2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Cascaded Evolutionary Multi-objective Optimization for Solving the Unbiased Universal Electric Motor Family Problem

Timo Friedrich and Stefan Menzel

Abstract—**For a successful business model the efficient development and design of a comprehensive product family plays a crucial part in many real world applications. A product family as it occurs, e.g., in the automotive domain consists of a product platform which covers the commonalities of product variants and the derived product variants. While product variants need to be fast and flexibly adjusted to market needs, from manufacturing and development point of view an underlying product platform with a large number of common parts is required to increase cost efficiency.**

For the design and evaluation of optimization methods for product family development, in the present paper the universal electric motor (UEM) family problem is considered, as it provides a fair trade-off between complexity and computational costs compared to real world application scenarios in the automotive domain. Since especially solving this problem without usage of pre-knowledge comes with high computational costs, a cascaded evolutionary multi-objective optimization based on NSGA-II with concatenation of product Pareto fronts is proposed in the present paper to efficiently reduce computational time. Besides providing sets of Pareto solutions to the unbiased UEM family problem the effects of considering solutions of prior platform optimizations as starting point for follow-up optimizations under changing requirements are evaluated.

I. Introduction

THE efficient specification of a successful product family which is comprised of a set of products sharing a common platform, i.e. some common parts, components, modules, or physical dimensions, is a key strategic issue for many companies [1]-[3]. The automotive industry derives different car types, i.e. products, based on one common physical platform to minimize development and manufacturing costs. The target of a platform optimization is to provide a high diversity within the product family while maintaining a large set of common parts and dimensions. Within the family, each product is derived from the platform with an individual instantiation of the non-platform parameters [1]. The number of shared parameters is called commonality.

Product families are distinguished into scale-based families, module-based families, and a combination of both [4]-[6]. Personal computers, e.g., are module-based families since they allow for a specific equipment line a combination of fixed parts like mainboards, power supply and casing, and individual parts like CPU or hard drive. Scale-based product families share a set of parameters instead of modules. A family of electric motors may share as platform the same values for the diameter of the motor and the cross sectional area of the winding wires but vary in the length of the produced motors to fulfill different needs like torque output. Of course, many real world applications as in the automotive industry combine a module- and scale-based strategy.

The economic benefits of product families are obvious. Components have to be developed only once and are utilized in multiple products. Similar products which share dimensions are manufactured on the same machines [3]. The challenging part of product family development is to define the optimal platform which requires a combined optimization of product and platform parameters. Hence, product family optimization usually yields a huge set of optimization parameters and additional flags for common parameters. By utilizing the commonality as an additional objective, multi-objective optimization approaches are applicable which result in Pareto sets of optimal solutions allowing the user to select the optimal platform and product parameters. With respect to commonality, two extremes exist which are called *total platform* with maximum commonality, i.e. all parameters are the same for all products, and *null platform* with no commonality, i.e. all parameters are optimal for each product. Of course the total platform provides minimal manufacturing costs but offers lowest performance for the product family, while the null platform provides the best performance for the product family but with highest costs [7], [8].

In the literature, one-stage or two-stage optimization approaches are proposed which either consider pre-defined platform parameters or collect platform parameters in the post processing of an optimization [6]. Because of the above mentioned optimization requirements of high parameter number and stochastic search for the global Pareto front, evolutionary computation is advantageous to offer solutions to this framework. Taking also into account that products and their specifications usually change over time, these optimizations may also be termed robust dynamic optimization or optimization under uncertainty. In the ideal case, a superior platform allows a future integration of products with presently unknown exact specifications.

In order to prepare the development, evaluation, and comparison of multi-objective approaches for optimization under uncertainty, we consider the Universal Electric Motor (UEM) family problem [6], [9], [10]-[12], since it provides a fair trade-off between complexity and computational cost. Since the model is based on physical equations which are well described in literature, the runtime of one motor evaluation is fast. Nevertheless, despite the fast single motor runtime, the generation of acceptable results for an unbiased optimization

Timo Friedrich and Stefan Menzel are with the Honda Research Institute Europe, Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany (email: {timo.friedrich, stefan.menzel}@honda-ri.de).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

in reasonable time is challenging for standard multi-objective optimization frameworks.

In the present paper, we propose a cascaded evolutionary multi-objective optimization which utilizes a practical approach for product Pareto front concatenation to solve the unbiased UEM problem in acceptable runtime. We provide the results of a three-objective UEM optimization with static environment and with a change of product specifications.

The paper is structured as follows. In section II, the underlying UEM scenario is described. Section III provides details on the implementation of the proposed optimization framework followed by a result section for the static environment. Effects of changes in product specifications are explained in section V. Concluding remarks are given in section VI.

II. Universal Electric Motor Optimization

The UEM optimization problem is widely used for product platform and product family optimization [6], [9], [11], [12]. It is based on the motor development of Black & Decker back in 1970 [1]. Due to regulation changes the company decided to reduce the amount of different electric motors for hand driven tools drastically. Furthermore, the motors were standardized to allow the production of all motor variants on a single production line resulting in a reduction of development and production costs. Inspired by the work of Black & Decker, Simpson et al. [9] developed a scale-based UEM family of 10 motors. In Fig. 1, the cross section of a UEM and the geometric parameterization are exemplary shown.

