Developing Automatic Multi-Objective Optimization Methods for Complex Actuators

Radu CHIȘ[1], Lucian VINȚAN[1]
[1]*"Lucian Blaga" University of Sibiu, Bulevardul Victoriei 10, 550024, Sibiu, Romania*
radu.chis@ulbsibiu.ro, lucian.vintan@ulbsibiu.ro

***Abstract*—This paper presents the analysis and multiobjective optimization of a magnetic actuator. By varying just 8 parameters of the magnetic actuator's model the design space grows to more than 6 million configurations. Much more, the 8 objectives that must be optimized are conflicting and generate a huge objectives space, too. To cope with this complexity, we use advanced heuristic methods for Automatic Design Space Exploration. FADSE tool is one Automatic Design Space Exploration framework including different state of the art multi-objective meta-heuristics for solving NP-hard problems, which we used for the analysis and optimization of the COMSOL and MATLAB model of the magnetic actuator. We show that using a state of the art genetic multi-objective algorithm, response surface modelling methods and some machine learning techniques, the timing complexity of the design space exploration can be reduced, while still taking into consideration objective constraints so that various Pareto optimal configurations can be found. Using our developed approach, we were able to decrease the simulation time by at least a factor of 10, compared to a run that does all the simulations, while keeping prediction errors to around 1%.**

***Index Terms*—Actuators, Computer Aided Engineering, Machine Learning, Pareto Optimization, Response Surface Methodology**

I. Introduction

Complex actuators are used in many different fields ranging from door openers, to car parts and robotics. Their optimization for various conflicting performance objectives, with adverse interactions is becoming more interesting, necessary, and important. Due to their ubiquity, any reduction of the manufacturing costs (smaller size), energy consumption or operating temperatures has a significant impact in a very competitive global market. Finding optimal solutions in a reasonable amount of time for such NP-hard Design Space Exploration (DSE) problems, which optimize multiple objectives, can be done through Automatic Design Space Exploration (ADSE) tools. These methods have been proven to be efficient in finding, in a feasible amount of time, using different advanced meta-heuristics, an acceptable Pareto front (hyper-surface) approximation, consisting of good configurations regarding multiple objectives.

In this article, we present the automatic optimization process of a magnetic actuator model created in COMSOL and MATLAB provided by Continental Automotive Systems, Sibiu branch using our developed FADSE optimization tool. The model has 8 input parameters that can be varied and a number of 8 outputs. Some objectives have to be maximized and some minimized, most of them being conflicting. We used state of the art genetic multiobjective algorithms, machine learning techniques, design of experiments and response surface modelling methods to minimize the number of simulations to be executed in our optimization process.

The article is structured as follows: Section 2 provides an overview of the related work. The used software tools (FADSE, COMSOL and MATLAB) are presented in Section 3, while Design Space Exploration and Response Surface Modelling concepts along with the model's objectives and parameters used for the optimization algorithms are presented in Section 4. Our research methodology and the obtained results are shown in Section 5 and, finally, Section 6 concludes the paper and suggests directions for further work.

II. Related Work

Different researches have been conducted to optimize major engineering branches. Our FADSE software tool [1] has been extensively used for the automatic design space exploration of advanced computer architectures, as it is shown in [2-3]. The authors focus their research exclusively on optimizing advanced computing systems, like the Sniper multicore simulator and the Grid ALU processor (GAP). In [2] a new meta-optimization layer is added in order to improve the performance of the DSE algorithms, by driving two different multi-objective metaheuristics in parallel. The authors show that this approach generates better configurations for the GAP superscalar micro-architecture in half of the time compared to a classical sequential optimization approach. In [3] a multi-objective co-optimization approach is shown, where both the hardware and software parameters of the Sniper multi-core simulator are optimized. It was shown that by running a DSE process that can concurrently optimize both hardware and software variables, better configurations in regard to the 4 objectives can be found. Application-specific multiprocessor systems-on-chip (MPSoCs) are optimized using ReSPIR framework in [4]. The authors present a new DSE methodology that leverages design of experiment (DoE) and response surface modeling (RSM) to manage system-level constraints. During the DoE phase, the RSM is trained to coarsely estimate the objective space. Then the RSM method is used to guide the simulation process towards feasible solutions.

Useful optimizations of wings in aerospace engineering using design of experiments, response surface models, data-fusion methods and computational fluid dynamics are presented in [5]. Metamodels are computed using a three-dimensional Computing Fluid Dynamic (CFD) code and the authors show that they are more accurate than the initial empirical model generated by a RSM technique. Their

optimization process is much quicker than direct searches of the CFD. The work presented in [6] describes a new approach to design and optimize embedded systems in the design phase, based on Pareto multi-objective optimization. The authors defined two Domain Specific Languages and develop a framework that will design the architecture model and the component diagram of embedded systems, while the parameters for the models are automatically generated using the Text Template Transformation Toolkit.

Moving towards electromagnetic devices optimizations, these have been done in different directions: [7] shows an optimal approach for designing electrical machines to reduce mechanical deformation in magneto mechanical systems by deriving a design sensitivity equation by employing the adjoint variable method (AVM) in order to avoid multiple sensitivity evaluations for the coupled analysis. The authors optimize a simple core used in a magnetic levitation system. In [8] a DSE process is run to find a current system for solenoids that produce a uniform magnetic field inside the solenoid and a low stray field outside of it. The authors present four interesting different approaches for the DSE process using evolution strategies to minimize all objectives.

In contrast with the approaches presented in [6-8] we are developing an original method to optimize a magnetic actuator using state of the art multi-objective genetic algorithms and also using machine learning techniques, DoE and RSM methods to minimize the number of simulations, thus accelerating the whole DSE process. We also implemented some useful data mining techniques to find dependence relationships between inputs and outputs in order to reduce the huge output space. Strictly related to the used RSM method, our approach is in part similar to the one proposed in [4], but in contrast to that approach, we used all our previously simulated configurations stored in the database to train the neural net and took the DSE process further: we ran a multi-objective genetic algorithm on generations consisting of 20.000 individuals using our "Fast Simulator" that will be presented in Paragraph V. In addition, we simplified the neural networks complexity, by removing dependent outputs through a feature selection method. This way, after the training process we could almost instantly predict all outputs for all the 6 million configurations of our design space. This way we could select our top candidates and run the simulations. We also tried a "backwards-training-approach" to predict the input parameters from our "desired" objectives. This way we could get some different configurations and also predict the inputs for a "super-configuration" - a configuration that has the objectives set to the minimum values found so far that could in theory dominate all other configurations.

