Reducing the Number of Simulations in Operation Strategy Optimization for Hybrid Electric Vehicles

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

[1] Institute of Computer Graphics and Algorithms, Vienna University of Technology, Vienna, Austria
{bacher,raidl}@ads.tuwien.ac.at

[2] Institute for Powertrains and Automotive Technology, Vienna University of Technology, Vienna, Austria
thorsten.krenek@ifa.tuwien.ac.at

Abstract. The fuel consumption of a simulation model of a real Hybrid Electric Vehicle is optimized on a standardized driving cycle using metaheuristics (PSO, ES, GA). Search space discretization and metamodels are considered for reducing the number of required, time-expensive simulations. Two hybrid metaheuristics for combining the discussed methods are presented. In experiments it is shown that the use of hybrid metaheuristics with discretization and metamodels can lower the number of required simulations without significant loss in solution quality.

Keywords: Hybrid Electric Vehicles, hybrid metaheuristics, search space discretization, metamodels

1 Introduction

For the automotive industry these days are game changing. Todays customer expectations and up-coming legal restrictions require the continuous development of vehicles with lower fuel consumptions and less emissions. Ongoing improvement of Internal Combustion Engines (ICEs) is one way to meet this challenge. But increasingly, hybridization of drives is seen as a promising alternative too — especially when the improvement of traditional drives reaches its limits.

The most popular form of hybridization today are different variants of hybrid electric powertrains built into Hybrid Electric Vehicles (HEVs). They integrate an ICE with one or more Electric Machines (EMs) and complement their fuel tank with electro-chemical energy storages (typically). The powertrain structure defines how different machines are able to interoperate; e.g., in a series hybrid the ICE and an EM generate electricity for storage while a second EM propels the vehicle, and in a parallel hybrid all machines are used for propulsion concurrently. Some HEVs, like the HEV considered in this paper, combine multiple concepts.

Operational modes specify the concrete interaction behaviour of a HEV's components. Specific driving situations constrain the allowed operational modes according to internal and environmental factors, like the State of Charge (SOC) of the battery, axle torques or requirements of driving dynamics. Performance and efficiency of HEVs depend strongly on the active mode chosen by the operation strategy for different driving situations. Optimizing the parameters of the operation strategy's decision criteria is therefore of utmost importance for the HEV's fuel efficiency. As specifics of HEVs and operation strategies are often vastly different, experience in manual adjustment is sparse. Therefore, automated algorithmic approaches pose promising alternatives.

In this paper we investigate how to efficiently adjust the (continuous) parameters of a new operation strategy for a "real" HEV[3] by applying metaheuristic optimization methods in combination with metamodels and simple, but effective post-processing techniques.

Our metaheuristic optimization techniques rely on the evaluation of a simulation model of the considered HEV to test different parameter configurations, i.e. candidate solutions, for the operation strategy. Parameter configurations are evaluated by simulating the HEV on a standardized driving cycle defining a required velocity for each second of the cycle. Fuel consumption and SOC of the battery are measured throughout the driving cycle and are used to compute an objective value for a candidate solution, as described in Section 7. Besides measuring these output values of the simulated HEV, the simulation is considered to be a black box function. No further information about the internal state and calculations of the simulation software is currently available to the optimization.

A major challenge for the metaheuristic optimization are the considerably long simulation times, limiting the overall number of simulations which can be performed in practice. For the model at hand the mean simulation time is about seven minutes per candidate solution.Some parameter configurations even lead to simulation times up to twenty minutes, hamstringing the optimization significantly. Further soft constraints in the context of practical application have to be considered. In our concrete case, license restrictions of the simulation software[4] constrain the maximum number of parallel simulations to only eight.