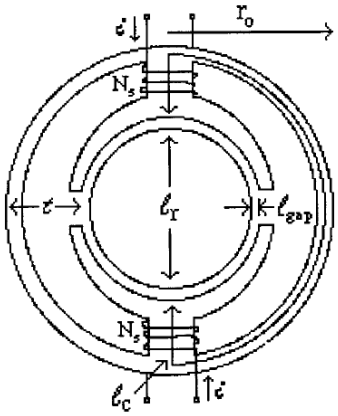

Fig. 1. Cross section of a physical model of UEM_i [9]

Each motor UEM_i with $i \in [1,10]$ consists of 8 parameters which are either common across the motors of the family or differ from motor to motor. The only strict difference between the motors is the torque requirement T_i. Detailed physical equations and parameter constraints are provided in [6] and [9]. The constrained optimization problem for a single motor UEM_i is given following [6]:

$$\text{Find:} \quad x_i = \{N_{c,i}, N_{s,i}, A_{wa.i}, A_{wf,i}, r_{o,i}, t_i, I_i, L_i\} \tag{1}$$

$$\text{Min:} \quad f(x_i) = \{Mass_i, -\eta_i\} \tag{2}$$

$$\text{s.t.:} \quad H_i(x_i) \leq 5000 \ \frac{Amp \cdot turns}{m} \tag{3}$$

$$\frac{r_{o,i}}{t_i} \geq 1 \tag{4}$$

$$Mass_i(x_i) \leq 2\ kg \tag{5}$$

$$\eta_i(x_i) \geq 15\% \tag{6}$$

$$P_i(x_i) = 300\ W \tag{7}$$

$$T_i(x_i) = \{0.05, 0.1, 0.125, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5\}\ Nm \tag{8}$$

The two objectives $Mass_i$ and efficiency η_i of each motor are calculated based on the following 8 parameters:

$N_{c,i}$, $N_{s,i}$	Number of windings for the armature and the stator [-]
$A_{wa,i}$, $A_{wf,i}$	Cross-sectional area of the armature and field wire [mm^2]
$r_{o,i}$, t_i	Stator dimensions [mm]
I_i	Electric current [A]
L_i	Length of motor [mm]

Thus, the parameter set of the 10 motors optimization problem consists of 80 parameters in total subject to 60 constraints. Nevertheless, the result of a complete product family optimization also requires to include the specification of the common set of parameters across motors. Hence, the performance so far calculated by mass and efficiency has to be extended by a fitness value which reflects the commonality between the products/motors inside the family. Commonality is handled, e.g., by a generalized commonality [6], [11], a product family penalty function (PFPF) [6], [7] or a simple count of platform variables per family [14]. Alternatively, the commonality is included as an additional objective which of course increases the dimensionality of the problem.

III. Implementation of the UEM optimization

A. NSGA-II with product family penalty function

In [6], Simpson et al. proposed and evaluated four principles to solve the UEM family problem which differ in when platform parameters are specified, *a priori* or *a posteriori*, and in the realization of the optimization framework, *two-stage* or *single stage* approach. If prior knowledge is accessible, the number and/or values of the platform variables can be specified before the optimization. Of course, an a priori definition of the search space accelerates the optimization on the one side but might fail to find the desired global optima. In contrast to prior specification, motor families are initialized with free platform parameters and proper solutions are determined during the post processing of the optimization which nevertheless might contain a reasonable large set of different solutions. Thus, the a posteriori process requires a higher computational effort which is needed to produce the set of solutions while it maintains the attractive benefit of a higher chance to find globally optimal solutions.

With respect to implementing an optimization framework, two-stage or single stage approaches are distinguished. In the two-stage approach the values of the platform variables are optimized first. The optimal numbers of the variables which are not common between motors are neglected during this step and are separately optimized afterwards. In contrast, the single stage approach optimizes platform and individual parameters at once.

Results show that the a posteriori, single stage approach is most promising in terms of generating highly varying Pareto optimal solutions [6]. The findings are based on an evolutionary optimization framework which implements the NSGA-II algorithm [13] for optimizing the motor families. In addition to the 8 design parameters of each motor the search parameter string is further extended with 8 binary switches to define whether a design variable is a platform or an individual variable across the family of motors [6].

Hence, the optimization problem for the family of motors UEM_i, $i \in [1,10]$ described in section II changes to:

$$\text{Find:} \quad x_i = \{N_{c,i}, N_{s,i}, A_{wa.i}, A_{wf,i}, r_{o,i}, t_i, I_i, L_i\} \tag{9}$$

$$x_{cc} = \{x_{cc,j}\} \tag{10}$$

x_{cc} is the vector of parameters $x_{cc,j}$, $j \in [0,7]$ for controlling the commonality of the motors. In order to include the commonality in the optimization, the fitness function is extended to

$$\text{Min:} \quad f(x) = \{\sum Mass_i, \sum -\eta_i, PFPF\} \tag{11}$$

where PFPF is a function to reflect commonality across the family. The function not only rewards exact same values for one search design parameter across the family but also the similarity of the values. In [6], more details on the PFPF set-up are given. Fig. 2 illustrates the representation for the NSGA-II algorithm which is based on 88 parameters. The binary values in the commonality controlling gene define whether a parameter is a platform variable or product variable. If a parameter $x_{cc,j}$ equals 1, parameter j maintains the same value across all products. If the parameter $x_{cc,j}$ equals 0, each motor may have a different value.