III. The Used Software Tools: FADSE, COMSOL and MATLAB

In this section we are going to present the software tools used for our DSE process. Framework for Automatic Design Space Exploration (FADSE) has been developed by former Ph. D. student Horia Calborean under the supervision of Prof. Lucian Vinţan at the "Lucian Blaga" University of Sibiu and first presented in [9]. This framework allows the automatic DSE of different problems using state of the art evolutionary algorithms (NSGA-II, SPEA-2, etc.), particle swarm (SMPSO) multi-objective (Pareto) algorithms from the jMetal library [10]. In order to improve the algorithms' convergence but also the solutions' quality domain-knowledges related to the optimized target systems have been used by FADSE in [11]. For the performance evaluations of the different algorithms FADSE implements some quality indicators that do not require the true Pareto front, like the followings: Coverage [12], Hypervolume [12] and Two Set Hypervolume Difference [13].

FADSE can be connected to any simulator, computer system simulator or of another kind, through a specific connector. For the current research, a new connector for MATLAB and Comsol was written and will be presented in the next paragraph. Among the computer architecture simulators optimized through FADSE we would mention the Sniper Multicore / Manycore Simulator [14], GAP superscalar processor simulator and its compiler, called Gaptimize [15], M-SIM3 multicore simulator
[16], and many others. It is the first time when we adapt and use our FADSE tool to optimize a complex magnetic actuator simulator.

COMSOL represents a multiphysics modeling software solution. It is used to develop easy-to-use software for modeling and simulation of real world multiphysics systems and is one of the primary tools for engineers, researchers and lecturers in the education and hi-tech product design fields. It has expanded to include a suite of discipline-specific add-on modules for Structural Mechanics, High and Low Frequency Electromagnetics, Fluid Flow, Heat Transfer, Chemical Reactions, MEMS, Acoustics, and more. The developers have added LiveLink products for CAD software, MATLAB®, and Excel® that deliver a seamless integration between COMSOL Multiphysics simulations and CAD packages, MATLAB and Excel. The integration with MATLAB is of importance to us.

MATLAB (MATrix LABoratory) is a well-known multi-paradigm numerical computing environment. MATLAB allows matrix manipulations, plotting of functions and data, implementation of advanced control algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python.

IV. Actuator Model, Design Space Exploration and Response Surface Modeling

Our design space exploration process is conducted on a COMSOL and MATLAB implemented model provided by Continental Automotive Systems, Sibiu Branch (Romania). The designers created two models, one magnetic and one thermal, which we used for the simulations and optimizations processes. The magnetic model is depicted in Fig. 1 while the thermal model is presented in Fig. 2.

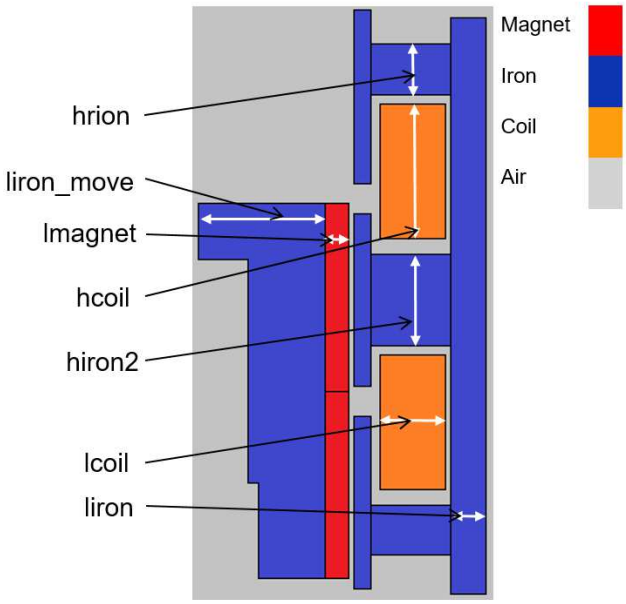

Figure 1. Magnetic Model

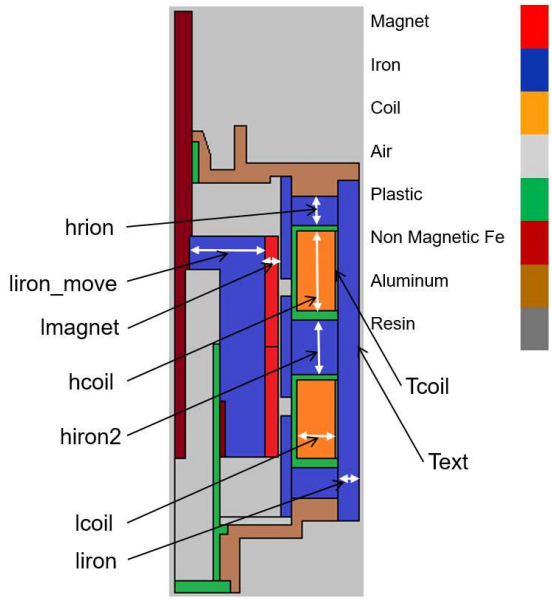

Figure 2. Thermal Model

The input parameters that can be varied in the model are presented in Table I. Although any values between the minimum and maximum are valid, in order to have a reasonable search space we have set also the corresponding step. This way our search space consists of more than 6 million configurations. As we will show further (Par. 5), an automatic exhaustive search in this huge space is prohibited. Therefore we need to develop and implement heuristic optimization methods.

The output parameters are presented in Table II. Some of these objectives have to be maximized while others have to be minimized. In order to run our ADSE method we changed the objectives that have to be maximized to the value of 1/objective to run a minimization problem for all the outputs. Additionally, some of the parameters have also some constraints that should not be exceeded. In our case the configurations that have an iron magnetic flux density >2.2 Tesla or a coil fill factor > 0.75 or an external temperature > 80 degrees Celsius degrees are infeasible and should not be taken into consideration.

The first six objectives in Table II are computed by the magnetic model and the last two are computed by the temperature model. The actuator works by sliding up and down the mobile part, which is on the left side of Fig. 1 and Fig. 2. This can be done by varying the current through the coils, thus changing the magnetic field. The changes in the magnetic field move the entire left part that contains a permanent magnet and iron up and down, thus creating the actuation.