The main goal of this paper is therefore to explore ways to reduce the number of necessary simulations during optimization, while adhering to the solution quality of well known, unmodified reference algorithms. We intend to achieve this by training neural networks and ensemble methods as regression models of the simulations. We also consider how to evaluate the performance of the regression models under the aspect of their integration in the optimization process. Further we exploit a priori characteristics of the search space to limit its effective size. This is done in two ways. First by considering inter-parameter constraints in the form of inequalities and correspondingly repairing infeasible solutions. Second by exploiting the experience that the search space in similar HEV optimization problems often contain many plateaus or areas with shallow slopes. This obser-

[3] Due to Non-Disclosure Aggrements we are a not allowed to disclose the actual vehicle.
[4] We use the automotive simulation package GT Suite 7.2 from Gamma Technologies, Inc. http://www.gtisoft.com

vation allows to discretize the continuous domains of the problem to finite sets, without losing significant solution quality.

In Section 2 we give an overview of related work. Section 3 describes the used metaheuristics, and Section 4 details the search constraints. Section 5 introduces the employed regression models, while Section 6 integrates the mentioned modifications into a hybrid, phased metaheuristic. In Section 7 we describe the considered HEV model, its operation strategy and the objective function in more detail and present our experimental results. Finally Section 8 concludes this paper and gives an outline of possible future work.

2 Related Work

Concerning HEV optimization, in [13] a control strategy for a parallel HEV is optimized by Sequential Quadratic Programming (SQP). Optimization is applied on a response surface fit to data from a Design of Experiments (DOE). Their results indicate that building metamodels for simulations of a driving cycle can be done in principle.

Several metaheuristics are studied for a multi-objective HEV optimization in [10]. The methods try to find pareto-optimal solutions that minimize fuel consumption and multiple emission type like CO, Hydrocarbons, and NOx. In contrast, we focus solely on minimizing the fuel consumption.

The current paper builds upon previous work in [15], where a new and effective hybrid metaheuristic — PSAGADO — for fuel consumption optimization has been presented, and the first author's master thesis [3]. The hybrid metaheuristic combines a Particle Swarm Optimization, a Genetic Algorithm, and a Downhill-Simplex to improve the overall performance of the heuristic. Here, we pick up some of the open questions regarding approximation of fitness functions by surrogate models and search space properties.

An overview of basic techniques for simulation-based optimization is given in [1]. The paper describes the use of metamodels/surrogates as replacements for the original simulation, for reducing high computation times, e.g., by using the metamodel as filter for the simulation.

In [12] a framework for combining neural networks with an Evolution Strategy (ES), with Covariance Matrix Adaption (CMA), is proposed. Their strategies are evaluated on a set of standard test functions. Two approaches are described. The first approach, called "controlled individuals", falls back to the original fitness function for a specified fraction of the population, after all individuals have been evaluated with the approximative fitness function. Depending on how the reevaluated individuals are chosen, the approximative fitness function acts as a filter. The second approach is termed "controlled generations", where every few generations the whole population is evaluated with the original fitness function. Further the authors of [12] propose a method for managing the approximative fitness functions and an online training schedule for selecting and weighting new training data, based on the covariance matrix of the ES.

3 Used Metaheuristics

Optimization of the HEV's operation strategy is carried out by population-based metaheuristics. A priori, no information is available which metaheuristic performs best on the given problem, especially with the modifications to reduce running times in place. Based on our experience from [15], the following, well-known metaheuristics have been selected and tested in different usage scenarios, as described in Section 6.

Canonical Particle Swarm Optimization As first metaheuristic we consider a Canonical Particle Swarm Optimization (CPSO) in the "velocity update with inertia"-form as described in [17]. Originally proposed in [14], Particle Swarm Optimization (PSO) is a population-based metaheuristic where each particle, i.e. candidate solution, possesses a position $\boldsymbol{x_i}$ and a velocity $\boldsymbol{v_i}$ in the search space. The algorithm proceeds every iteration by evaluating the objective function at positions $\boldsymbol{x_i}$, updating the velocities according to

$$\boldsymbol{v_i} := \omega \boldsymbol{v_i} + \phi_1 \cdot \mathbf{rand}_{[0,1]} \cdot (\boldsymbol{x_{L_i}} - \boldsymbol{x_i}) + \phi_2 \cdot \mathbf{rand}_{[0,1]} \cdot (\boldsymbol{x_G} - \boldsymbol{x_i}) \tag{1}$$

where the constants ϕ_1 and ϕ_2 control the influence of their respective terms on the update and $\mathbf{rand}_{[0,1]}$ returns uniform random vectors in $[0,1]$. The velocity update in (1) decreases the velocity in the particle's previous direction, and reorients it towards its local best solution $\boldsymbol{x_{L_i}}$ and global best solution $\boldsymbol{x_G}$. The updated velocity is used to move each particle to $\boldsymbol{x_i} := \boldsymbol{x_i} + \boldsymbol{v_i}$. We modified the standard algorithm s.t. a particle repositions itself randomly if it has not improved after l iterations.