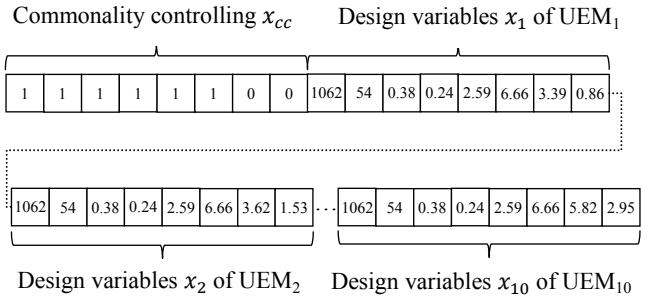

Fig. 2. NSGA-II representation as described in [6]

Following the optimization framework proposed in [6], our initial reimplementation showed high computational costs as well as convergence problems. As an example, a simplified optimization neglecting the commonality controlling genes, i.e. a null platform optimization with 80 search design variables, took about 1.5 billion evaluations to produce an acceptable Pareto front for the total mass and total efficiency, see Fig. 3.

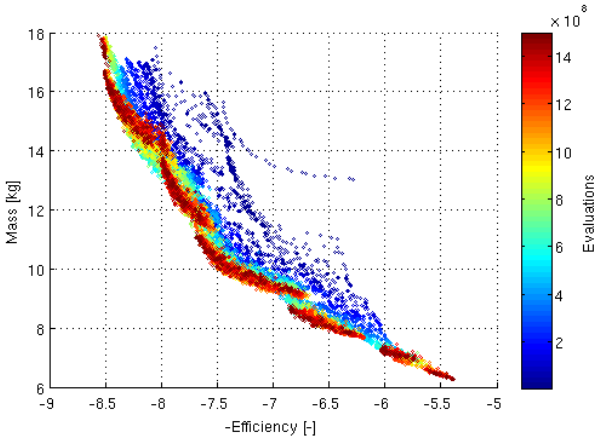

Fig. 3. Null platform optimization after 1.5 billion evaluations

Of course, efficient seeding of the optimization with feasible solutions as proposed in [10] would be advantageous with respect to optimization time but may provide pre-converged solutions and local optima. Hence, for solving the unbiased optimization problem in reasonable computational time, we developed a cascaded evolutionary multi-objective optimization approach. In section B, the framework hierarchy and parameter distribution is described while section C details the fitness calculation by concatenation of product Pareto fronts.

B. Cascaded multi-objective optimization framework

The cascaded optimization approach which finds the Pareto front without the need of seeding required an adaptation of the optimization setup which is illustrated in Fig. 4. The optimization is split into an outer platform/product family optimization, called outer optimization loop, and multiple inner optimization loops for each motor.

For the outer platform optimization the problem is now given as:

$$\text{Find:} \quad x_c = \{N_c, N_s, A_{wa}, A_{wf}, r_o, t, I, L\} \tag{12}$$

$$x_{cc} = \{x_{cc,j}\} \tag{13}$$

$$\text{Min:} \quad f(x) = \{\sum Mass_i, -\sum \eta_i, -\sum x_{cc,j}\}. \tag{14}$$

The two objectives inner optimization problem subject to the constraints (3) to (8) is given as:

$$\text{Find:} \quad x_i = \{N_{c,i}, N_{s,i}, A_{wa,i}, A_{wf,i}, r_{o,i}, t_i, I_i, L_i\} \tag{15}$$

$$\text{Min:} \quad f(x_i) = \{Mass_i, -\eta_i\} \tag{16}$$

The outer loop maintains the common platform parameters, i.e. the number (x_{cc}) and values (x_c) of platform parameters. The inner loops inherit the common platform parameters and optimize the non-common parameters of x_i for each of the 10 motors to result in 10 sets of Pareto solutions for mass and efficiency. Following the definitions

