A Hybrid Soft Computing Approach for Optimizing Design Parameters of Electrical Drives *

Alexandru-Ciprian Zăvoianu[1,3], Gerd Bramerdorfer[2,3], Edwin Lughofer[1], Siegfried Silber[2,3], Wolfgang Amrhein[2,3], Erich Peter Klement[1,3]

[1] Department of Knowledge-based Mathematical Systems/Fuzzy Logic Laboratorium Linz-Hagenberg, Johannes Kepler University of Linz, Austria
[2] Institute for Electrical Drives and Power Electronics, Johannes Kepler University of Linz, Austria
[3] ACCM, Austrian Center of Competence in Mechatronics, Linz, Austria

Abstract. In this paper, we are applying a hybrid soft computing approach for optimizing the performance of electrical drives where many degrees of freedom are allowed in the variation of design parameters. The hybrid nature of our approach originates from the application of multi-objective evolutionary algorithms (MOEAs) to solve the complex optimization problems combined with the integration of non-linear mappings between design and target parameters. These mappings are based on artificial neural networks (ANNs) and they are used for the fitness evaluation of individuals (design parameter vectors). The mappings substitute very time-intensive finite element simulations during a large part of the optimization run. Empirical results show that this approach finally reduces the computation time for single runs *from a few days to several hours* while achieving Pareto fronts with a similar high quality.
Keywords: hybrid soft computing methods, multi-objective genetic algorithms, feed-forward artificial neural networks, electrical drives

1 Introduction

1.1 Motivation

Today about 70% of the total consumption of electrical energy in industry and about 40% of global electricity is used for electric drives. In [3] it is stated that about 200TWh are actually needless wasted energy in the European Union and could be saved by increasing the efficiency of electrical drives. For that reason, a regulation was concluded in 2009 by the European Union forcing a gradual

* This work was conducted in the realm of the research program at the Austrian Center of Competence in Mechatronics (ACCM), which is a part of the COMET K2 program of the Austrian government. The work-related projects are kindly supported by the Austrian government, the Upper Austrian government and the Johannes Kepler University Linz. The authors thank all involved partners for their support. This publication reflects only the authors' views.

increase of the energy efficiency of electrical drives [8]. However, manufacturers of electrical machines need to take more than just the efficiency into account in order to be competitive on the global market. To be able to successfully compete, the electrical drives should be fault-tolerant, should offer easy controllable operational characteristics and compact dimensions, and, last but not least, should have a very competitive price. As such, the need to simultaneously optimize electrical drives with regards to several objectives is self evident.

In order to evaluate (predict) the operational behavior of an electrical machine for a concrete design parameter setting, very time-intensive finite element (FE) simulations (on partial differential equations) need to be performed due to the nonlinear behavior of the used materials. Even the state-of-the-art optimization algorithms, typically require the evaluation of thousands of such designs in order to produce high quality results. By using computer clusters, several evaluations can be performed in parallel, significantly speeding up the optimization procedure. Still, the need to evaluate every single design by means of FE-simulations remains a major drawback. Because of this dependency on FE-simulations, optimization runs can take several days to complete, even when distributing computations over a computer cluster. In this paper we describe an effective hybrid soft computing approach which is able to greatly reduce the optimization run-time by relaxing the dependency on FE-simulations.

1.2 Problem Statement and State-of-the-art

The design of an electrical machine usually comprises at least the optimization of the geometric dimensions of a pre-selected topology. Furthermore, because of volatility in the global raw material market, companies tend to investigate the quality of the electrical drive design with regard to different construction materials. Formally, the three multi-objective optimization problems (MOOPs) that we use in this study, can be defined as:

$$\min\left(f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})\right)$$

where

$$f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x}) \quad \text{and} \quad \mathbf{x}^T = \left[\, x_1\ x_2\ \ldots\ x_n \,\right] \tag{1}$$

are the objectives and the design parameter vector (e.g. geometric dimensions, material properties, etc.). Additionally, hard constraints like $\mathbf{g}(\mathbf{x})$ can be specified in order to make sure that the drive exhibits a valid operational behavior (e.g. the torque ripple is upper bound). Such constraints are also used for invalidating designs with a very high price.