In our DSE process we used a well-known state of the art multi-objective meta-heuristic, called NSGA-II, which utilizes the concept of Pareto efficiency. This genetic algorithm was first described by Deb in [17]. It uses the three genetic operators (selection, crossover and mutation) to generate future generations of individuals. The selection operator is an elitist one and it is implemented as follows: the parent and offspring population are combined and sorted into so called "Pareto fronts" (surfaces in 3-D orthogonal spaces or even hyper-surfaces in n-D spaces, n>3) where the first front contains all the nondominated individuals, the second front contains the nondominated solutions if the first front is removed and so on. The selection of individuals for the next population starts at the first nondominated front. To differentiate individuals from the same Pareto front, which are quasi-equal in respect to their objectives, a crowding distance to the surrounding individuals is computed.

In order to compare our different runs we need some quality estimators. Because the true Pareto front is not known beforehand we are using the following estimators:

1. Coverage, proposed in [12], computes the percentage of individuals from a run (optimization method) that are not dominated by individuals in another run.

2. Hypervolume, proposed in [12], computes in a minimization problem the n-dimensional hypervolume enclosed by the current Pareto front and the hypervolume reference point (given by the maximum found values of all the objectives), while in a maximization problem it computes the n-dimensional hypervolume enclosed by the current Pareto front and the orthogonal axes.

3. Two set hypervolume difference, proposed in [13], represents an "extension" of the hypervolume indicator. Considering X1 and X2 as two sets of phenotype decision vectors, the TSHD is defined by the following formula:

$$TSHD(X1, X2) = HV(X1+X2) - HV(X2) \quad (1)$$

where $X1+X2$ is the union of the two vector sets, $X1$ and $X2$, and HV is the normal hypervolume indicator. This way $TSHD(X1,X2)$ gives the hypervolume of the portion of objectives space that is dominated by the first set but not by the second. If $TSHD(X1,X2)=0$ and $TSDH(X2,X1)>0$ we can say that $X2$ is definitely better than $X1$.

In our research, we tried to accelerate the DSE process by minimizing the number of simulations by leveraging Machine Learning, Design of Experiments (DoEs) and Response Surface Modelling (RSM) techniques. We tried to find out the correlations between inputs and outputs but also between outputs and outputs to observe the potential independence of the different objectives. After this, we used

TABLE I. INPUT PARAMETERS

	Min	Max	Step	Values	Description
I	2	6	0.5	9	current through coils (A)
hcoil	0	4	0.5	9	coil height (cm)
lcoil	0	2	0.5	5	coil length (cm)
lmagnet	0	0.4	0.05	9	magnet length (cm)
liron_move	0	3	0.5	7	iron move length (cm)
liron	0	1.6	0.4	5	iron length (cm)
hiron	0	3	0.5	7	iron height (cm)
hiron2	0	3	0.5	7	iron2 height (cm)

TOTAL: 6251175 possible combinations

TABLE II. OPTIMIZATION OBJECTIVES

	Optimization	Constraint	Description
MEAN	Maximize		mean value of force curve (N)
STDDEV	Minimize		standard deviation of force
BMAX	Minimize	<= 2.2	iron magnetic flux density (T)
VOLM	Minimize		magnet volume (m3)
F_PER_MV	Maximize		force density per volume (N/m3)
FILLFACT	Minimize	<= 0.75	coil fill factor
TCOIL	Minimize		coil temperature (Celsius degree)
TEXT	Minimize	<= 80	exterior temperature (Celsius degree)

DoE [20] to gather information about the input parameters values and generate the initial set of points to be analyzed. We used two distinct DoE techniques, random and full factorial, for our test point selection but also all our previous simulations saved in the database.

The RSM model tries to find relations between the target systems' parameters (inputs) and the response variables (the outputs of the simulator). The RSM undergoes a phase where pairs of known inputs and known outputs are used to interpolate complex nonlinear relations g(x) with an error E. Thus, RSM techniques are introduced to estimate the objective functions $f(x) = g(x) + E$, therefore avoiding the time needed to simulate lots of configurations. The RSM model should use the DoE generated points and, based on these certain points it tries to build a response model of our actuator. Typically, a RSM has two phases: a training phase, where the response model is created and tuned and an evaluating/prediction phase, in which the outputs for unknown points are forecasted.

V. RESEARCH METHODOLOGY AND RESULTS TYPES OF GRAPHICS

As presented in Section IV, our configurations are evaluated on two COMSOL and MATLAB actuators models created by Continental Automotive Systems, Sibiu Branch. Our search space consists of 6,251,175 different configurations by varying the 8 input parameters presented in Table I. We developed some multiple optimization scenarios as we would show further in a detailed manner. Each DSE scenario was run with 200 individuals per generation for 30 generations. One simulation takes around 6 minutes to run and we have done the DSE process on 2 x Intel Xeon W3690 processors, clocked at 3.46 GHz, each having 6 Cores / 12 Threads and 12 GB of DRAM, running Windows 10, Matlab r2012b and Comsol 4.2. We were running 6 clients in parallel so that we were able to do 6 simulations in 6 minutes, which translates to ~ 1 minute per simulation on our PC system. Each 30-generation run optimization process takes about 3 days and generate ~4000 different configurations, which represents only ~ 0.6% of the full space of more than 6 million configurations. Simulating all the configurations on the same system would take more than 500 days; that is for sure unacceptable. Decreasing the step size in Table I would further increase the simulation time to infeasible amounts.