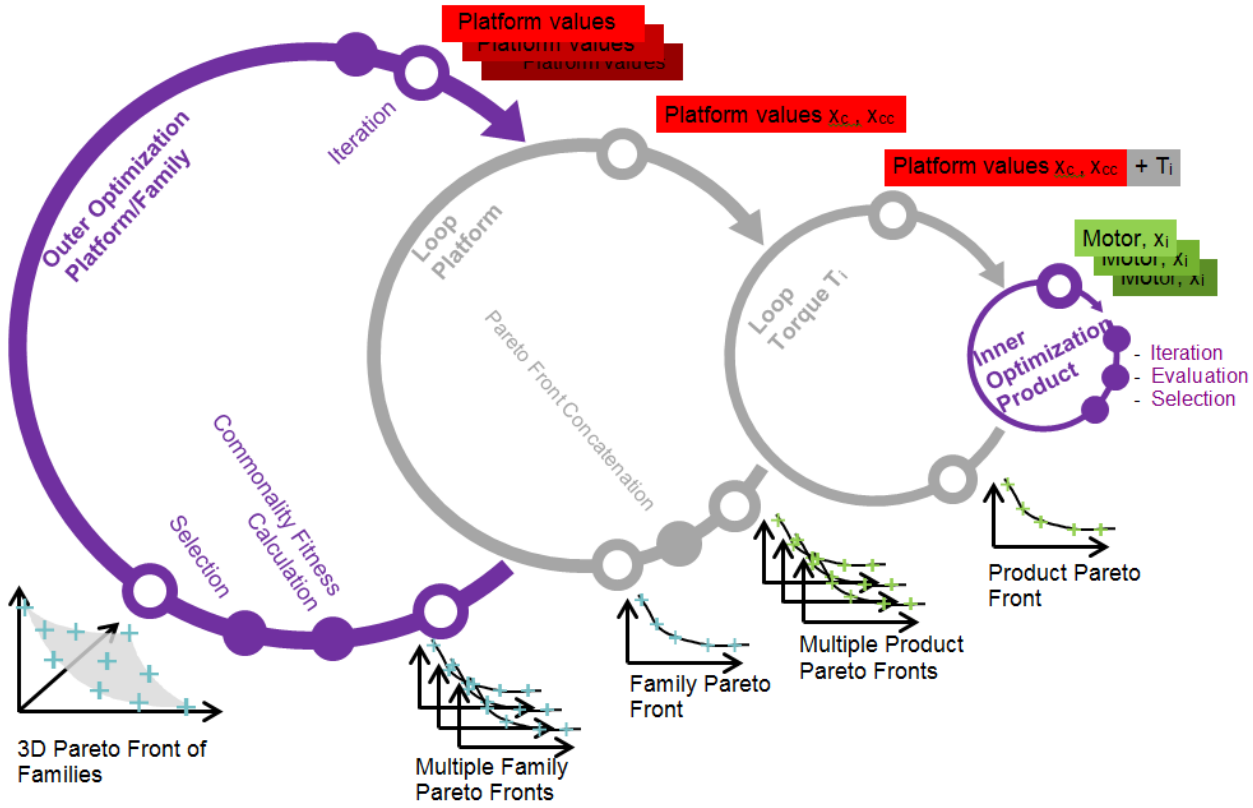

Fig. 4. Cascaded evolutionary multi-objective optimization for the UEM product family problem

given in [6], the complete cascaded optimization is a single-stage, a posteriori approach. The advantage of the cascaded optimization is the decoupling of the search parameters. The outer loop maintains 16 search parameters, while the inner loops have a maximum of 8 parameters for each single motor.

The commonality fitness values are calculated in the outer loop by a negative sum of the elements of the commonality controlling genes vector x_{cc} in a strict binary approach [14]. In difference to the product family penalty function in [6] this calculation rewards commonality across all products of a family.

Of course, depending on the commonality controlling gene in the outer loop the number of search parameters for the inner loop may become less than 8, i.e. reduced by all parameters j where $x_{cc,j} = 1$. In the extreme case of a total platform, i.e. where $x_{cc} = \{1,1,1,1,1,1,1,1\}$, the inner optimization is not executed at all, because all values are inherited from the corresponding platform, i.e. $x_i = x_c$.

In contrast to the benefit of fewer search parameters in the inner optimizations, a huge amount of independent inner optimizations has to be executed. Nevertheless, experiments have shown that this approach produces product families close to the optimal Pareto front quicker than an approach which optimizes all parameters for each motor at once. This is especially true for product families with a small amount of common platform variables. Product families with a platform coming close to the total platform need more time to converge to the optimal Pareto front, because the outer platform optimization has to find feasible platform parameters sets first. The adapted representation is shown in Fig. 5. In contrast to [11] a restricted commonality is applied.

Outer Optimization loop

Commonality controlling x_{cc} Platform design variables x_c

1	1	1	1	1	1	0	0	1062	54	0.38	0.24	2.59	6.66	3.39	0.86

Inner Optimization loops for each motor

Design variables x_1 of UEM_1:	1062	54	0.38	0.24	2.59	6.66	3.62	1.53
Design variables x_{10} of UEM_{10}:	1062	54	0.38	0.24	2.59	6.66	5.82	2.95

Fig. 5. Representation of the cascaded optimization approach

Outer and inner loops are based on the NSGA-II algorithm with standard operators for simulated binary crossover and polynomial mutation. Besides platform design variables the outer loop contains also parameters determining commonality, i.e. the commonality controlling gene.

The implementation of NSGA-II is based on the Shark 2.3 library [15]. Since the implementation supports continuous parameters only, the binary commonality switches and the integer motor parameters are stored as continuous parameters. The binary switch parameters determining commonality are corrected to 0 or 1 after each mutation. If a mutation of a commonality parameter is detected during optimization the commonality parameter is switched to its opposite value. Our

experiments showed that this method in combination with the polynomial mutation [16] leads to a reasonable switching rate. The integer values are always stored as floats and rounded before the motor calculations.

C. Product Pareto front concatenation

In section B, the first part of the proposed cascaded evolutionary multi-objective optimization approach has been described which splits the optimization in an outer and an inner loop and distributes the parameters accordingly. For a given platform, P_k, 10 inner loops are executed which optimize mass and efficiency for each torque specification T_i. Hence, the inner loops result in 10 Pareto fronts, one for each UEM_i, exemplary illustrated by blue lines in Fig. 6. To provide the result for P_k, these 10 Pareto fronts need to be accumulated into one Pareto front. Since the accumulation by a full combinatorial search is extremely expensive, we suggest two practical approaches for Pareto front concatenation to get a set of families which is close to the global Pareto front of all theoretic possible combinations.

In the first approach, the motors of each UEM_i Pareto front are sorted with respect to one objective. Due to the characteristics of Pareto optimal solutions the second objective is then sorted counter wise. The first motor family is calculated by adding the first motor of each front. The second family inherits each second motor and so on. This approach is limited to a two dimensional objective space and assumes an equal number of Pareto solutions on each front.