$$\mathbf{g}(\mathbf{x}) \leq 0 \in \mathbb{R}^m \tag{2}$$

Generally, in order to characterize the solution of MOOPs it is helpful to first explain the notion of *Pareto dominance* [4]: given a set of objectives, a solution A is said to Pareto dominate another solution B if A is not inferior to B with regards to any objectives and there is at least one objective for which A is better than B.

In MOOPs, a single optimal solution is extremely difficult to find and, in many cases, such a solution does not even exist. The result of an optimization process for a MOOP is usually a set of Pareto-optimal solutions named the *Pareto front* [4]. The ideal result of the optimization is an evenly spread Pareto front which is as close as possible to the *true Pareto front* of the problem, i.e. the set of all non-dominated solutions in the search space.

A widespread classical approach to solve MOOPs is to combine the multiple objectives into a single aggregating objective (e.g. a linear combination of the initial objectives) and then use a single-objective optimization technique to find a solution [4]. The downside of this method is that when trying to find multiple (Pareto-optimal) solutions, several independent runs are needed with the hopes that each run would yield a different solution. Unsurprisingly, population based optimization methods from the field of soft computing like particle swarm optimization [19], ant colony optimization [2] and especially evolutionary algorithms [4] perform much better in the context of MOOPs. This is because of the inherent trait of all these methods to step-wise improve sets (populations) of solutions. As such, various extensions aimed at making the contained populations store and efficiently explore Pareto fronts have enabled these types of algorithms to efficiently find multiple Pareto-optimal solutions for MOOPs in one single run.

During the last decade, the use of soft computing methods like genetic algorithms [15] and particle swarm optimization [17] has also become state-of-the-art in the design process of electrical machines and associated electronics. A detailed review regarding the performance of these and other optimization methods used in the field of electrical machine design can be found in [6].

1.3 Our Approach

Recent hybrid learning and optimization methods have proven to be very efficient at tackling complex real-world problems [1]. In order to obtain a fast and efficient optimization framework, we also resort to a hybrid approach that combines two well known methods from the field of soft computing [10]: 1.) evolutionary algorithms - in particular, we consider the specialized class of multiple-objective evolutionary algorithms (MOEAs) [4] and 2.) artificial neural networks (ANNs) - in particular, network models belonging to the multilayer perceptron (MLP) paradigm [12].

The use of MOEAs to efficiently explore the search space and converge (in relatively few generations) to an accurate Pareto can be considered a somewhat conventional approach and, throughout this paper, we shall refer to this basic application of MOEAs in the design process as **ConvOpt**. The fitness evaluation function in ConvOpt is very time-intensive as it dependents on FE-simulations. As such, despite the rather fast convergence in terms of generations that need to be computed, the entire optimization process is quite slow.

Our idea for reducing the evaluation time of individuals is to substitute the time-intensive evaluation function based on FE simulations with a very fast approximation function based on highly accurate regression models, i.e. *mappings* between the motor design parameters and the target values which should

be estimated. These targes are actually optimization specific objectives (1) and constraints (2). As the mappings are specific for each optimization scenario, they need to be *constructed on-the-fly, at each run of the evolutionary algorithm*. This means that only individuals from the first few generations will be evaluated with the time-intensive FE-based evaluation function in order to construct a training set for the target mappings. For the remaining generations, the mappings will substitute the FE simulation as the basis of the fitness function. This hybrid approach (**HybridOpt**) will yield a significant reduction in computation time, from *a few days to several hours*, as we verified during empirical tests.