A. The Connector FADSE – Actuator

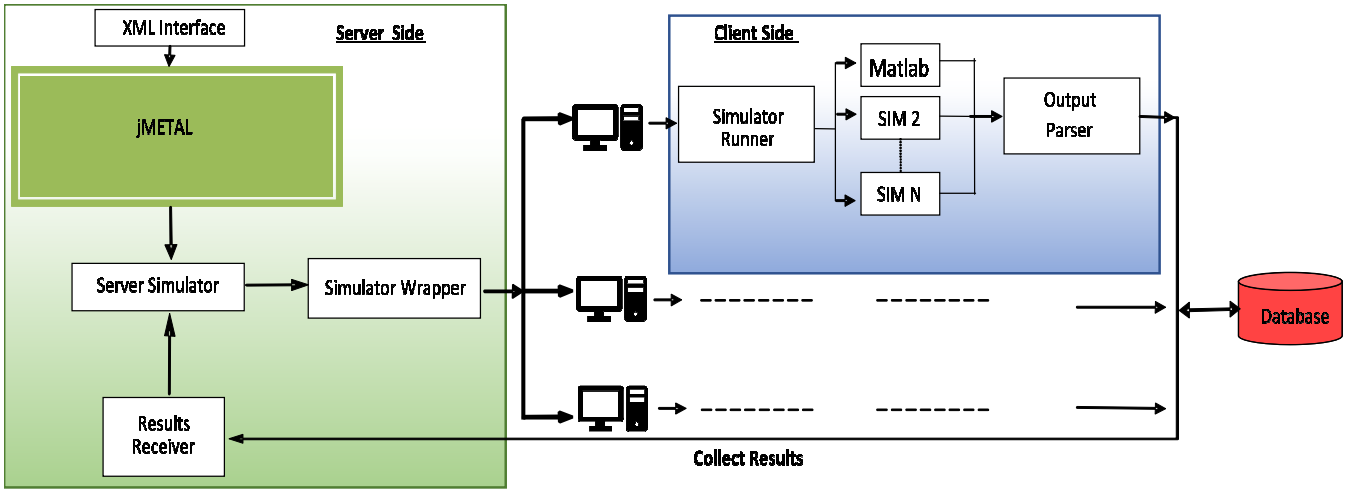

Figure 3. FADSE Architecture

In order to be able to run MATLAB from our FADSE tool a simple simulator connector had to be written as shown in Fig. 3. This connector has two main parts:

Simulator Runner, which translates the individuals from the NSGA-II algorithm to inputs for the actuator model simulator and starts the simulation process.

Output Parser, which waits for the simulation to finish, reads the outputs and translates these outputs back in the objectives of the NSGA-II individuals.

Until now our FADSE tool ran only with integer inputs (typical for our previous Computing Systems optimizations). For the simulation of the magnetic actuators we had to make some changes so that the inputs could accept not only the minimum and maximum values, but also the corresponding step, as presented in Table I.

Another modification that had to be made in FADSE tool was the addition of constraints to the outputs. FADSE supported rules and relations for the input parameters but we had to integrate the constraint values provided by Continental Automotive Systems Sibiu for some of the outputs, implementing thus a small specific domain-knowledge for our magnetic actuators.

As it is shown in Fig. 3, FADSE is a client-server application, where the multi-objective meta-heuristic is run on the server and the time-consuming simulations are distributed on multiple clients. Each client has a connector implemented, that translates the values from the individuals in the DSE algorithm (NSGA-II) to the actual simulator Matlab and Comsol and can translate the results of the simulation back in the individuals. We have seen that from a generation to another part of the best individuals remain and in order to not simulate them again we use a database to store and reuse our simulation results. This way we accelerate the DSE process by reusing the already computed results from the database.

We did run the DSE process over 30 generations of 200 individuals two times, starting from the initial random population: first time without the constraints on the objectives, hoping that we would get more diversity in the obtained results and the second time with the constraints (domain-knowledge) in place. We have computed the coverage, hypervolume and two set difference hypervolume quality estimators for the individuals that satisfy the constraints in both runs and the obtained results are shown in Paragraph C.

B. Developed Machine Learning Methods

We have regarded the actuator model until now as a black-box and regarded the inputs and outputs as independent for our two previous runs. Using numpy tool [21], a fundamental package for scientific computing with Python, we ran a script on over 6000 configurations that we have already simulated to highlight correlations, if any, among inputs and outputs or outputs and outputs. In other words, we designed and implemented useful Data Mining and Feature Selection Methods, in order to discover some relationships between inputs and outputs, respectively to reduce the huge outputs space, with significant benefits in reducing the RSM model's complexity. Table III shows our findings and we have considered the following:

- Green: strong correlation – absolute value bigger than 0.75
- Yellow: weak correlation – absolute value between 0.25 and 0.75
- Orange: no correlation – absolute value smaller than 0.25

Looking at Table III, we can conclude that there is a strong correlation between some input parameters and output parameters, as follows:

- I => Tcoil, Text and stddev. This means that the current through the coil strongly impacts the two temperatures and the standard deviation of the force.
- lcoil => Fillfact. This means that the coil length significantly impacts the fill factor.
- liron_move => Volm. This means that the correlation between the iron move length is indirect proportional to the volume.

Also looking at Table III we can conclude that the output parameters are not really independent, because strong correlations appear between:

- Tcoil, Text and stddev. This could be also inferred from the previous correlations through the distributive law because all 3 outputs significantly correlate to the

TABLE III. CORRELATIONS

	I	hcoil	lcoil	lmagnet	liron_move	liron	hiron	hiron2	TCOIL	F_PER_MV	MEAN	BMAX	VOLM	FILLFACT	TEXT	STDDEV
I	1.00	-0.12	0.07	-0.02	-0.06	0.01	0.00	0.00	0.97	0.32	0.32	0.51	0.05	-0.03	0.98	0.78
hcoil	-0.12	1.00	0.11	0.02	0.01	0.00	-0.02	-0.03	-0.13	-0.02	-0.02	-0.63	-0.02	0.68	-0.09	-0.02
lcoil	0.07	0.11	1.00	0.02	0.05	0.04	-0.05	-0.02	0.11	0.02	0.01	-0.09	-0.04	0.80	0.11	0.03
lmagnet	-0.02	0.02	0.02	1.00	0.21	0.06	0.05	0.01	0.00	0.05	-0.03	-0.11	-0.75	0.02	0.00	-0.13
liron_mov	-0.06	0.01	0.05	0.21	1.00	0.10	0.04	0.02	-0.03	-0.05	-0.13	-0.40	-0.81	0.04	-0.03	-0.46
liron	0.01	0.00	0.04	0.06	0.10	1.00	0.02	0.03	0.10	-0.03	-0.05	-0.11	-0.10	0.03	0.09	0.03
hiron	0.00	-0.02	-0.05	0.05	0.04	0.02	1.00	0.04	0.01	-0.08	-0.08	0.07	-0.06	-0.05	0.01	0.27
hiron2	0.00	-0.03	-0.02	0.01	0.02	0.03	0.04	1.00	0.00	-0.06	-0.06	0.21	-0.02	-0.04	-0.01	0.09
TCOIL	0.97	-0.13	0.11	0.00	-0.03	0.10	0.01	0.00	1.00	0.32	0.31	0.48	0.02	-0.01	1.00	0.76
F_PER_MV	0.32	-0.02	0.02	0.05	-0.05	-0.03	-0.08	-0.06	0.32	1.00	0.99	0.15	0.00	0.01	0.32	0.22
MEAN	0.32	-0.02	0.01	-0.03	-0.13	-0.05	-0.08	-0.06	0.31	0.99	1.00	0.18	0.11	0.00	0.31	0.26
BMAX	0.51	-0.63	-0.09	-0.11	-0.40	-0.11	0.07	0.21	0.48	0.15	0.18	1.00	0.34	-0.45	0.46	0.53
VOLM	0.05	-0.02	-0.04	-0.75	-0.81	-0.10	-0.06	-0.02	0.02	0.00	0.11	0.34	1.00	-0.04	0.02	0.39
FILLFACT	-0.03	0.68	0.80	0.02	0.04	0.03	-0.05	-0.04	-0.01	0.01	0.00	-0.45	-0.04	1.00	0.02	0.00
TEXT	0.98	-0.09	0.11	0.00	-0.03	0.09	0.01	-0.01	1.00	0.32	0.31	0.46	0.02	0.02	1.00	0.76
STDDEV	0.78	-0.02	0.03	-0.13	-0.46	0.03	0.27	0.09	0.76	0.22	0.26	0.53	0.39	0.00	0.76	1.00

