U.P.B. Sci. Bull., Series C, Vol. 84, Iss. 4, 2022 ISSN 2286-3540

IMPROVING MODEBI: MULTI-OBJECTIVE OPTIMIZATION BASED ON DIFFERENTIAL EVOLUTION AND BAYESIAN INFERENCE

Cătălin Vișan[1], Octavian Pascu[2], Marius Stănescu[3], Horia Cucu[4], Cristian Diaconu[5], Andi Buzo[6], and Georg Pelz[7]

Since the semiconductors' industry is very competitive, and Artificial Intelligence (AI) can be used to design automated solutions, we aim to speed up the development of the products using AI-based circuit sizing techniques. Automating repetitive tasks is instrumental to reduce the time-to-market of the products while increasing the productivity of the human resource by letting the designers focus on creative tasks. This paper analyzes one of the state-of-the-art circuit sizing algorithms. MODEBI combines a Differential Evolution engine with Gaussian Processes to efficiently find good circuit configurations. However, some internal processes of the algorithm can become computationally expensive. Thus, we are proposing two improved mechanisms that drastically reduces the time budget needed by the algorithm, while maintaining its high performance. This gives more freedom to the circuit designer when defining the problem, and it reduces the overall design time. In addition, we examine the impact of various hyperparameters on overall algorithm performance, and we propose suitable values for them.

Keywords: Multi-objective Optimization, Evolutionary Algorithms, Differential Evolution, Gaussian Processes, Circuit Design

1. Introduction

The status-quo in analog circuit design is that highly skilled engineers are using CAD tools to speed up the design process. These tools are in continuous development and manage to significantly ease the tasks of the designers. However, the engineers still require a vast experience in the domain to be able to properly take advantage of the modern tools. While some of their tasks involve a high degree of creativity and ingenuity that require a skilled human

[1]University "Politehnica" of Bucharest, e-mail: `catalin.visan2304@stud.etti.upb.ro`
[2]University "Politehnica" of Bucharest, e-mail: `octavian.pascu@upb.ro`
[3]University "Politehnica" of Bucharest, e-mail: `adrian.stanescu@upb.ro`
[4]University "Politehnica" of Bucharest, e-mail: `horia.cucu@upb.ro`
[5]Infineon Technologies, e-mail: `cristian-vasile.diaconu@infineon.com`
[6]Infineon Technologies, e-mail: `andi.buzo@infineon.com`
[7]Infineon Technologies, e-mail: `georg.pelz@infineon.com`

operator, others are solely dependent on the experience of the engineer. To reduce the R&D costs of semiconductors products, the human resource has to be used efficiently. So, the creative tasks, such as topology selection and chip architecture, should be the focus of the designers, while trial-and-error tasks such as circuit sizing can be handled with the help of Artificial Intelligence empowered optimizers.

The effort of automating circuit sizing was divided into two directions. First is a qualitative approach, using the prior knowledge of the circuit and its equations, one can build a golden model which can be used afterwards to find the best circuit configuration. The problem of this approach is the deviation of the real circuit from the golden model. In some cases this drawback makes this approach unfeasible. The second one is a quantitative approach based on circuit simulations. In this case, the search is based on particular configurations of the real circuit, so there is no deviation from the "ground truth". Basically, any hyperspace sampling technique can be used to search the optimal configuration of the circuit. Some examples in this regard are Simulated Annealing [1] [2], Particle Swarm Intelligence (PSO) [3] or Bayesian Optimization (BO) [4] [5]. Also, most of the state-of-the-art evolutionary algorithms (EAs) can be used.

In previous work, we compared the most promising 5 evolutionary algorithms [6] [7] resulting in GDE3 [8] being the best for this class of applications. Moreover, an evolutionary algorithm can use surrogate models to speed up the search by reducing the number of simulations [9] [10]. Gaussian Processes (GPs) are the most commonly used surrogate model in circuit sizing automation, not only in combination with EAs but also as support for BO [4]. We proposed Multi-objective Optimization based on Differential Evolution and Bayesian Inference (MODEBI) [11]. It uses a Differential Evolution (DE) engine, just like GDE3, and GPs as surrogate model. Even though its performances are very good, its candidate selection strategy can be quite time-consuming. Thus, in this paper we propose some trade-offs to drastically reduce the time budget needed for each epoch of optimization of MODEBI. In addition, we are tuning two of the hyperparameters of MODEBI, to maximize its performance.