Initially, for constructing the mappings, we considered easy and fast to train linear models. However, these models turned out not to be sufficiently accurate. Thus, we exploited non-linear techniques and finally decided to use ANNs because of the following reasons: 1.) they possess the universal approximation capability [14], 2.) they are known to perform very well on noisy data [16] (in our application, noise may arise due to slight environmental variations during the optimization) and 3.) they have already been successfully applied in evolutionary computation for designing mappings for *surrogate functions* on several instances [13]. In order to elicit ANN models of optimal complexity (# of neurons), we rely on a best parameter grid search and we designed a new selection method (Section 2.2) that successfully balances model accuracy and sensitivity on the one side and model complexity on the other.

2 Optimization Procedure

2.1 Multi-Objective Evolutionary Algorithms

Two of the mainstream evolutionary algorithms used for solving MOOPs are NSGA-II [5] and SPEA2 [21]. Because of the similar design principles and similar performance of the two algorithms with regards to our test scenarios, we decided to apply NSGA-II for the purpose of this research. The individuals with which the algorithm operates are represented by real valued design parameter vectors. The size of the design vector ranges from six to ten. For all considered objectives, minimization towards a value of 0 is the preferred option (as explained in Section 1.2).

NSGA-II stores at each generation t two distinct populations of the same size n, a parent population $P(t)$ and an offspring population $O(t)$. Population $P(t+1)$ is obtained by selecting the best n individuals from the combined populations of the previous generation, i.e., from $C(t) = P(t) \cup O(t)$. The fitness of an individual is assessed by using two metrics. The first metric is a classification of the individuals in the population into non-dominated fronts. The first front $F_1(t)$ is the highest level Pareto front and contains the Pareto optimal set from $C(t)$. The subsequent lower-level fronts $F_j(t), j > 1$ are obtained by removing higher level Pareto fronts from the population and extracting the Pareto optimal set from the remaining individuals, i.e., $F_j(t), j > 1$ contains the Pareto optimal set from $C(t) \setminus \bigcup_{k=1}^{j-1} F_k(t)$. Individuals in a higher-level front $F_j(t)$ are ranked

as having a higher fitness than individuals in a lower-level front $F_{j+1}(t)$. NSGA-II uses a second metric, the crowding distance, in order to rank the quality of individuals from the same front. The crowding distance associated to a certain individual is an indicator of how dense the non-dominated front is around that individual. Population $P(t+1)$ is obtained by adding individuals from the higher non-dominated fronts, starting with $F_1(t)$. If a front is too large to be added completely, ties are broken in favor of the individuals that have the higher crowding distance — are located in a less crowded region. Population $O(t+1)$ is obtained from population $P(t+1)$ by using binary tournament selection, simulated binary crossover and polynomial mutation. At every generation t, the individuals that do not satisfy the imposed constraints (2) are removed from $O(t)$ before constructing $C(t)$.

2.2 Fitness Function Calculation using ANNs

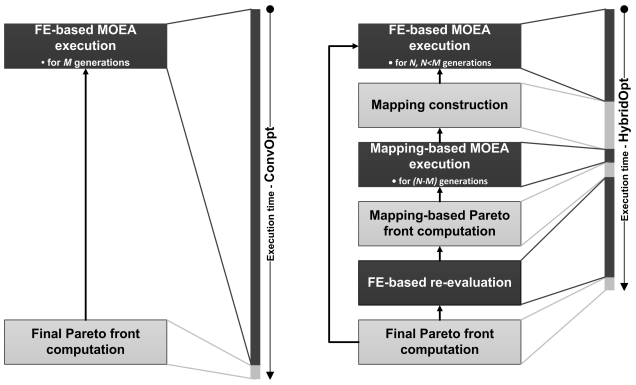


Fig. 1. Diagram of the state-of-the-art, conventional optimization process - ConvOpt (left side) and of our ANN-based hybrid approach - HybridOpt (right side)

Basic Idea Figure 1 contains an overview of the computational stages of the two optimization processes when wishing to evolve a total of M generations. Because of the very lengthy runs, for both methods, the result of a single run is the Pareto front extracted from the combined set of all the evaluated individuals.