A second approach called fan approach chooses motors for each family based on a geometric method. First, two families are generated by combining the motors of each UEM_i Pareto front which either have the maximum or minimum performance in one objective. Based on these two families a reference point is computed, (-0.4, 1.9) in Fig. 6. This reference point allows the calculation of a fan like structure. For each ray, the closest motor of each Pareto front is chosen to be part of the family represented by the corresponding ray.

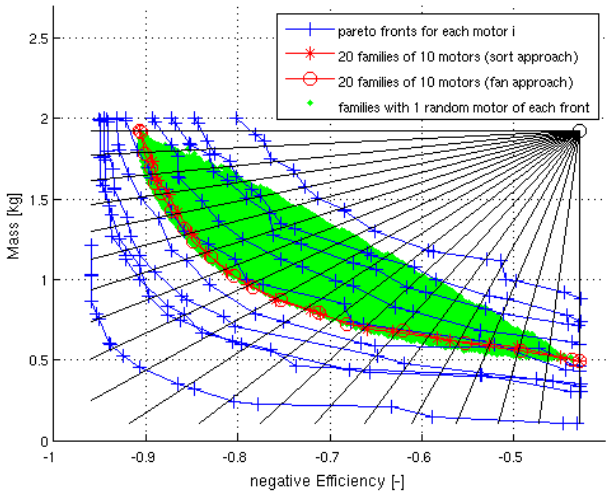

Fig. 6. Product Pareto front concatenation. The family values are averaged by the number of motors per family.

In Fig. 6, the results of approach 1 (∗) and approach 2 (o) are evaluated against a large set of solutions (●) which are based on combinations of 10 motors randomly chosen from each front. Both approaches are very close to the best front spanned by the randomly chosen motors. The fan approach delivers a slightly better distribution along the Pareto front especially in the lower right part where the distribution of solutions of the UEM_i is not very well balanced. Nevertheless, experiments have shown that the sort approach is sufficient for the present UEM optimization and has been chosen due to its lower computational costs.

Of course, after the step of product Pareto front concatenation for each platform P_k, which has been used to trigger the inner loops, a set of families is now available, i.e. the number of candidates which form the selection pool has increased. Thus, to avoid early dominance of a single platform the ratio between the total number of candidates and the candidates per platform has to be sufficiently high, from our experience greater than 10.

All sets of families which have been created by Pareto front concatenation for each P_k in the current generation are collected in a single solution set and have been assigned a third fitness value reflecting the commonality. This set is then used for the standard NSGA-II selection step in the outer loop. The comparison between the cascaded approach and the approach proposed in [6] shows that the individual motor parameters are decoupled from the platform parameters. The cascaded approach generates new parameter values in each iteration of the outer loop. This follows the idea that products are instantiations of the scaling variables and therefore should be reinitialized and optimized for each platform separately.

IV. Optimization Results

Based on the proposed optimization set-up described in section III we conducted a cascaded evolutionary three-objective optimization of the unbiased UEM family problem. The size of the parent population of the outer loop has been initialized with $\mu_{fam} = 80$ while the offspring population was chosen by $\lambda_{fam} = 180$ respectively. For the inner loop, we set $\mu_{prod} = 12$ and $\lambda_{prod} = 20$ respectively. Thus, the ratio between the total number of candidates and the candidates per platform is given by

$$\frac{\lambda_{fam}}{\mu_{prod}} = 15 \qquad (17)$$

Fig. 7 illustrates the results of the optimization run. It can be seen that solutions close to the Pareto front are found very fast. As long as less than 7 platform parameters are fixed for the family products, the inner optimizations are capable of producing reasonably good solutions using the remaining individual product parameters. In case of 6 fixed platform variables and 2 remaining individual variables, the 6 fixed variables already need to be optimized to reasonable values in the outer optimization. The total platform with 8 fixed platform variables of course never produces valid solutions because all UEM_i are identical and at least 9 motors violate the torque constraint T_i.

Fig. 7 shows the Pareto front (●) produced with 500 outer generations and 10000 generations for each single motor optimization. The results for a commonality of 7 and 8 are not

shown, because of the bad fitness values resulting from constraint violations as explained above.

One remarkable result of the optimization is the dominance of solutions with a commonality between 3 and 5. Even solutions with a higher commonality provide similar fitness values for mass and efficiency compared to solutions with a lower commonality. Thus, there is a high chance that they dominate solutions with a lower commonality. This is due to the fact that the corresponding single motor optimizations have to deal with less search variables if the commonality is high, while the number of generations is kept constant. To overcome this domination and achieve a well-balanced Pareto front, we suggest to couple the number of search variables of the inner optimizations to the number of generations in future optimizations. As a consequence, single motor optimizations with lower commonality are given more iterations for convergence. Thus, more solutions are found which are compatible during the Pareto sorting of the NSGA-II.

