carried out for acceleration and breaking scenarios separately. Below the acceleration case is explained, the breaking case is analogous respective to some sign changes. Sort the transient scenarios by acceleration and split them into chunks whenever $|a(s_{i+1}) - a(s_i)| > \varepsilon_a$.
4. For each transient chunk scenarios are iteratively taken from both ends of the sorted list. Even numbered elements are taken from the left end and odd numbered elements from the right. A list of sets $\mathcal{X}_k$ is kept where each set is identified by the acceleration $a_k = a_i + \varepsilon_a$ of the scenario s_i first added to the set. For the current scenario s_i all sets are identified s.t. $|a_k - a_i| \leq \varepsilon_a$. If no such set is found then a new set is created and s_i is added as its first element. Otherwise s_i is added to the set $\mathcal{X}_k$ whose duration

$$d_k = \frac{\max\limits_{s \in \mathcal{X}_k}(v_\mathrm{e}(s)) - \min\limits_{s \in \mathcal{X}_k}(v_\mathrm{b}(s))}{a_k} \tag{6}$$

changes least if s_i is added. After processing all scenarios, the scenario

$$\left(\min_{s \in \mathcal{X}_k}(v_\mathrm{b}(s)), \max_{s \in \mathcal{X}_k}(v_\mathrm{e}(s)), \operatorname*{avg}_{s \in \mathcal{X}_k}(a(s)) \right) \tag{7}$$

is added to the training set for each $\mathcal{X}_k$.

5 Experiments

We implemented a prototype of the bottom-up metamodel for the HEV model used in [2] and [1] (Model A). For a more complete description the reader is advised to consult these sources. The vehicle possesses an ICE and two EMs coupled by a planetary gear set. There are four different propulsion modes and two recuperation modes available.

– **EV1/2** use only one or two EMs for propulsion and consume battery charge.
– **ER1** uses the larger EM for propulsion and the smaller one as a generator powered by the ICE. Consumes battery but also charges the battery. The amount of charging depends on the requested charge power which has been identified as an internal signal in $\mathcal{J}$ to be predicted by the metamodel.
– **ER2** is a power split mode using both the larger EM and the ICE for propulsion. Further the load point of the ICE is shifted to a higher, more efficient level. The resulting surplus energy is used to charge battery via the smaller EM. The load point shift is determined by the charge power signal.
– **Recup1/2** are recuperation modes, i.e. performing regenerative braking with a single or both EMs active respectively.

The twelve parameters forming $\mathcal{P}$ of the model are the same as in [2] and [1] with the same discretization levels. Of those parameters we identified the number of ring and sun teeth of the planetary gear set as the only structural parameters.

The remaining parameters are considered operational parameters. The output set $\mathcal{O}$ has been identified to consist of the fuel consumption, SOC, charge power, power demand, and axle torque. The only signal in $\mathcal{J}$ is the requested charge power. For this signal five equidistantly spaced power levels have been chosen to be included in the coarse training set (explained below) and six for the more fine grained set. For the structural parameters $\mathcal{P}_s$ and the internal signals $\mathcal{J}$ a full factorial design has been chosen. All scenario models are evaluated prematurely and their results are cached for future use.

As combination method $f_{\mathbf{P}_s^*, s^*}$, a k-Nearest-Neighbour (kNN) model for every combination of mode, structural parameters and internal signal $\mathcal{J}$ is built, once for fuel consumption and once for SOC. The remaining output signals can be calculated directly by mathematical relationships with information from the previous time step.

To evaluate the quality of the fit, the Mean Total Order Deviation (MTOD) is considered. The optimization algorithms in [2] do not use the absolute objective values of candidate solutions to select among them but their ranks, e.g. by using tournament selection in genetic algorithms. MTOD takes this into account by ordering the sequence of expected output values and the sequence of the actual output values. Then for each candidate solution the percental shift between its positions in both lists is calculated. The mean of those percental shifts forms the MTOD which measures rank errors, but not the absolute deviation between the expected and actual output values.

The 903 evaluated candidate solutions are taken from Phase I in [2] s.t. the MTOD results for both model types are comparable. Simulations of random shuffling show an empirical MTOD of about 0.33. The experiments have been carried out with two different $\mathcal{S}_T$, one coarse and one more fine grained. Both sets are obtained by the procedure described in Section 4 by compressing the US06, NEDC, and WLTC3 with Δt =1sec all other units use km/h as base unit.