In the *FE-based MOEA execution stage* the first N generations of each MOEA run are computed using FE simulations and all the valid individuals evaluated at this stage will form the training set used to construct ANN mappings. Each sample in this training set contains the initial electrical motor design parameter values and the corresponding target output values computed using FE simulation software. In the *mapping construction stage*, we use systematic parameter variation and a selection process that takes into consideration both accuracy and architectural simplicity in order to find the most robust ANN de-

sign for each of the considered target variables and train the appropriate ANN mapping models.

The next step is to switch the MOEA to a mapping-based evaluation function for the remaining generations that we wish to compute (*mapping-based MOEA execution stage*). The mapping-based evaluation function is extremely fast when compared to its FE-based counterpart, and it enables the prediction of targets based on input variables within milliseconds.

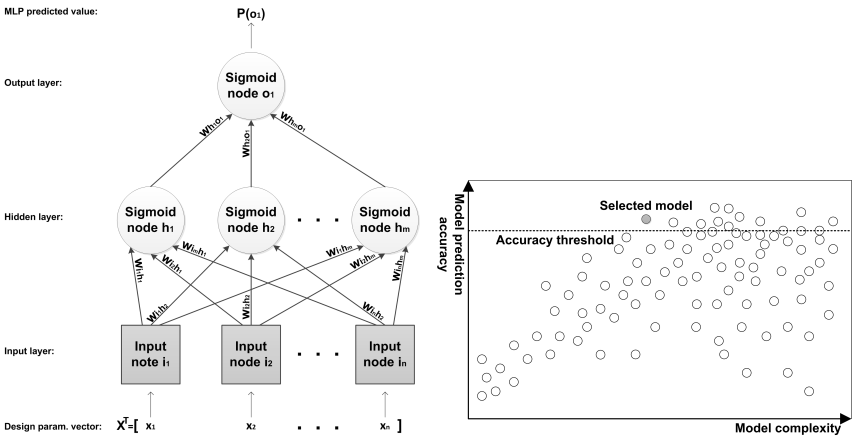
In the *mapping-based Pareto front computation stage* a preliminary mapping-based Pareto front is extracted only from the combined set of individuals evaluated using the mappings. All the solutions contained in the mapping-based Pareto front are re-evaluated using FE computations in the next stage of HybridOpt (the *FE-based reevaluation stage*). This is necessary in order to assure geometric valid solutions. In the *final Pareto front computation stage*, the final Pareto front of the simulation is extracted from the combined set of all the individuals evaluated using FE simulations.

ANN-based mapping - Structure and Training The multilayer perceptron (MLP) architecture (Figure 2(a)) consists of one layer of input units (nodes), one layer of output units and one or more intermediate (hidden) layers. MLPs implement the feed-forward information flow which directs data from the units in the input layer through the units in the hidden layer to the unit(s) in the output layer. Any connection between two units u_i and u_j has an associated weight $w_{u_i u_j}$ that represents the strength of that respective connection. The weights are initialized with small random values and they are subsequently adjusted during a training process based on the standard back-propagation algorithm [18].

In our modeling tasks, we use MLPs that are fully connected and have a single hidden layer. The number of units in the input layer is equal to the size of the design parameter vector. Also, as we construct a different mapping for each target variable in the data sample, the output layer contains just one unit and, at the end of the feed-forward propagation, the output of this unit is the predicted regression value of the elicited target, e.g. $P(o_1)$ for the MLP presented in Figure 2(a).

We have chosen to adopt an early stopping mechanism that terminates the execution whenever the prediction error computed over a validation subset V does not improve over 200 consecutive iterations, thus preventing the model from over-fitting. This validation subset is constructed at the beginning of the training process by randomly sampling 20% of the training instances.