The paper is organized as follows. In Section 2 we describe the concepts used in the rest of the paper, and we explain the selection strategy of MODEBI. In Section 3 we explain several possible issues and how we address them. Then, in Section 4 we present the results of the experiments and discuss the implications. Finally, in Section 5 we draw the conclusions and discuss future plans.

2. Background

The circuit sizing task involves a set of design parameters (DPs) that define the circuit configuration, and a set of circuit responses that define the performance of the circuit. Usually, the responses have to satisfy certain specifications, but some of them have to be further optimized to maximize the circuit's performance. Thus, this formulation corresponds to classical constrained Multi-objective Optimization problem (MOOP) with the DPs being the input vector and the responses having associated constraints. Some responses are also optimization objectives that have to be optimized after the constraints are met.

2.1. Differential Evolution

The most widely used methods for solving MOOPs are based on evolutionary algorithms (EAs). In previous work we have analyzed the performances [6] of State-of-the-art EAs, with a focus on versatility and maintaining population's diversity [7]. Overall, we concluded that GDE3 [8] is the most promising algorithm for circuit sizing tasks. Based on its intrinsic characteristics it produces a steady evolution keeping a high degree of diversity in its population. GDE3 uses a Differential Evolution engine that generates a new offspring solution starting from a parent solution and altering one or more components of its input vector. The algorithm decides based on a stochastic method if each of the components will be altered or not. The alteration is based on other 3 randomly selected solutions from the population. The component of the first one is used as the base term, while the components of the other two are used for the differential term (Equation 1).

$$C_{new}(i) = C_1(i) + F \times (C_2(i) - C_3(i)), \tag{1}$$

where F is a tuning parameter balancing the importance of the differential term.

2.2. Metrics

When it comes to evaluating the generated solutions, it is important to understand if we are comparing feasible solutions (that are meeting the specifications) or unfeasible ones. It is clear that a feasible solution is better than an unfeasible one, but in MOOPs it is not trivial to compare solutions having the same feasibility level. To compare unfeasible solutions, an aggregate metric called Constraint Violation (CV) can be used. It represents the distance between the responses of the solution and the specification as a weighted sum of the distance of each response (Equation 2).

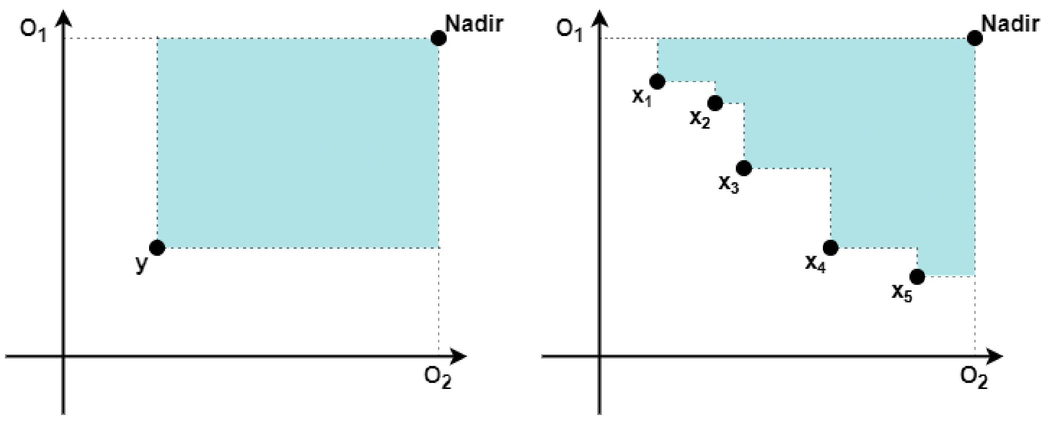
$$CV(x) = \frac{1}{N_c} \sum_{i=1}^{N_c} \frac{cv_i(x)}{\max\limits_{x \in P0}\{cv_i(x)\}}, \tag{2}$$

where N_c is the number of constraints, $cv_i(x)$ is the violation component of the response i of solution x and $P0$ is the initial population.