Evolution Strategy with Active Covariance Matrix Adaption An ES with Active Covariance Matrix Adaption (A-CMA)[5] has been implemented, as described in [11]. In each generation the ES samples λ individuals around a mutation center $\boldsymbol{x_w}$ according to a $\mathcal{N}(0, \boldsymbol{C})$ normal distribution. After evaluating the new population, the mutation center is moved towards the averaged position of the μ best individuals. The direction of the search, stored in $\boldsymbol{C}$, and the actual step size σ are decoupled and are updated seperately. The covariance matrix $\boldsymbol{C}$ is updated s.t. existing covariance information is reduced and augmented by information from the evolution path $\boldsymbol{p}_c$ and information about the current population $\boldsymbol{Z}$:

$$\boldsymbol{C} := (1 - c_{\text{cov}})\boldsymbol{C} + c_{\text{cov}}\boldsymbol{p}_c\boldsymbol{p}_c^T + \beta\boldsymbol{Z} \tag{2}$$

The evolution path $\boldsymbol{p}_c$ stores weighted information about the averaged best search directions of the μ best individuals, over all generations. The population term $\boldsymbol{Z}$ modifies the original covariance matrix by rotating and stretching $\boldsymbol{C}$ towards the μ best individuals and away from the μ worst individuals of the current generation. Constants c_{cov} and β weight the update terms. The update

[5] As the algorithm description is rather complex, we recap only the intention of the algorithm. For a complete description we refer to [11].

scheme is a variant of [9], with the intent to increase the adaption speed of $\boldsymbol{C}$, by actively penalizing bad mutation steps.

Sampling Genetic Algorithm Last but not least, a very simple Genetic Algorithm (GA) is implemented for sampling purposes, which are described in detail in Section 6. The GA controls a population of N individuals $\boldsymbol{x_i}$, $i = 1, \dots, N$. In each generation the population is evaluated and a new population is generated. The new population is formed by two mechanisms. The remaining $\lceil R \cdot N \rceil$ individuals, $0 \leq R \leq 1$, are randomly sampled from the search space to introduce variance into the population and to generate a diverse set of training data. The remaining $\lfloor (1 - R) \cdot N \rfloor$ individuals are created by recombination: By a tournament selection of size k, to select two parents $\boldsymbol{x_{p_1}}$, $\boldsymbol{x_{p_2}}$ are selected and an offspring is derived, by treating the selected parents as corners of a hypercube from within which a point is chosen uniformly at random.

4 Employed Post-Processing Techniques

The developed optimization framework allows to integrate post-processors for modifying solutions generated by the afore-mentioned metaheuristics. Modification of solutions is used to enforce search space constraints, like parameter domains or interdependencies between parameters. Interdependencies occur, for the HEV model at hand, in the form of simple inequalities like $\max_{i \in L} b_i \leq a \leq \min_{j \in U} b_j$ bounding some parameter a by lower bound parameters L and upper bound parameters U. Violation of inequalities renders a solution infeasible for simulation. The post-processor repairs infeasible solutions by assigning parameter a of the violated inequality a feasible random value. Feasible values can be easily determined by considering the parameter's domain and the inequalities to be satisfied.

Besides constraining parameters, a post-processor is used to discretize selected dimensions of the search space. Experience shows that search spaces for HEV models contain many plateaus with solutions of similar fitness. Evaluation of multiple solutions on such a plateau is costly and should therefore be avoided. Discretization supports this, as it limits the domains of the selected parameters to a fixed number of equidistant points. Parameter values are then mapped to the closest discretization point in the parameter's domain.