ANN-based mapping - Evaluation and Automatic Model Selection One important issue in our approach concerns an appropriate automatic selection of parameters in order to construct an accurate and robust model (mapping) in terms of expected prediction quality on new data. The basic idea is to conduct a best parameter grid search, iterating over different parameter value combinations (including number of hidden neurons, the learning rate and the momentum). For



(a) Diagram of MLP (b) The model selection strategy

Fig. 2. Multilayer perceptron architecture and model selection strategy

each parameter combination, a model is constructed and its predictive quality is assessed using 10-fold cross validation. This leads to a fairly large pool of different models.

Our new, automatic, model selection strategy (Figure 2(b)) is divided into two stages. Initially, all the models are ranked according to a metric that takes into account the accuracy of their predictions measured in terms of the mean R^2 over all folds minus the standard deviation of R^2 over all folds. The latter reflects the sensitivity of the model for the concrete choices of the folds, indicating whether a mapping is biased towards specific regions of the search space. Next, an *accuracy threshold* is computed as the mean of the accuracies of the best performing 2% of all models. The choice of the final model is made using a complexity metric that favors the least complex model (i.e., lowest number of hidden units) which has a predictive accuracy higher than the accuracy threshold.

3 Evaluation and Results

3.1 The Optimization Scenarios

We consider three multi-objective optimization scenarios from the field of designing and prototyping electrical drives:

The first scenario (Scenario *OptS1*) is on an electrical drive featuring a slotted stator with concentrated coils and an interior rotor with buried permanent magnets. The rotor and stator topologies are shown in Figure 3. The design parameter vector is given by $\mathbf{X}^T = \left[h_m \ \alpha_m \ e_r \ d_{si} \ b_{st} \ b_{ss} \right]$, where all parameters are shown in Fig. 3 except for α_m, which denotes the ratio between the actual magnet size and the maximum possible magnet size as a result of all other geometric parameters of the rotor. For this scenarios, the targets of the ANN mapping construction phase are the four, unconstrained, Pareto objectives:

- $-\eta$ - where η denotes the efficiency of the motor. In order to minimize the losses of the motor, the efficiency should be maximized and therefore $-\eta$ is selected for minimization.
- T_{cogPP} - the peak-to-peak-value of the motor torque for no current excitation. This parameter denotes the behavior of the motor at no-load operation and should be as small as possible in order to minimize vibrations and noise due to torque fluctuations.
- $TotalCosts$- the material costs associated with a particular motor. Obviously, minimizing this objective is a very important task in most optimization scenarios.
- T_{rippPP} - the equivalent of $T_{cog,PP}$ at load operation. The values of this objective should also be as small as possible.

The second problem (Scenario *OptS2*) is on an electrical machine featuring an exterior rotor. The design parameter vector contains seven geometric dimensions. The aim of this optimization problem was to simultaneously minimize the total losses of the system at load operation (unconstrained Pareto objective) and the total mass of the assembly (constrained Pareto objective). This scenarios also contains a secondary constraint (2) imposed on a geometrical dimension of the evolved motor designs. This means that, for this scenario, we have a total of three targets in the mapping construction stage.