TABLE IV. THE OBTAINED NORMALIZED ROOT MEAN SQUARE ERRORS

Input->Output	**MEAN**	**BMAX**	**VOL**	**FILLFACT**	**TEXT**	**STDDEV**		
NRMSE	0.46%	1.74%	0.07%	0.07%	0.07%	0.07%		
Output->Input	**I**	**hcoil**	**lcoil**	**lmagnet**	**liron_move**	**liron**	**hiron**	**hiron2**
NRMSE	0.14%	5.63%	4.36%	7.93%	7.19%	7.05%	14.55%	22.72%

current (I).

- F_per_mv and mean. This is also a valid observation because the formula for the F_per_mv in the actuator model is: F_per_mv = Force/magnet volume, while the mean represents the mean of the Forces computed in 20 different places.

One conclusion for the observations in Table III could be that not all the output parameters are independent and the very strong correlations, having values about 1, between Tcoil - Text and F_per_mv - mean allow us to remove the dependent values. In this way, we can reduce the objective space from 8 dimensions to just 6 dimensions. This output selection process will significantly reduce the complexities of our implemented neural RSM models, as it would be shown further.

Another conclusion is that in order to be able to approximate the other outputs from the inputs in an analytical manner (thus avoiding laborious simulations) we need more than just a simple correlation/linear interpolation; higher interpolation orders are needed. For this purpose, we have used DoE and neural RSM models.

For our DoE, we have used two well-known techniques for selecting 256 individuals: random (the configurations are selected randomly by following a uniformly distributed function) and full factorial (we have selected the minimum and maximum values of the input parameters and created a Cartesian product of 8 lists).

Also, because we already have about 20000 different simulated individuals in our database, we selected these configurations for our DoE.

We decided to use a RSM method to provide an analytical representation of the input parameters. As a RSM model we used supervised multi-layer perceptron Artificial Neural Networks (ANN) from the TensorFlow [18] using Keras [19] libraries.

TensorFlow™ is an open source software library for numerical computation using data flow graphs. It was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation.

We have created two different ANN models. The first one is predicting the outputs values from some given inputs values (6 different ANNs, each one with 8 inputs and one output). The second one tries to predict the corresponding inputs values from some certain outputs values (8 different ANNs, each one with 6 inputs and one output). We have done a manual DSE of the ANN that we used as RSM. We have tried with one and two hidden layers, each having 10, 20, 50 and 100 neurons. We also varied the number of epochs between 500, 1000, 5000 and 10000 and the batch sizes between 50, 100 and 250. Based on the obtained results we have selected a multi-layer perceptron ANN with two hidden layers, each layer having 50 neurons. During the training phase, we have supplied 70% of our 20000 individuals in batches of 100 for 5000 epochs for the Adam [22] learning algorithm, where all parameters were left to the default values presented in the initial article. During the evaluation phase, we have used the remaining 30% of our individuals to compute the normalized root mean square error.

First, we used the RSM neural models that predict the outputs to estimate the whole 6 million search space. This way, after providing as inputs all possible configurations we would have all the objectives values "computed" (predicted). Afterwards, we selected the top 200 individuals, as being the approximate Pareto set. In other words, we provide an original method to approximate the ideal unknown Pareto set for this NP-hard optimization problem.

Second, using the "backwards" training, from the given outputs values to the predicted inputs values, we tried to directly predict the input parameters values for some of our desired outputs values. This way, we could set each of the output to the minimum found value belonging to our

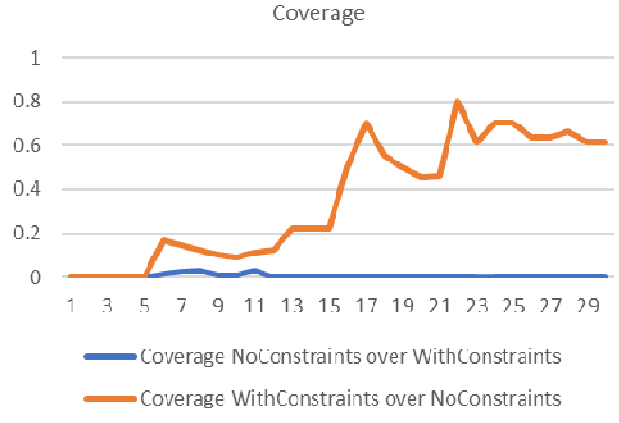

Figure 4. Coverage of feasible individuals

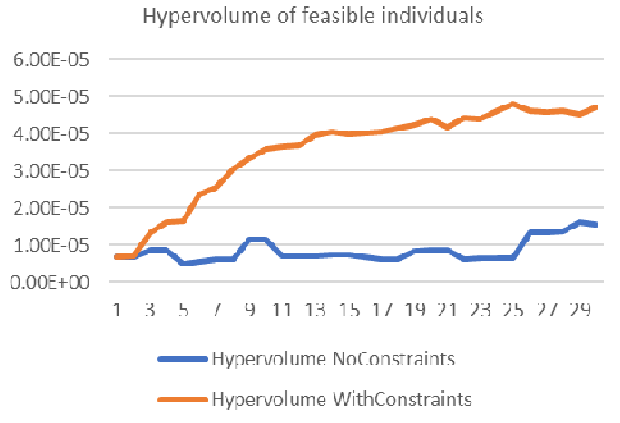

Figure 5. Hypervolume of feasible individuals

implemented database and the input parameters for that specific configuration would be computed (predicted).