For feasible solutions, the non-dominance criterion can be used. A solution is dominating another, if all its objectives values are better. However, there is the case when two solutions are mutually non-dominated and additional metrics can be used. A computationally cheap metric is Crowding Distance (CD). It can be used to eliminate the solutions that are negatively impacting the diversity of a group of mutually non-dominating solutions. However, the only indicator that guarantees a fair comparison is the Hypervolume (HV). It represents the area in the hyperspace dominated by a set of solutions and bounded by a Nadir point. Formally, the Hypervolume of a set $S \subset R^d$ is defined [12] in Equation 3.

$$H(S) = \Lambda(\{q \in R^d \mid \exists p \in S : p \leq q \text{ and } q \leq n\}) \tag{3}$$

where $n \in R^d$ is the nadir point, $\Lambda(\cdot)$ is the Lebesgue measure and d is the number of objectives. An example for a two-dimensional objectives space is presented in Figure 1. The HV can be computed for a single solution (Figure 1a) or for a set of solutions (Figure 1b). A larger value of the indicator represents a better set of solutions. The main drawback of HV is the computational cost [12].



(A) Single point (B) Set of points

Figure 1. Hypervolume indicator for 2 objectives

Apart from performance, the diversity of a group of solutions is important for the overall performance of an algorithm. For unfeasible solutions, the CV measures only the performance of a certain solution. For the feasible ones, both CD and HV take into account the diversity, but just for the non-dominated solutions. Thus, we need a metric to compute the diversity of the entire population no matter the nature of the solutions. Distribution Metric (DM) [13] measures both spread and uniformity of the population giving results that correspond to the judgement of a human actor. It uses the projections of the solutions on the axis to compute statistical parameters such as mean, standard deviation, nadir and ideal points, and it computes an aggregate on the entire population. In addition, it is not computationally expensive O(N^2), since it uses just the projections and not the solutions in the hyperspace.

2.3. GDE3

To maintain diversity, GDE3 is always comparing each offspring with its parent. That means that a solution is usually replaced by its offspring. So, if the initial (often randomly selected) population is diverse enough, the populations in the following epochs of optimization are also going to maintain some degree of diversity.

For unfeasible solutions GDE3 uses CV. For feasible solutions it uses the non-dominance criterion, so the selection is more complex. If the solutions are mutually not-dominating, both the parent and the offspring are kept in the population. Due to this mechanism, the population tends to increase. Thus, at the end of the optimization epoch, a pruning process is required. Here, the Pareto dominance is used. The solutions are assigned a rank based on their respective front. Solutions from front 0 are not dominated, the ones from front 1 are dominated only by solutions from front 0 and so on. The solutions from numerically higher fronts are discarded until the population is back to the original size. If a certain front cannot be entirely discarded, CD is used to eliminate the solutions that are negatively impacting the diversity of the certain front.

2.4. MODEBI

To improve the performance of GDE3, we proposed Multi-objective Optimization based on Differential Evolution and Bayesian Inference (MODEBI) [11]. The block diagram of MODEBI algorithm is presented in Figure 2.

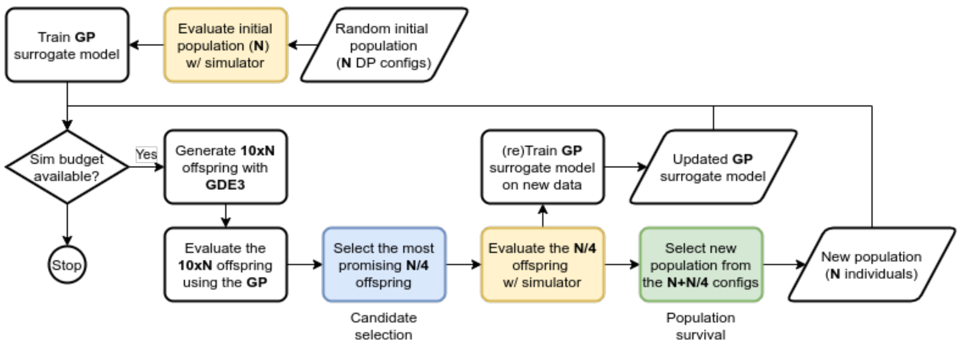

Train GP surrogate model
Evaluate initial population (N) w/ simulator
Random initial population (N DP configs)
Sim budget available?
Yes
Generate 10xN offspring with GDE3
(re)Train GP surrogate model on new data
Updated GP surrogate model
Stop
Evaluate the 10xN offspring using the GP
Select the most promising N/4 offspring
Evaluate the N/4 offspring w/ simulator
Select new population from the N+N/4 configs
New population (N individuals)
Candidate selection
Population survival