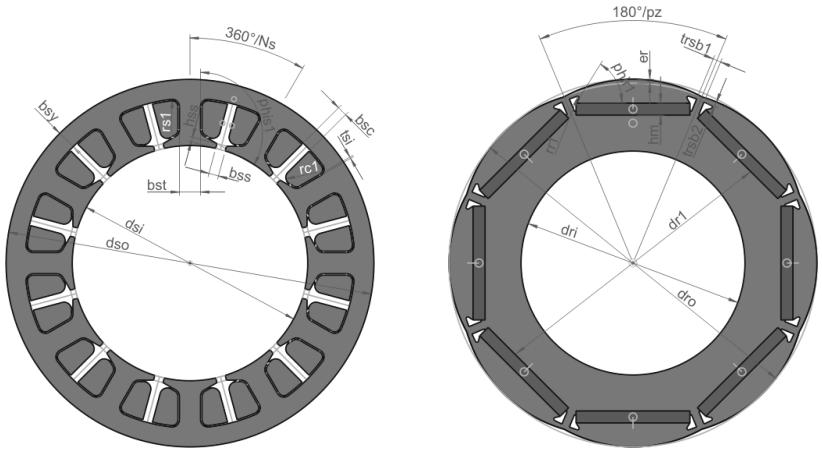
The third problem (Scenario *OptS3*) also concerns a motor with an exterior rotor. The design parameter vector has a size of ten. This scenario proposes four constrained Pareto objectives: 1.) l_s - the total axial length of the assembly, 2.) $TotalMass$ - the total mass of the assembly, 3.) P_{Cu}- the ohmic losses in the stator coils, 4.)P_{fe} - the total losses due to material hysteresis and eddy currents in the ferromagnetic parts of the motor.

3.2 The Testing Framework

Both ConvOpt and HybridOpt use the NSGA-II and SPEA2 implementations provided by the jMetal package [7]. In case of all the tests reported in Section 3.4, we used NSGA-II with a crossover probability of 0.9, a crossover distribution index of 20, a mutation probability of 0.2 and a mutation distribution index of 20. These are standard values recommended by literature [5] and set as default in jMetal. We conducted a preliminary tuning phase with ConvOpt to check whether different settings for the crossover and mutation distribution indexes would yield better results but we found no improvement over the standard values.

In the case of HybridOpt, we performed the mapping training stage after $N = 25$ generations (for the motivation of this choice, please see Section 3.4). As we used a population size of 50, the maximum possible size of the training sets is 1250. The size of the actual training sets we obtained was smaller, ranging from 743 to 1219 samples. This is because some of the evolved design configurations were geometrically unfeasible or invalid with regards to given optimization constraints.

The MLP implementation we used for our tests is largely based on the one provided by the WEKA (Waikato Environment for Knowledge Analysis) open



(a) Stator (b) Rotor

Fig. 3. Interior rotor topology with embedded magnets

source machine learning platform [11]. In the case of the best parameter grid searches that we performed in order to create the MLP-based mappings: 1.) the *number of hidden units* was varied between 2 and double the number of design variables, 2.) the *learn rate* was varied between 0.05 and 0.40 with a step of 0.05 and 3.) the *momentum* was varied between 0.0 and 0.7 with a step of 0.1.

The search is quite fine grained as it involves building between 704 (scenario *OptS1*) and 1216 (scenario *OptS3*) MLP models for each elicited target. This approach is possible because we make use of the early stopping mechanism in the model training process (see Section 2.2) which in turn assures a quite low average MLP model training time of 361.12 seconds. We achieve a considerable speedup in the mapping creation stage by distributing all the MLP training tasks over a cluster computing environment that is also used to run in parallel required FE-simulations. As a result, the mapping creation stage took, on average, 149.26 minutes, over all performed tests.

3.3 Considered Performance Metrics

In order to compare the performance and behavior of the conventional and hybrid optimization processes we use four performance metrics: 1.) the hypervolume metric $\mathcal{H}$ [9] measures the overall coverage of the obtained Pareto set; 2.) the generalized spread metric $\mathcal{S}$ [20] measures the relative spatial distribution of the non-dominated solutions; 3.) the FE utility metric $\mathcal{U}$ offers some insight on the efficient usage of the FE evaluations throughout the simulation (higher values are better); 4.) the run-time metric $\mathcal{T}$ records the total runtime in minutes required by one simulation; The $\mathcal{H}$ metric has the added advantage that it is the only MOEA metric for which we have theoretical proof [9] of a monotonic behavior.