The training phase with just 256 configurations, random and full factorial, yielded bad results in the evaluation phase, but this is understandable because 256 individuals account for just 0.0042% of the entire search space.

Next, we used about 14000 individuals belonging to our database (70% of the entire database) to train the neural networks. Training each ANN takes around 15 minutes for the 5000 epochs. During the evaluation phase, we computed the Normalized Root Mean Square Error (NRMSE) of the ANNs; our obtained results are shown in Table IV. We used in this evaluation phase about 6000 individuals, representing 30% of the total number of individuals belonging to our database. After the training and evaluation phases, we used the models to predict some "good" configurations that we have simulated after. The prediction of all 6 million configurations took about 5 minutes, which means that we can predict 1 million configurations per minute, while we could simulate just one configuration per minute. Thus, the needed time for prediction is much smaller than the time for simulation. We have simulated about 0.6 % of the whole space in 10 days, whereas we could predict 100% of the space in under two hours (6 ANNs, each with a 15 minutes training time and 5 minutes for prediction), with a small error value. Taking the very small NRMSE into consideration we could approximate the whole Pareto Front without the need to simulate the configurations.

C. Results

The following results show the different metrics of the two different runs: one without the actuator's parameters constraints (NoConstraints) and the second one with the constraints on the outputs (WithConstraints). As we can see in the following three figures, the differences are significant and the overhead of implementing constraints for the outputs pays out. We have computed all the results just on the feasible individuals, the ones that satisfy the output thresholds, from each generation.

Looking at Fig. 4 we can see that the coverage of the run with the constraints outperforms the feasible individuals in the run without constraints. This means that up to 80% of the individuals in the run with constraints are nondominated by feasible individuals from the other run; the run without constraints does not favor individuals that are feasible and are of interest for us.

Looking at Fig. 5 it can be seen again that the hypervolume dominated by the run with constraints is almost three times bigger than the run with no constraints. The number of feasible individuals on the Pareto Front Approximation reaches 200 after 10 generations, meaning

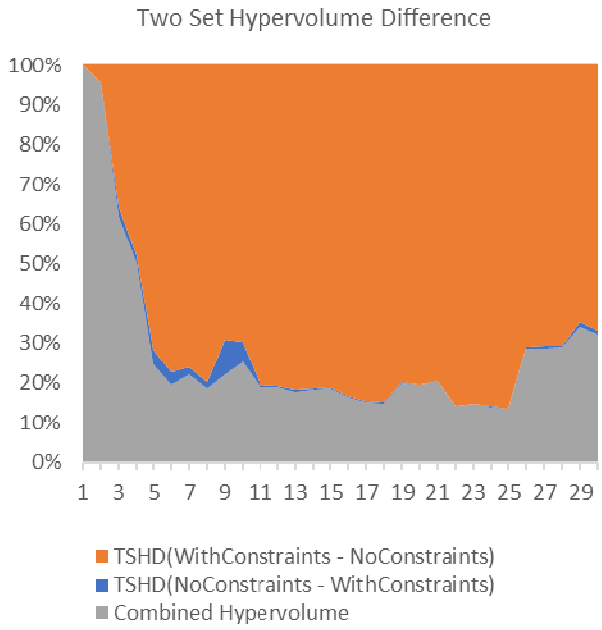

Figure 6. Two Set Hypervolume Difference

that all individuals in a generation belong to the Pareto Front Approximation, whereas on the other run, the Pareto Front Approximation of feasible individuals is consisted of only around 10 individuals. One shortcoming of the NSGA-II selection method can be seen where the hypervolume is decreasing from one generation to another one (e.g. generation 20 to 21). If all individuals belong to the Pareto Front Approximation, the density estimation is taken into consideration. Because this metric is not correlated to the hypervolume, an individual that is more crowded but contributes more to the hypervolume, can be disregarded in favor of another individual that isn't crowded, but contributes less to the hypervolume. The hypervolume of the run without constraints doesn't actually grow because the feasible individuals are added and removed from the generations mainly because they are crowded in some parts of the design space. Again, the selection method of NSGA-II exchanges feasible individuals for individuals from other non-crowded parts of the hyperspace. The values of the dominated hypervolume seem very small, 5.00E-5, but we must take into consideration that we are in an 8-dimensional space. Taking the 8-th root out of 5.00E-5 we get to a value of around 0.29, meaning that on average each dimension is 29% occupied.

Fig. 6 comes as an aggregated conclusion to the previous two. The combined hypervolume is around 30% and is almost the same as the hypervolume created by the feasible individuals from the run with no constraints. The

TABLE V. NEW INPUT PARAMETER RANGES

	Min	Max	Step	Values	Description
I	2	6	0.1	41	current through coils (A)
hcoil	0	4	0.1	41	coil height (cm)
lcoil	0	2	0.1	21	coil length (cm)
lmagnet	0	0.4	0.01	41	magnet length (cm)
liron_move	0	3	0.1	31	iron move length (cm)
liron	0	1.6	0.1	16	iron length (cm)
hiron	0	3	0.1	31	iron height (cm)
hiron2	0	3	0.1	31	iron2 height (cm)

TABLE VI. NRMSE FOR THE BEST-FOUND INDIVIDUALS USING THE RSM

Inputs->Outputs	Mean	Bmax	Vol	Fillfact	Text	Stddev
NRMSE	1.14%	2.43%	0.55%	0.27%	0.21%	1.20%

TABLE VII. EXAMPLES OF INDIVIDUALS FOUND

I1	I2	I3	I4	I5	I6	I7	I8	Mean	Bmax	Volm	Fillfact	Text	Stddev
4.1	3.1	0	0.13	3	0	0.3	0.3	19.411	2.161	3.46E-06	0.7425	68.229	4.7240
2	2.9	0	0.27	0.9	0	0.1	0.5	12.313	2.189	3.74E-06	0.7276	30.219	3.7450
2.8	2.6	0.1	0.13	2.5	0	0.2	0.8	14.407	2.167	3.60E-06	0.7180	41.086	3.8182
3.3	2.7	0.1	0.36	3	0.4	0.4	0.3	15.012	2.134	3.05E-06	0.7240	50.614	3.9426
3.3	3.1	0	0.02	1.3	0.7	1.5	0.5	18.920	2.193	4.11E-06	0.7425	50.934	6.0299

constrained run adds most of the new hypervolume. Looking at Fig. 5 and 6, at generations 8 to 10, we can see that a small increase in the hypervolume of the not constrained run is also present in the TSHD. Looking at these three figures we can conclude that the run with constraints is much better than the other run and our implemented constraints guide the search process towards the feasible individuals.