Currently this mapping is performed in a Lamarckian way i.e. the discretized solution actually replaces the original one. Temporary discretization i.e. mapping only for the evaluation process and retaining the original solution for the optimization process, is also an option which might be considered in future work.

In our implementation discretization points are always equidistantly distributed and their number is adapted during the optimization process, starting with very few points and progressing to a finer resolution of the search space. Different ways to refine the number of discretization points during optimization have been considered, like adapting them according to a linear function every iteration or adapting them only two or three times during optimization, in a step-wise fashion. Discretization makes it reasonable to store all computed solutions in a database, which acts as cache for objective function values. Preliminary experiments have shown that step-wise adaption is beneficial, as the number of cache hits is higher due to the fact that the positions of the discretization points is not modified every iteration.

5 Regression Models as Approximative Fitness Functions

Another way to decrease the number of simulations is to use regression models as approximative fitness functions, as done in [12] and [1]. Information about previous solutions is integrated into regression models to either act as a filter for bad solutions or to stretch the gathered information for several generations. In Section 6 we present hybrid metaheuristics for both approaches.

The range of functions which the regression models can fit to the gathered data is extremely important, as both over- and underfitting may have a negative impact on the optimization performance. As an ideal shape of the regression function cannot be known before the optimization is finished, a heuristic approach has to be considered. Therefore different regression techniques — so called ensemble methods [16] — based on Multilayer Perceptrons (MLPs) are evaluated beforehand and the "best" model is chosen for use.

A typical error function used for regression is the Sum-of-Squares Error:

$$\mathrm{SSE}(\varphi(.), \boldsymbol{X}) = \sum_{i=1}^{|X|} (y_i - \varphi(\boldsymbol{x_i}))^2 \tag{3}$$

The set $\boldsymbol{X}$ denotes all inputs $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_{|X|}}\}$ and target values y_i over which the SSE is computed. The trained regression function $\varphi(.) : \mathbb{R}^d \to \mathbb{R}$ receives the d input parameters of the HEV model as input.

For our purpose, however, SSE is not an appropriate error function for model selection. If the described metaheuristics are considered, it can be seen that the exact objective value of a solution is not required. Rather the order of candidate solutions is important to the metaheuristics' selection criteria.

Therefore Mean Total Order Deviation is proposed as error function:

$$\mathrm{MTOD}(\boldsymbol{X}) = \frac{1}{|\boldsymbol{X}|^2} \sum_{i=1}^{|X|} |\pi_o(\boldsymbol{x_i}) - \pi_\varphi(\boldsymbol{x_i})| \tag{4}$$

Where $\pi_o(\boldsymbol{x_i})$ denotes the rank of solution $\boldsymbol{x_i} \in \boldsymbol{X}$, when all solutions in $\boldsymbol{X}$ are ordered according to their real objective value y_i. Similar, $\pi_\varphi(\boldsymbol{x_i})$ denotes the rank of solution $\boldsymbol{x_i} \in \boldsymbol{X}$, when all solutions in $\boldsymbol{X}$ are ordered according to their predicted objective value $\varphi(\boldsymbol{x_i})$.
MTOD is designed to indicate the mean ordering shift of a solution within the evaluated solution set $\boldsymbol{X}$. Unfortunately, MTOD is inappropriate for classical MLP training methods due to lack of differentiability. Therefore we resort to

Sum-of-Squares Error (SSE) for training — under the assumption that SSE is a close approximation of MTOD in many cases — and to MTOD for selection.

MLPs form the base learners for all further methods, as they are able to express a wide range of functions depending on the number of hidden neurons, i.e., neurons between the input and output layer. The considered MLP architectures consist of one or two hidden layers, with different numbers of hidden neurons. Hidden layers use sigmoid activation functions, while the output layer uses a linear activation function. The neural networks are trained with a (modified) Levenberg-Marquardt algorithm [8] provided by the used neural network library ALGLIB[6]

To improve the generalization performance of the regression models, we consider different ensemble methods for combining multiple neural networks. Bagging [4] is the first evaluated approach, which trains multiple models on sets randomly sampled from the original training set $\boldsymbol{X}$. Bagging then averages the outputs of these models to reduce their variance and to improve generalization performance.