Table 1. The average performance over five runs of ConvOpt and HybridOpt

Metric	Scenario *OptS1*		Scenario *OptS2*		Scenario *OptS3*	
	ConvOpt	HybridOpt	ConvOpt	HybridOpt.	ConvOpt	HybridOpt
$\mathcal{H}$	**0.9532**	0.9393	**0.8916**	0.8840	**0.4225**	0.3691
$\mathcal{S}$	0.7985	**0.6211**	0.8545	**0.8311**	**0.4120**	0.4473
$\mathcal{U}$	0.1315	**0.2210**	0.0016	**0.0064**	**0.2901**	0.2362
$\mathcal{T}$	2696	**991**	3798	**1052**	4245	**2318**

This means that the maximization of the hypervolume constitutes the necessary and sufficient condition for the set of solutions to be *maximally diverse Pareto optimal solutions*. In the case of the $\mathcal{S}$ metric, a value closer to 0.0 is better, indicating that the solutions are evenly distributed in the result space.

3.4 Results

In order to obtain a quick overview of the performance of linear models compared to ANN non-linear mappings, we conducted some preliminary modeling tests on the 11 targets of all three optimization scenarios: it turned out that the average model quality measured in terms of R^2 was 0.859 in case of the linear models, whereas ANNs achieved a value of 0.984. On three targets, the linear models were almost useless, as they achieved an R^2 of around 0.6.

In the current version of HybridOpt, it is very important to choose a good value for the parameter N that indicates for how many generations we wish to run the initial FE-based execution stage.

Finally, based on extensive tests, we have chosen $N = 25$ as this is the *smallest value of N for which the trained models exhibit both a high prediction accuracy as well as a high prediction stability*. Over all three data sets and the 6 non-linear targets, the generational coefficients of determination (for generations 31 to 100) obtained by the models constructed using samples from the first 25 generations 1.) are higher than 0.9 in 94.52% of the cases; and 2.) are higher than those obtained by the models constructed using 20-24 generations in 57.52% of the cases and by those obtained by the models constructed using 26-30 generations in 42.52% of the cases.

The comparative performance of ConvOpt and HybridOpt is presented in Table 1. The results for each scenario are averaged over five optimization runs. On the highly constrained scenario *OptS3*, the hybrid optimization process is a little bit worse. The main reason for this is that the hard constraints determine a high ratio of invalid individuals to be generated during the mapping based evaluation stage. However, the computation time could still be reduced by $\approx$ 45%. Even though, for this scenario, ConvOpt produces Pareto fronts with a better $\mathcal{H}$, HybridOpt is still able to evolve well balanced individual solutions in key sections of the Pareto front — please see Figure 4 for two such examples.

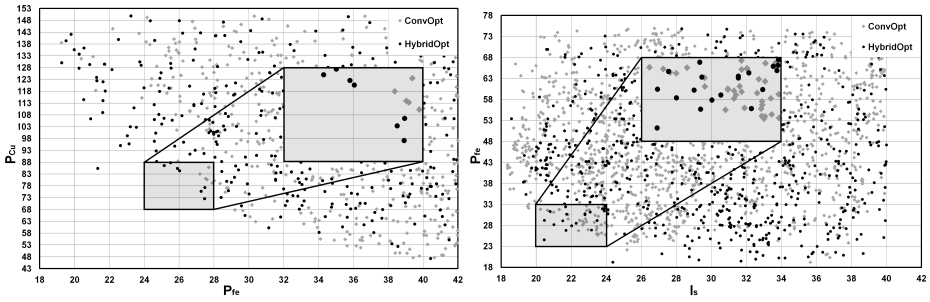


Fig. 4. 2D projections of the full Pareto fronts for the highly constrained scenario *OptS3*. The black dots denote solutions obtained using HybridOpt. These solutions are relatively equally distanced, with regards to the considered objectives, from the origin of the projected Pareto space.