Our automatic design space exploration methods found individuals that are better than some of the original optimized designs provided by Continental Automotive Sibiu, where the DSE process was probably not automated with state of the art multi-objective meta-heuristics. One of our found individuals is better than one of the original configurations: our design has a smoother curve (standard deviation is smaller), the temperatures are almost the same and the total actuator diameter is smaller by 1mm (1% to 5% decrease in size). This would translate to an actuator configuration with a force that has almost the same mean, but on different positions of the stroke the extreme values are closer to the mean, the temperatures are almost the same, while the actuator is smaller, meaning that fewer materials are needed for the manufacturing process.

Because the NRMSE was very small and the step size seemed quite big (for each parameter we have less than 10 different values), we decided to further decrease the step size from 0.5 to 0.1 (and 0.01 respectively), and try to predict the entire new space shown in Table V. Although we can predict using the RSM method up to 1 million configurations during a minute, the time needed to predict the entire about 700 billion design space is unfeasible, taking more than 450 days. Also, the system requirements for these computations are huge: for the initial 6 million configurations, we saved the predicted values in a 2 GB csv file. We would need a 100.000 bigger (6 million to 700 billion) file to save all the configurations.

In order to cope with the new huge search space, we tried to run a DSE process where we exchanged the actual Matlab and Comsol simulator with the designed RSM module. This way we could evaluate the solutions almost instantly. For doing this, we changed FADSE to allow a Fast Simulator instead of the Server Simulator. All the improvements used until now to accelerate the DSE process, like distributed evaluations and save already simulated configurations in the database, bring huge overhead with the RSM prediction and we did not use them in the Fast Simulator mode. The Fast Simulator used DeepLearning4J [23] in order to use the previously trained ANNs from Keras and Tensorflow. We could run a DSE process using NSGA-II on 30 generations of 20.000 individuals (100 times more than in the initial runs) in a fraction of the time - instead of days; it took just 2 hours for the whole run. After this, we have selected the best 200 Pareto individuals using the NSGA-II selection algorithm. We have run the simulations to be able to compute the NRMSE. The obtained results are shown in Table VI. The NRMSE show us that the predictions are very good and our run using just the RSM is feasible.

Some of the found results belonging to the obtained Pareto hypersurface are presented in Table VII, where I1 to I8 are the input parameters from Table I. These optimized individuals are just a part of the obtained results; we could provide a huge number of configurations that satisfy some constraints in a small amount of time.

Fig. 7 shows a hypervolumes comparison between the last generation of the run with constraints that does the actual simulations (Simulation) and the 200 Pareto individuals selected from the last generation of the RSM run (RSM). The obtained results are notable: the two hypervolumes are almost equal, but the needed time to compute the 200

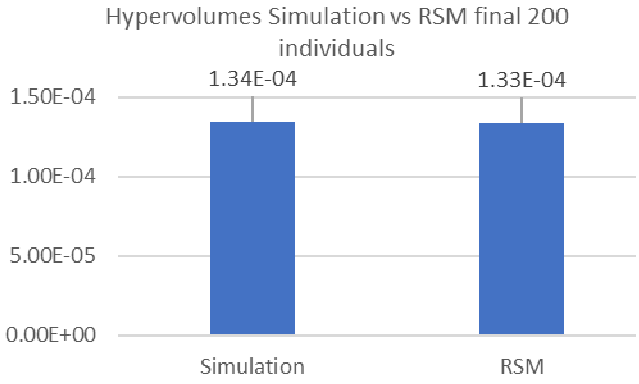

Figure 7. Hypervolume Simulation Run vs RSM Run on 8 objectives on final 200 individuals

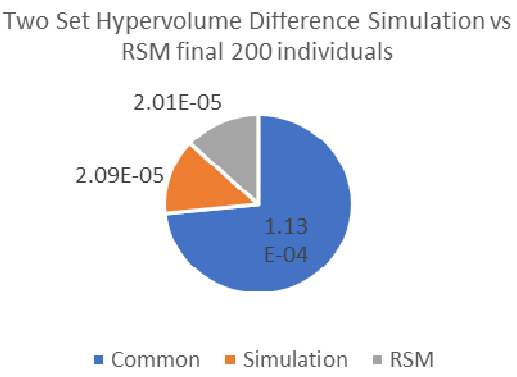

Figure 8. Two Set Hypervolume Difference Simulation Run vs RSM Run on 8 objectives on final 200 individuals

individuals for the RSM run is much lower. More precisely, we needed 2 hours to run the 30 generations and around 3 hours to do the actual simulations for the 200 individuals, compared to around 3 days. The NRMSE in Table VI shows that we could skip the simulations if an error of around 2% is acceptable.

Looking at Fig. 8, we can see that both runs are almost equally good, the main advantage of the RSM run being that the computation time is much smaller, hours instead of day. Both runs bring almost the same proportion of dominated hyperspace.

Looking over the last two figures and the computed NRMSE values in Table VI, we can definitely say that our RSM run provides very good results (error of around 1%) in a much shorter amount of time and that our approach is feasible in extremely huge design spaces (733 billion configurations).

VI. Conclusions and Further Work

Related to our original scientific contributions we pointed out the followings significant aspects: we were able to find significantly better individuals than some of the original values taken into consideration by Continental Automotive Sibiu for the implementation of the magnetic actuator. We also implemented constraints for the outputs, which yield much better results, guiding the DSE process towards feasible configurations. We accelerated the DSE process by the addition of a neural RSM method, that provided very small errors, but could decrease the simulation time by up to a factor of one million (not taking into consideration the training phase).

FADSE tool is a very versatile framework, which can run a DSE process on any simulator, from CPU simulators to actuator simulators, as shown in this paper. We adapted FADSE to be able to run a Matlab and Comsol simulator and run a heuristic search for the magnetic actuator technical particularities, by implementing a specific domain-knowledge for magnetic actuator. These changes allowed us to guide the DSE process towards feasible individuals faster; some of our obtained optimized individuals provided better results than some of the original configurations taken into consideration by the manufacturer.