Table 1 depicts the MTOD for the objective function, the fuel consumption, and the SOC. It can be seen that low values for k yield better results and that the fine grained training set outperforms the coarse one. Nevertheless if compared to the results in [2] the MTOD of a simple neural network is only 0.0952. To discover the reason for this discrepancy, additional experiments have been run where the operation strategy is removed from the metamodel and the same mode switches as in the respective simulations are performed instead. An additional $\mathcal{S}_T$ is evaluated which consists of scenarios sampled at four second intervals from the US06 cycle without compression and predicted with $k = 1$. This results in a close reconstruction of the US06 driving cycle by the metamodel.

Table 2 shows these baseline errors, i.e. errors not resulting from a “wrong” mode switch. Surprisingly the results are worse than in the standard strategy variant. From the current state of experiments no conclusions about the reason for this behaviour can be made. The results are subject to further investigations.

Table 1: MTOD results for US06 standard strategy experiments

(a) Results with $\varepsilon_a = 0.4, \varepsilon_v = 2.5$

k	obj.	fuel	SOC
1	0.2072	0.1743	0.2158
2	0.1898	0.1868	0.2189
3	0.2182	0.2380	0.2649
4	0.2251	0.2478	0.2629

(b) Results with $\varepsilon_a = 0.7, \varepsilon_v = 2.5$

k	obj.	fuel	SOC
1	0.2279	0.1615	0.2555
2	0.2182	0.1687	0.2522
3	0.2480	0.2302	0.2468
4	0.2483	0.2255	0.2478

Table 2: MTOD baseline errors for US06

(a) $\varepsilon_a = 0.4, \varepsilon_v = 2.5$

k	obj.	fuel	SOC
1	0.2920	0.1640	0.2834
2	0.2670	0.1207	0.2784
3	0.2791	0.2003	0.2981
4	0.2872	0.2750	0.3061

(b) $\varepsilon_a = 0.7, \varepsilon_v = 2.5$

k	obj.	fuel	SOC
1	0.2564	0.2622	0.2854
2	0.2383	0.1550	0.2696
3	0.2814	0.2830	0.3000
4	0.2845	0.3710	0.3114

(c) $\mathcal{S}_T$ = US06, $\mathit{\Delta}t$ = 4sec

k	obj.	fuel	SOC
1	0.2111	0.1278	0.3253

6 Conclusion and Future Work

In this paper we introduce bottom-up metamodels for HEV optimization which are able to generalize over different driving cycles at no additional simulation costs. While those concepts appear to be highly promising, our preliminary experimental results for prediction performance are behind expectations for unclear reasons and under further investigation. In future work optimization techniques exploiting the metamodel structure shall be developed.

References

1. Bacher, C.: Metaheuristic optimization of electro-hybrid powertrains using machine learning techniques. Master's thesis, TU Wien, Vienna, Austria (2013)
2. Bacher, C., Krenek, T., Raidl, G.: Reducing the number of simulations in operation strategy optimization for hybrid electric vehicles. In: Esparcia-Alczar, A.I., Mora, A.M. (eds.) Applications of Evolutionary Computation, Lecture Notes in Computer Science, vol. 8602, pp. 553–564. Springer Berlin Heidelberg (2014)
3. Hu, X., Wang, Z., Liao, L.: Multi-objective optimization of HEV fuel economy and emissions using evolutionary computation. In: Society of Automotive Engineers World Congress and Exhibition. vol. SP-1856, pp. 117–128 (2004)
4. Johnson, V.H., Wipke, K.B., Rausen, D.J.: HEV control strategy for real-time optimization of fuel economy and emissions. Society of Automotive Engineers transactions 109(3), 1677–1690 (2000)
5. Krenek, T., Bacher, C., Lauer, T., Raidl, G.R.: Numerical optimisation of electro-hybrid powertrains. MTZ Worldwide 76(3), 46–52 (2015)
6. Rodemann, T., Narukawa, K., Fischer, M., Awada, M.: Many-objective optimization of a hybrid car controller. In: Mora, A.M., Squillero, G. (eds.) Applications of Evolutionary Computation, Lecture Notes in Computer Science, vol. 9028, pp. 593–603. Springer International Publishing (2015)