Second, (Stochastic) Gradient Boosting as described in [6] and [5] is adapted for using neural networks. In Gradient Boosting several regression models are used in succession. The first model $\varphi_0(.)$ is the mean over all target values y_i of $\boldsymbol{X}$. Then different neural networks are trained successively on the errors $(y_i - \varphi_j(\boldsymbol{x_i}))$ and a new candidate regression model is formed by

$$\varphi_j(\boldsymbol{x}) = \varphi_{j-1}(\boldsymbol{x}) + \rho\phi(\boldsymbol{x}), \quad j = 1, \ldots, E \tag{5}$$

where ρ weights the newly added model $\phi(.)$ and is determined by treating the SSE as a function of ρ only and setting its gradient to zero. At each step j the candidate with the lowest SSE over the training set $\boldsymbol{X}$ is chosen. The algorithm repeats these steps E-times to build the final model.

Third a partitioning approach similar to [7] is used. The training set $\boldsymbol{X}$ is clustered using the K-means++ algorithm [2] and different neural networks are trained for each cluster $\boldsymbol{C}_k \subset \boldsymbol{X}$. Selecting a neural network for a cluster uses a validation approach. Validation is done by splitting $\boldsymbol{C}_k$ into a training and a validation set and using the validation set for measuring the generalization performance. This is repeated several times and the model architecture with the lowest mean validation error is chosen.

Last a partial simulation and extrapolation approach is explored. The $[0, p]$-fraction of the driving cycle, with $p \in (0, 1)$, is simulated with the HEV model and its output values are recorded, i.e., the fuel consumption. The simulated parameter set $\boldsymbol{x}$ is then augmented with the recorded output values to form $\boldsymbol{x}^{[0,p]}$. The collected set $\boldsymbol{X}^{[0,p]}$ of all $\boldsymbol{x}_i^{[0,p]}$ is then used to train neural networks for predicting the $[p, 1]$-fraction of the driving cycle. Using the additional input data is expected to improve the prediction performance at the cost of higher computation times.

[6] For all (single) neural networks, (ALGLIB (www.alglib.net), Sergey Bochkanov) in version 3.6 is used; accessed: 2013-11-04

6 A Two-Phase Optimization Approach

We propose an approach for integrating the different metaheuristics, the described post-processing techniques, and regression models into new hybrid metaheuristics. Thereby the optimization is split into two phases.

The first phase is responsible for aggregating the initial training data. We decided to use the GA from Section 3 for this sampling purpose, as preliminary experiments showed that it produces a more diverse and larger set of training data than the CPSO or the ES. Further we use a step-wise adaption of the number of discretization points, where the times of adjustment correspond to the phases of the optimization algorithm. At the end of this first phase, the different regression models are evaluated. Their performance is measured by averaging the Mean Total Order Deviation (MTOD) values of a 10-fold crossvalidation. The regression model with the lowest validation error is chosen to be used in the optimization. In the second phase the resolution of the search space is enhanced by increasing the number of values per dimensions. Two different approaches for regression model integration have been tested.

A generational approach, similar to the one in [12], in combination with the CPSO is implemented. Optimization uses the regression models as main objective function and switches to the simulation model every m generations. Another integration method pairs the described ES with regression models as filter, as in [1], before passing the best to the simulation model. Far higher numbers of individuals can be sampled this way and only the best κ individuals are simulated each generation.

Further the regression models are updated every τ iterations to include the newly evaluated solutions. For the model update, a regression model with the same architecture as selected at the end of the first phase is used.

7 Experimental Results and Discussion

The considered HEV model possesses an ICE and two EMs — the “generator” and the “motor”. ICE and generator are situated on the same shaft, while the “motor” is coupled to the former with a planetary gear set. The HEV is able to operate in two different mode types: pure-electric (EV) and range-extended (ER), with two modes each. The first electric mode uses only the motor-EM for propulsion and is designed for low speeds. The second mode activates both EMs, disables the ICE and decouples it by opening a clutch. It is intended for higher velocities to lower the machine speeds, increasing the efficiency. The first range-extended mode — a so called series mode — targets lower velocities. The motor-EM propels the HEV, while the ICE/generator unit is decoupled from the driving shaft and is solely used to charge the battery. In difference, the second range-extended mode uses a power-split setup, where both ICE and motor-EM are propelling the HEV, but the power output of the ICE is split s.t. the generator-EM is used to charge the battery.