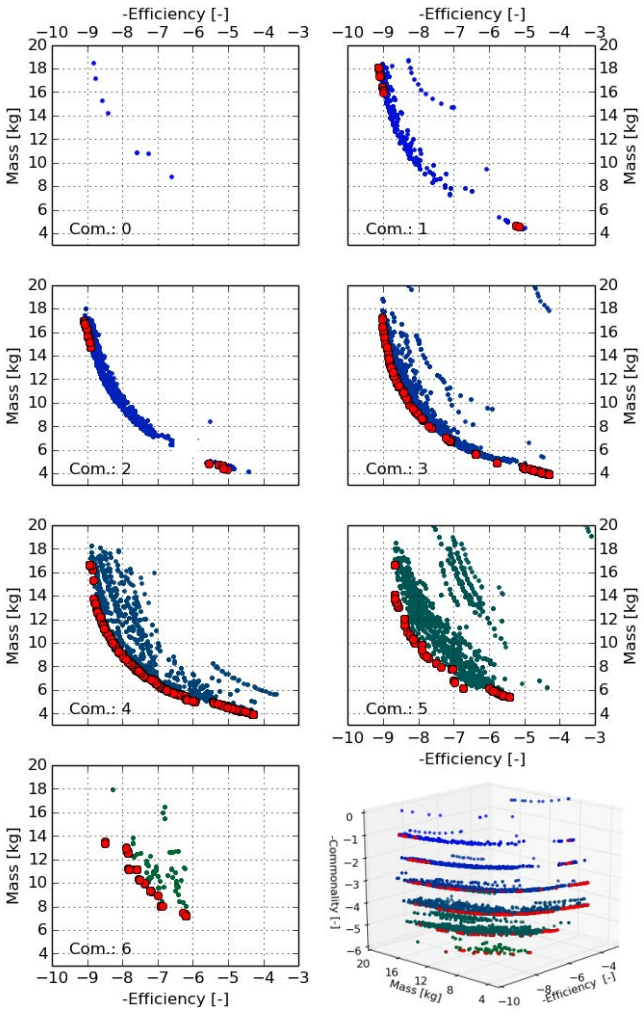

Fig. 7. Optimization results. Each dot marks one of approx.40.000 product family solutions. Red dots mark the Pareto dominant solutions. Com.: 0 to Com.: 6 show cross sections of the 3D Pareto front

The results shown in Fig. 7 are based on 1.5 billion evaluations. Compared to the initial set-up explained in section III the number is the same, but now the complete commonality part is also added and thus the complexity is higher and the search space much larger. Besides that, the optimization has reached the optimal Pareto front a lot faster and used the evaluations to spread along the front. The total computation time using 5 CPUs with 2.4GHz is about 8 hours if the families consist of all 10 motors.

V. Variation of Product Specifications

In real world product family development the complete product portfolio is usually unknown at the time of platform optimization. The introduction of newly designed products or products with updated specifications requires a re-design of the underlying platform. As a solution to this dynamic process, the platform optimization can either be restarted from scratch or using the solutions of a prior optimization. The expected benefit of former results usage is a reduced optimization time which is especially important if computational costly simulations are required to determine the product performance as, e.g., finite element analysis or computation fluid dynamic simulations for structural or aerodynamic design problems. On the contrary, the chances of pre-convergence to a local optimum increase especially if the variation of the new/updated product specification is large. To evaluate both effects in the UEM problem, experiments using the cascaded optimization framework have been carried out to study the difference in convergence behavior for optimizations from scratch and optimizations based on prior results.

A. Problem set-up

The original UEM problem comprises 10 single motors distinguished by the torque requirements. These torque values T_i are

$$T_{0-9} = \begin{Bmatrix} 0.05, 0.1, 0.125, 0.15, 0.2, 0.25, \\ 0.3, 0.35, 0.4, 0.5 \end{Bmatrix} \text{Nm}. \quad (18)$$

We extended the framework to allow an optimization of a subset of these motors. The subset of motors is specified by a vector containing the requested torque values. For example, if only UEM_1, UEM_2 and UEM_3 are considered, the torque requirements vector reduces to

$$T_{1-3} = \{0.1, 0.125, 0.15\} \text{Nm}. \quad (19)$$

For simplicity, we call this set-up UEM123 for which an initial set of Pareto solutions is computed using the cascaded optimization approach for 500 iterations. This set $UEM123_{500}$ defines the initial solution which is used to initialize follow-up optimizations with modified specifications. The idea behind the follow-up optimizations is to extend the initial motor specification UEM123 by two additional motors D and E. By changing the number of products the specifications change because the platforms $UEM123_{500}$ which are optimal for UEM123 need to be adapted to satisfy the additional constraints. The modified specifications are called UEM123+DE or short 123+DE. So an experiment with additional motors 6 and 7 is called 123+67. As explained above, each additional experiment is carried out in two ways.

On the one hand we optimized from scratch for 1000 generations and called these experiments with the extension _1000. On the other hand we utilized an initialization with the UEM123 platform solution set $UEM123_{500}$ and optimized for another 500 iterations to have the same overall generation number. These experiments are tagged with _restart. All optimization runs were executed 50 times with different random seeds to allow statistical analysis. The statistical analysis has been carried out using a t-test at a 5% significance level.

To evaluate the optimization progress and the convergence behavior, the S-Metric-Hypervolume [17] is calculated for each generation. The reference point is set manually to 100 for each objective and kept constant for all setups. It is important to note that the absolute values of the hypervolumes are only comparable between set-ups consisting of the same motors, i.e. 123+45_1000 is only comparable to 123+45_restart, since the sum of efficiency and sum of mass of course differs depending on the chosen specification.