4 Conclusion

In this paper, we investigated multi-objective optimization algorithms based on evolutionary strategies, using the famous and widely used NSGA-II algorithm, for the purpose of optimizing the design of electrical drives in terms of efficiency, costs, motor torque behavior, total mass and others. As the design and the target parameter space is quite large, we end up having complex optimization scenarios that require very long computation runs.

In order to alleviate this problem, we experimented with a system that automatically creates, on-the-fly, non-linear MLP-based mappings between design and target parameters. Empirical observations over averaged results indicate that, by replacing the time-intensive FE simulations with fast mapping based estimations of the target values, we are able to reduce the overall run-time of the optimization process, in average, by more than 60%. At the same time, the hybrid optimization process is able to produce Pareto fronts of similar quality to the ones obtained using the conventional FE-based optimization.

References

1. Abraham, A., Corchado, E., Corchado, J.M.: Hybrid learning machines. Neurocomputing 72(13-15), 2729–2730 (2009)
2. Alaya, I., Solnon, C., Ghedira, K.: Ant colony optimization for multi-objective optimization problems. In: Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on. vol. 1, pp. 450 –457 (oct 2007)
3. De Keulenaer, H., Belmans, R., Blaustein, E., Chapman, D., De Almeida, A., De Wachter, B., Radgen, P.: Energy efficient motor driven systems. Tech. rep., European Copper Institute (2004)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley-Interscience Series in Systems and Optimization, John Wiley & Sons, Chichester (2001)

5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182 –197 (apr 2002)
6. Duan, Y., Ionel, D.: A review of recent developments in electrical machine design optimization methods with a permanent magnet synchronous motor benchmark study. In: 2011 IEEE Conference on Energy Conversion Congress and Exposition (ECCE). pp. 3694 –3701 (sept 2011)
7. Durillo, J.J., Nebro, A.J.: JMETAL: A java framework for multi-objective optimization. Advances in Engineering Software 42, 760–771 (2011)
8. European Union: Regulation (eg) nr. 640/2009
9. Fleischer, M.: The measure of pareto optima. applications to multi-objective metaheuristics. In: Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003. pp. 519–533. Springer (2003)
10. Grosan, C., Abraham, A.: Intelligent Systems: A Modern Approach. No. ISBN 978-3-642-21003-7 in Intelligent Systems Reference Library Series, Springer Verlag, Germany (2011)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18 (November 2009)
12. Haykin, S.: Neural Networks: A Comprehensive Foundation (2nd Edition). Prentice Hall Inc., Upper Saddle River, New Jersey (1999)
13. Hong, Y.S., Lee, H., Tahk, M.J.: Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. Engineering Optimization 35(1), 91–102 (2003)
14. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359 – 366 (1989)
15. Jannot, X., Vannier, J., Marchand, C., Gabsi, M., Saint-Michel, J., Sadarnac, D.: Multiphysic modeling of a high-speed interior permanent-magnet synchronous machine for a multiobjective optimal design. Energy Conversion, IEEE Transactions on 26(2), 457 –467 (june 2011)
16. Paliwal, M., Kumar, U.A.: Neural networks and statistical techniques: A review of applications. Expert Systems with Applications 36(1), 2 – 17 (2009)
17. del Valle, Y., Venayagamoorthy, G., Mohagheghi, S., Hernandez, J.C., Harley, R.: Particle swarm optimization: Basic concepts, variants and applications in power systems. Evolutionary Computation, IEEE Transactions on 12(2), 171 –195 (april 2008)
18. Werbos, P.J.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis, Harvard University (1974)
19. Zhao, S.Z., Iruthayarajan, M.W., Baskar, S., Suganthan, P.: Multi-objective robust pid controller tuning using two lbests multi-objective particle swarm optimization. Information Sciences 181(16), 3323 – 3335 (2011)
20. Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., Tsang, E.: Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 892 –899 (2006)
21. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001). pp. 95–100. International Center for Numerical Methods in Engineering (CIMNE) (2002)