Twelve parameters of the HEV model are optimized. A switch between modes of the same type is performed above specific speeds $\mathit{speedup}_{\mathrm{EV/ER}}$ if the axle torque is below specified thresholds $torqueup_{\mathrm{EV/ER}}$, respectively. Switching from EV to ER is done at a defined speed $speedmin_{\mathrm{ER}}$, which has to be less than $speedup_{\mathrm{ER}}$. Further the allowed percental deviation from the initial SOC of the battery *socband* is optimized. It contributes to the decision of switching between different mode types. Also dependent on the *socband* is the power used to charge the battery, which is further determined by the interpolation between charge powers for low and high SOC deviations $chargepower_{\mathrm{L/H}}$. The required charge power is influenced by the decision if the current power demand of the motor-EM is to be covered by the generator, regulated by the *powerdemand* switch, too. Last $generatorpower_{\min}$ determines the minimal output of the generator-EM. Beside these parameters of the operation strategy, the teeth count of the ring and sun gear of the planetary gear set, connecting the engine- and driving shafts, is optimized too. Ring and sun gear require the constraint that their teeth difference has to be even.

The objective function which shall be minimized sums the HEV's fuel consumption in L/100km and a penalization term for SOC deviations between its initial and final state. The driving cycle we considered for optimization is the standardized EPA US-06[7] driving cycle. The penalization term estimates the fuel needed (measured in L) to charge the battery to the initial SOC state with the integrated ICE/generator unit, if the SOC is lower at the end of the cycle:

$$\frac{E_{\Delta\mathrm{SOC}}}{E_{\mathrm{fuel}}\rho_{\mathrm{fuel}}} \cdot \frac{100}{G_{\mathrm{eff}}} \cdot \frac{10^5}{s_{\mathrm{cycle}}} \tag{6}$$

where $E_{\Delta\mathrm{SOC}}$ is the energy equivalent to the SOC difference, $E_{\mathrm{fuel}} \approx 43\mathrm{MJ\,kg^{-1}}$ denotes the energy density of the fuel, $\rho_{\mathrm{fuel}} \approx 0.75\mathrm{kg\,L^{-1}}$ the fuel density, G_{eff} specifies the average generator efficiency estimated during simulation and s_{cycle} the length of the driving cycle in m. Penalization is included to favour SOC-balanced solutions, to be comparable to other HEVs. If the SOC is higher at the end than initially, then the penalization term becomes negative and even promotes the solution. The effect is limited by high energy losses during conversion.

A major challenge for proper experimental examination of the optimization problem at hand is the limited number of experiments which we could perform in a reasonable time. The number of simulations was restricted to 16 per iteration.

First, experiments with the unmodified CPSO and the unmodified ES, as described in Section 3, have been performed. Due to high computation times, we have only been able to run 3 experiments per unmodified algorithm. Best solutions for the experiments and information about the number of simulations are given in Table 2. The algorithms' parameters are given in Table 1. For the two-phase optimization the number of discretization points, for each dimension, changes from 6 in the first phase to 16 in the second.

The results for the unmodified metaheuristics clearly indicate that the CPSO dominates the ES. A closer analysis in [3] shows that the ES exhibits problems regarding the solution variance, which we attribute to the existence of plateaus

[7] See http://www.fueleconomy.gov/feg/fe_test_schedules.shtml for more information; accessed: 2013-11-11