We suppose that a restart of the optimization with former optimization results is only beneficial if the new specifications are in a similar range to the initial UEM123 setup. The similarity is estimated based on the torque requirements of each motor. If a similar motor is added to the family, we assume a restart with prior results is faster and produces better solutions within fewer generations. With decreasing similarity an optimization from scratch should become superior because if the added motors are too different from the initial setup, the restarted optimization will more likely converge to local optima. To increase the difference of the specifications we added two motors UEM_{10} and UEM_{11} with torque requirements T_{10}=0.75 and T_{11}=1.0. Thus:

$$T_{0-11} = \begin{Bmatrix} 0.05, 0.1, 0.125, 0.15, 0.2, 0.25, \\ 0.3, 0.35, 0.4, 0.5, 0.75, 1.0 \end{Bmatrix} \text{Nm.} \quad (20)$$

B. *Experimental results*

Fig. 8 provides an overview on the average performance over 50 runs of the different optimizations. Colored lines mark optimization runs from scratch while dashed lines mark optimization runs with $UEM123_{500}$ initialized solutions. We added the dashed lines in two ways. One set starts at generation 0 to compare the initial optimization speed between both runs and one starts at generation 500 to compare the final values at generation 1000.

The plot in Fig. 8 shows that the final results of runs 123+45 to 123+89 are nearly identical for both set-ups. Optimization runs 123+45_restart up to 123+78_restart show a higher initial optimization speed since the initial solutions $UEM123_{500}$ are already good initial estimations. With increasing difference of specifications, i.e. higher torque requirements, the initialization has no advantage. 123+89_restart has a similar performance as 123+89_1000, while 123+910_restart and 123+1011_restart even perform poorly. Both runs seem to be stuck in local optima and lose drastically compared to the runs carried out from scratch.

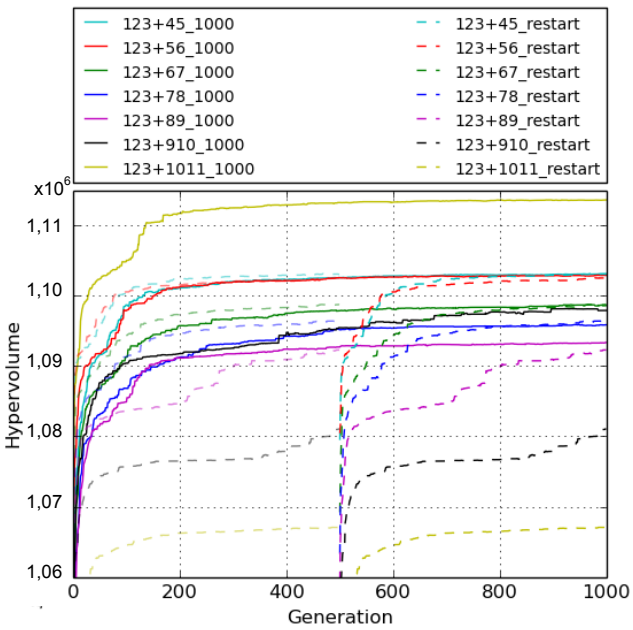

Fig. 8. Experimental results of the cascaded multi-objective optimization with and without pre-initialized solutions

To improve the readability of the results, we normalized the averaged hypervolumes $HV_{ABC+DE\_1000,ave,generation}$ and $HV_{ABC+DE\_restart,ave,generation}$ of all runs. Thus, at first for each optimization set-up which had been started from scratch the hypervolumes at generation 0 have been averaged over all 50 runs, and called $HV_{ABC+DE\_1000,ave,0}$, i.e. $HV_{123+45\_1000,ave,0}$ for run 123+45. The idea is that these hypervolume values indicate a baseline quality which the Pareto solutions already provide just with randomly chosen initial parameters. For each set-up the averaged hypervolumes are now adjusted for each generation by

$$HV_{ABC+DE\_1000,adj,generation} = HV_{ABC+DE\_1000,ave,generation} - HV_{ABC+DE\_1000,ave,0} \quad (21)$$

$$HV_{ABC+DE\_restart,adj,generation} = HV_{ABC+DE\_restart,ave,generation} - HV_{ABC+DE\_1000,ave,0} \quad (22)$$

Finally the performance is calculated by

$$HV_{ABC+DE,generation} = HV_{ABC+DE\_restart,adj,generation} / HV_{ABC+DE\_1000,adj,generation} \quad (23)$$

TABLE I
$HV_{ABC+DE,GENERATION}$

123+45	1,06	1,03	1,02	1,02	1,02	1,02	1,02	1,02	1,01	1,01
123+56	1,09	1,05	1,04	1,03	1,02	1,01	1,01	1,01	1,01	1,00
123+67	1,07	1,03	1,02	1,01	1,01	1,01	1,01	1,01	1,01	1,01
123+78	1,09	1,05	1,05	1,04	1,03	1,03	1,03	1,03	1,03	1,02
123+89	1,00	1,00	0,99	0,99	0,99	1,00	1,00	1,00	1,00	1,00
123+910	0,84	0,84	0,84	0,84	0,84	0,83	0,83	0,83	0,83	0,84
123+1011	-0,36	-0,23	-0,18	-0,15	-0,13	-0,11	-0,10	-0,08	-0,07	-0,07
ABC+DE / Generation	50	100	150	200	250	300	350	400	450	500