We accelerate and improve the DSE process by minimizing the number of simulations by leveraging Machine Learning, Design of Experiments and Response Surface Modelling techniques. Particularly, we designed and implemented useful Data Mining and Feature Selection Methods, in order to discover some relationships between inputs and outputs, respectively to reduce the huge outputs space. We managed to decrease the objectives space from 8 to 6 dimensions, thus a 25% decrease.

We provide an original very fast method to approximate the ideal unknown Pareto set for such a NP-hard problem through some RSM neural methods. The proposed method uses a neural network for each of the outputs and by reducing the objective space through Data Mining techniques we were able to decrease the RSM systems complexity (6 instead of 8 ANNs were used). Our initial approach to use DoE with just 256 individuals (full-factorial or random) but also the approach to predict the inputs from the desired outputs yielded some higher errors than predicting the outputs from the inputs.

Because the manual ANNs configuration process yielded very small errors (mostly less than 1%), we were confident to take the DSE process a step further, running NSGA-II algorithm on huge generations of 20.000 individuals, something we have never tried before with our FADSE tool. All our improvements for the acceleration of the optimization processes (distributed evaluations, database access) are now overhead, when we are able to predict up to one million configurations per minute.

Further developments would be the integration of another automatic optimization layer for the RSM. This could optimize each of the ANNs, by automatically varying the number of hidden layers and neurons on each layer, but also the hyper parameters of the ANNs like learning algorithm, learning rate, epoch and batch sizes. This could provide specific neural networks for each of the outputs, some simpler while other more complexes.

We think also about implementing a meta-optimization approach where we run both: a genetic algorithm that runs a part of the simulations and another algorithm that has a trained RSM, which predicts what individuals should be in the next population. The ones predicted will be then simulated.

Because of the small changes needed to implement the RSM layer using ANNs we would like to integrate a RSM for other simulators that we have used. Combining this with the automatic optimization layer for the RSM, we would be able to decrease, while in some extremes cases even bypass the time-consuming simulation part of the DSE process.

VII. References

[1] H. A. Calborean, “Multi-Objective Optimization of Advanced Computer Architectures using Domain- Knowledge,” Ph.D. Thesis, “Lucian Blaga” University of Sibiu, Sibiu, 2011 (Ph.D. Supervisor: Prof. L. Vinţan)

[2] L. Vinţan, R. Chiş, M. A. Ismail, C. Coţofană, “Improving Computing Systems Automatic Multiobjective Optimization Through Meta-Optimization, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems”, Volume: 35, Issue: 7, July 2016, DOI: 10.1109/TCAD.2015.2501299

[3] R. Chiş, L. Vinţan, “Multi-Objective Hardware-Software Co-Optimization for the SNIPER Multi-Core Simulator”, Proceedings of 10th International Conference on Intelligent Computer Communication and Processing, pp. 3-9, Cluj-Napoca, September 2014

[4] G. Palermo, C. Silvano, V. Zaccaria, “ReSPIR: A Response Surface-Based Pareto Iterative Refinement for Application-Specific Design Space Exploration”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume: 28, Issue: 12, Dec. 2009, DOI: 10.1109/TCAD.2009.2028681

[5] A. J. Keane, “Wing Optimization Using Design of Experiment, Response Surface, and Data Fusion Methods”, Journal of Aircraft, Vol. 40, No. 4 (2003), pp. 741-750, http://dx.doi.org/10.2514/2.3153

[6] P. V. Huong, N. N. Binh, “An approach to design embedded systems by multi-objective optimization”, 2012 International Conference on Advanced Technologies for Communications (ATC), 10-12 Oct. 2012, DOI: 10.1109/ATC.2012.6404251

[7] H. Shim, H. Moon, S. Wang, K. Hameyer, “Topology Optimization for Compliance Reduction of Magnetomechanical Systems”, IEEE Transactions on Magnetics 44(3): 346 – 351, April 2008, DOI: 10.1109/TMAG.2007.914670

[8] C. A. Borghi, P. Di Barba, M. Fabbri, A. Savini, “Loney’s Solenoid: A Multi-Ojective Optimization Problem” IEEE Trans. on Magnetics, vol. 35, no. 3, pp. 1706-1709, 1999

[9] H. Calborean, L. Vinţan, “An Automatic Design Space Exploration Framework for Multicore Architecture Optimizations”, Proceedings of the 9th IEEE RoEduNet International Conference, pp. 202-207, Sibiu, June 24-26, 2010

[10] J. J. Durillo, A. J. Nebro, "jMetal: A Java framework for multi-objective optimization," Adv. Eng. Softw., vol. 42, pp. 760–771, 2011
[11] R. Jahr, H. Calborean, L. Vintan, and T. Ungerer, "Boosting Design Space Explorations with Existing or Automatically Learned Knowledge," in Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, 2012, pp. 221–235.
[12] E. Zitzler, L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," Evol. Comput. IEEE Trans. On, vol. 3, no. 4, pp. 257–271, Nov. 1999
[13] E. Zitzler, "Evolutionary algorithms for multiobjective optimization: Methods and applications", vol. 63, Shaker Ithaca, 1999
[14] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations," in International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Nov. 2011
[15] S. Uhrig, B. Shehan, R. Jahr, and T. Ungerer, "A Two-Dimensional Superscalar Processor Architecture," in Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World: 2009, pp. 608–611
[16] "M-Sim: The Multithreaded Simulator." [Online]. Available: http://www.cs.binghamton.edu/~msim/
[17] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," Evol. Comput. IEEE Trans. On, vol. 6, no. 2, pp. 182–197, Apr. 2002
[18] M. Abadi, P. Barham, J. Chen, Z. Chen et al., "TensorFlow: a system for large-scale machine learning", Proceeding OSDI'16 Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation, Pages 265-283, Savannah, GA, USA — November 02 - 04, 2016
[19] C. Francois, "Keras", https://github.com/fchollet/keras, github, 2015
[20] T. J. Santner, B. Williams, and W. Notz, "The Design and Analysis of Computer Experiments.", New York: Springer, 2003
[21] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation", Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37
[22] D. P. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization", 3rd International Conference for Learning Representations, San Diego, 2015, arXiv:1412.6980
[23] Deeplearning4j Development Team. Deeplearning4j: Open-source distributed deep learning for the JVM, Apache Software Foundation License 2.0. http://deeplearning4j.org