Table 1: Algorithm parameters

Algorithm	**#Iterations**	**Parameters**
unmod. CPSO	200	#Part. = 16, $\omega = 0.7298$, $\phi_1 = \phi_2 = 1.496$, $l = 20$
unmod. A-CMA-ES	200	$\lambda = 16$, $\mu = 4$, init. as in [11]
CPSO-Phase I	65	#Part. = 16, $\omega = 0.7298$, $\phi_1 = \phi_2 = 2.0$, $l = 20$
GA-Phase I	65	$N = 16$, $R = 0.3$, $k = 2$
CPSO-Phase II	480	as Phase I; $m = 8$, $\tau = 15$
A-CMA-ES-Phase II	60	$\lambda = 100$, $\kappa = 16$, $\mu = 4$, $\tau = 15$

Table 2: Results for the unmodified optimization algorithms

Experiment	**Fuel c.**	**#Sim$_{\textbf{best}}$**
unmod. CPSO 1	5.86	3168
unmod. CPSO 2	5.83	1872
unmod. CPSO 3	5.83	2816

Experiment	**Fuel c.**	**#Sim$_{\textbf{best}}$**
unmod. A-CMA-ES 1	6.00	528
unmod. A-CMA-ES 2	6.00	2176
unmod. A-CMA-ES 3	6.00	2050

in the search space. The results for the first phase in Table 3 show that the CPSO outperforms the GA if the search space is discretized. Although, the GA generates more distinct solutions — 777 on average — and has therefore been selected as the first phase for all further algorithms.

Table 4 depicts the best results for each type of regression model. Extrapolation after simulating 75% of the driving cycle clearly outperforms the other models. Nevertheless, due to the still high simulation costs, Bagging has been selected for further experiments.

In the second phase the variants described in Section 6 are evaluated. Preliminary experiments have shown that the CPSO performs better if it is randomly initialized as opposed to starting from a set of good solutions. For the ES, on the other hand, it is beneficial to estimate the mutation center $\boldsymbol{x_w}$, the covariance matrix $\boldsymbol{C}$ and the step size σ from the 30 best solutions and 20 random solutions. The results for the second phase are given in Table 5. Both algorithms have been able to reach solutions below 5.9L/100km. The difference in solution quality compared to the unmodified algorithms is negligible as the HEV model itself exhibits errors at similar magnitude. Comparing the best cases of these algorithms to the unmodified CPSO's best case, then the two-phase optimization with CPSO reaches its best solution at 57.64% and the more consistent ES variant at 80.07% of the number of simulations. This implies that using discretization and metamodels reduces the required runtime significantly if compared to the unmodified reference algorithms. Further the performance of the ES improved considerably if compared to the unmodified variant. We attribute this to the local search behaviour of the used ES variant.

8 Conclusion and Future Work

We optimized the continuous parameters of an operation strategy for a HEV model based on a real HEV. We explored different ways to reduce the number of simulations required by the optimization by employing search space discretization and metamodels. Search space discretization has proven to be a valuable

Table 3: Results for Phase I

Experiment	Fuel c.	#Sim$_{\text{best}}$
CPSO 1	5.95	320
CPSO 2	6.11	211
CPSO 3	6.02	48
CPSO 4	5.95	105
CPSO 5	5.97	191
CPSO 6	5.96	266
CPSO 7	5.93	148
CPSO 8	6.07	210
CPSO 9	5.96	476
CPSO 10	5.91	87

Experiment	Fuel c.	#Sim$_{\text{best}}$
GA 1	6.01	136
GA 2	5.98	501
GA 3	6.04	121
GA 4	6.04	625
GA 5	6.00	727
GA 6	6.06	694
GA 7	5.99	631
GA 8	6.02	594
GA 9	6.00	433
GA 10	6.04	357

Table 4: Results for the best regression models per type

Model	Parameters	tMSE	vMSE	tMTOD	vMTOD
Neural network	$L = (12)$, $\omega = 1.0$	1.37	6.25	0.0746	0.0952
Gradient Boosting	$E = 16$, $S = 1.0$	0.16	5.49	0.0330	0.0984
Bagging	$E = 24$, $L = (24, 24)$, $\omega = 1.0$, $S = 1.25$	0.89	5.00	0.0339	0.0783
Partitioning	$O_p = 50\%$, $K = 5$	3.25	15.89	0.0749	0.1298
Partial simulation	$p = 0.25$, $L = (13)$, $\omega = 0.1$	0.01	0.02	0.0615	0.0810
Partial simulation	$p = 0.50$, $L = (13, 13)$, $\omega = 1.0$	0.00	0.01	0.0489	0.0703
Partial simulation	$p = 0.75$, $L = (78)$, $\omega = 1.0$	0.00	0.00	0.0351	0.0387