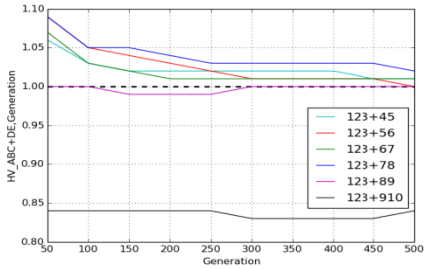

Fig. 9. Optimization comparison of set-ups 123+45 to 123+910

The results are given in Table I and visualized in Fig. 9. Statistical significant results are marked with a grey background. Table I underlines the results stated above. In the beginning of optimization set-ups 123+45 to 123+78 the initialization with the $UEM123_{500}$ dataset provides an advantage of up to 9 %. From generation 200 the hypervolume of the solution set is very similar and the advantage of pre-initialization vanishes. For run 123+89 both set-ups perform equally well. Initializing run 123+910 reduces the performance by 20 % which is even worse for 123+1011. In this run the pre-initialized solutions are worse compared to just randomly chosen values. It can also be seen that both runs were stuck in local optima since both runs can not catch up to the optimization runs carried out from scratch.

VI. Conclusion

Real world applications from the automotive industry, which include structural or aerodynamic simulations, usually come with a high simulation cost which makes method development difficult. In particular, platform optimization requires a large number of function evaluations if we strive for global optimal solutions. The UEM problem promises to be a fair trade-off between problem complexity and computational runtime. Because of its popularity in the platform optimization domain, it is an adequate choice as benchmark problem which allows the study of different optimization approaches. We also think that it is an interesting problem from the point of view of dynamic optimization by exchanging the motor specifications over time.

For solving the unbiased UEM problem in reasonable computational time, in the present paper a cascaded evolutionary multi-objective optimization is proposed. Main features of the framework are the hierarchical optimization loops for platform parameter optimization and motor optimization based on NSGA-II as well as the concatenation of Pareto fronts. Due to the strict separation of the platform and product parameters, the developed algorithm is able to find product families close to the optimal Pareto front quickly. An evolutionary three-objective optimization including commonality as objective shows the good performance of the proposed framework.

Based on the proposed optimization method, the effects of a change in system requirements have been evaluated to verify the applicability of the framework. Initial experiments underline the intuitive hypothesis that the advantage of pre-initialization with optimal prior solution sets vanishes with increasing variations in product specifications. Nevertheless, changing product requirements over the lifetime of a platform play of course a major role in real world applications since the efficiency of a platform drastically increases if it provides the capacity to be evolvable within dynamic and uncertain environments.

Acknowledgment

The authors cordially thank Prof. Dr. Timothy W. Simpson, Department of Industrial and Manufacturing Engineering, Pennsylvania State University for his kind support in setting up the universal electric motor scenario.

References

[1] M. H. Meyer and A. P. Lehnerd, “The power of product platforms: building value and cost leadership,” New York, NY, 1997.
[2] M. E. McGrath, “Product strategy for high-technology companies: how to achieve growth, competitive advantage, and increased profits,” Irwin Professional Pub., 1995.
[3] D. Robertson and K. Ulrich, “Planning for Product Platlorms,” Sloan Management Review, 1993.
[4] T. W. Simpson, Z. Siddique, and J. R. Jiao, “Platform-Based product family development,” Springer-Verlag New York Inc., 2010.
[5] S. Chowdhury, A. Messac, and R. A. Khire, “Comprehensive product platform planning (CP3) framework,” Journal of Mechanical Design, vol. 133, 2011.
[6] T. W. Simpson, Zahed Siddique, and J. (Roger) Jiao,” Product Platform and Product Family Design Methods and Applications,” Gardners Books, 2010.
[7] T. W. Simpson and B. S. D’Souza, “Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm,” Concurrent Engineering, vol. 12, no. 2, pp. 119–129, 2004.
[8] R. Fellini, M. Kokkolaras, P. Papalambros, and A. Perez-Duarte, “Platform Selection Under Performance Bounds in Optimal Design of Product Families,” Journal of Mechanical Design, vol. 127, no. 4, p. 524, 2005.
[9] T. W. Simpson, J. R. Maier, and F. Mistree, “Product platform design: method and application,” Research in engineering Design, vol. 13, no. 1, pp. 2–22, 2001.
[10] S. V. Akundi, T. W. Simpson, and P. M. Reed, “Multi-objective design optimization for product platform and product family design using genetic algorithms,” 2005.
[11] A. Khajavirad, J. J. Michalek, and T. W. Simpson, “An efficient decomposed multiobjective genetic algorithm for solving the joint product platform selection and product family design problem with generalized commonality,” Structural and Multidisciplinary Optimization, vol. 39, no. 2, pp. 187–201, Nov. 2008.
[12] D. Kumar, W. Chen, and T. W. Simpson, “A market-driven approach to product family design,” International Journal of Production Research, vol. 47, no. 1, pp. 71–104, 2009.
[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” Evolutionary Computation, IEEE Transactions on, vol. 6, no. 2, pp. 182–197, 2002.
[14] A. Khajavirad and J. J. Michalek, “A single-stage gradient-based approach for solving the joint product family platform selection and design problem using decomposition,” in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, p. 17, 2007.
[15] Igel, Christian, Verena Heidrich-Meisner, and Tobias Glasmachers. "Shark." The Journal of Machine Learning Research 9, pp. 993-996, 2008.
[16] K. Deb and D. Deb, “Analyzing Mutation Schemes for Real-Parameter Genetic Algorithms,” KanGAL Report Number 2012016.
[17] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima, “Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions,” in Proceedings of the 12th annual conference on Genetic and evolutionary computation, pp. 527–534, 2010.