L	— the number of neurons per layer
ω	— the value for the weight decay parameter of the training algorithm
O_p	— percentage of closest solutions taken from each neighbouring cluster
E	— number of (internal) regression models
S	— size factor of the new training set sampled from the original training set $\boldsymbol{X}$

tXXX . . . training XXX, vXXX . . . validation, MSE = $\frac{1}{|X|}$ SSE

Table 5: Results for Phase II

Experiment	Fuel c.	#Sim$^{1}_{\text{best}}$
CPSO 1	5.93	189 (966)
CPSO 2	5.97	294 (1071)
CPSO 3	5.97	535 (1312)
CPSO 4	5.89	277 (1054)
CPSO 5	6.10	127 (904)
CPSO 6	5.97	428 (1205)
CPSO 7	6.01	46 (823)
CPSO 8	5.88	302 (1079)
CPSO 9	5.98	192 (969)
CPSO 10	5.93	149 (926)

Experiment	Fuel c.	#Sim$^{1}_{\text{best}}$
ES 1	5.87	893 (1670)
ES 2	5.92	803 (1580)
ES 3	5.89	334 (1111)
ES 4	5.90	508 (1285)
ES 5	5.87	722 (1499)

[1] Numbers in braces give the number of simulations combined with the average number of simulations (777) of the GA in phase I

tool in the presence of search spaces with plateaus. Bagging ensembles have been used to improve the generalization performance and mixing partial simulation and extrapolation yielded even better results. The presented hybrid two-phase metaheuristics have been able to reach similar results as the reference algorithms, while reducing the number of simulations to 57.64% and 80.07%, depending on the metaheuristic. In future work, advanced ways for incorporating metamodels should be studied, like integrating partial simulation into the optimization process, or using metamodels early in the optimization process. Different performance measures for metamodels, e.g., measures capturing the performance for solutions expected to be generated by the algorithms, should be considered.

References

1. April, J., Glover, F., Kelly, J.P., Laguna, M.: Practical introduction to simulation optimization. In: Simulation Conference, 2003. Proceedings of the 2003 Winter. vol. 1, pp. 71–78. IEEE Press (2003)
2. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2007)
3. Bacher, C.: Metaheuristic optimization of electro-hybrid powertrains using machine learning techniques. Master's thesis, Vienna University of Technology, Vienna, Austria (2013)
4. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
5. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics and Data Analysis 38, 367 – 378 (1999)
6. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189 – 1232 (2000)
7. Frosyniotis, D., Stafylopatis, A., Likas, A.: A divide-and-conquer method for multi-net classifiers. Pattern Analysis & Applications 6(1), 32–40 (2003)
8. Hagan, M., Menhaj, M.: Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks 5(6), 989 –993 (1994)
9. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9, 159195 (2001)
10. 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)
11. Jastrebski, G., Arnold, D.: Improving evolution strategies through active covariance matrix adaptation. In: IEEE Congress on Evolutionary Computation, 2006. pp. 2814–2821. IEEE Press (2006)
12. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. Evolutionary Computation, IEEE Transactions on 6(5), 481–494 (2002)
13. 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)
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, 1995. Proceedings. vol. 4, pp. 1942 – 1948 vol. 4. IEEE Press (1995)
15. Krenek, T., Ruthmair, M., Raidl, G.R., Planer, M.: Applying (hybrid) metaheuristics to fuel consumption optimization of hybrid electric vehicles. In: Chio et al., C. (ed.) Applications of Evolutionary Computation, Lecture Notes in Computer Science, vol. 7248, pp. 376–385. Springer Berlin Heidelberg (2012)
16. Mendes-Moreira, J.a., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: A survey. ACM Comput. Surv. 45(1), 10:1–10:40 (Dec 2012)
17. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1(1), 33–57 (2007)