FIGURE 2. Multi-objective Optimization based on Differential Evolution and Bayesian Inference [11]

MODEBI uses a Gaussian Process (GP) trained on simulation data to create a surrogate model for the simulator. The GP is further used to predict circuit responses faster than using the simulator and it is retrained periodically (i.e. when new simulation data is available). MODEBI is generating 10 times more candidates than the population size, which are first evaluated using the GPs in order to select the ones that will be sent to the simulator.

Based on the results provided by the GPs, a number of offspring equal to 25% of the population size are going to be selected and evaluated on the real simulator. This ensures a high optimization speed, while maintaining a large population to maintain diversity.

There are two selection strategies. The first one is called Hereditary Selection (HSel) and it takes into account the idea behind GDE3 that the offspring should be compared only with the direct parent. Thus, it selects the best offspring amongst the ones of a certain parent. Then, 25% of the selected offspring that have the highest chance to have better performance than their direct parents are evaluated on the real simulator.

The second selection strategy is called Pool Selection (PSel) and it doesn't use the information about the parents of the offspring. It selects a number of offspring equal to 25% of the population size based solely on their performance. When comparing feasible solution, HV is used. In the case of unfeasible solutions, PSel uses a metric based on CV and DM. They are treated as "virtual" objectives of the optimization, and their HV is used as the performance metric of unfeasible offspring. To balance the importance of CV and DM, a hyperparameter called Distribution Metric Reduction Factor (DM-RF) is used.

After the offspring are selected and evaluated, they will go through a survival process together with the solutions in the current population. Since there are 4 times fewer offspring than parents, a minimum of 75% of the population will remain the same. MODEBI proposes two survival mechanisms. The first one is called Pareto Survival (PSv) and it is similar to the GDE3 approach. Each offspring is compared to its parent and the best one is kept. The second mechanism is called Improved Survival (ISv) and it concatenates the list of offspring with the population resulting an "extended" population. The next population is then built based on HV. If the solutions are unfeasible, the HV is computed on CV and DM, and if the solutions are feasible, the HV is computed on the optimization objectives.

MODEBI offers better results than GDE3, but it is extremely dependent on the predictions of the surrogate model. Thus, it is essential to have the best configuration for the GPs, which means hyperparameter tuning can have an important positive impact. In addition, when using PSel, the performance of the algorithm depends on the value of DM-RF hyperparameter. This is especially important since generally PSel has better performance than HSel. Another important aspect is the time budget. The computational complexity of HV grows exponentially with the number of objectives and the number of considered solutions. Consequently, MODEBI can be computationally unfeasible for problems with a high number of optimization objectives.

3. Methodology

In this section, we are presenting the strategy used for parameter tuning as well as the comparison between the original MODEBI mechanisms for candidate selection and survival, and the trade-offs we propose.

3.1. Parameter Tuning

In terms of candidate selection, MODEBI uses Gaussian Processes (GPs) as a surrogate model to reduce the number of simulations required. Based on the performance of the candidates evaluated during the previous epochs of optimization, the GPs are trained to predict the values of the circuit responses of the offspring in the current epoch. In addition to the plain prediction, a GP is also providing its level of uncertainty. The optimization algorithm uses an *acquisition function* to compute the considered value of the response from the prediction and its level of uncertainty. The acquisition function used by MODEBI is called Lower Confidence Bound (LCB) (Equation 4).

$$LCB = \mu_{pred} - k \times \sigma_{pred} \tag{4}$$

where μ_{pred} is the prediction, σ_{pred} is the level of incertitude. Parameter "k" is weighing the importance of the level of incertitude. Thus, a high value of "k" means a strong exploratory behavior, the algorithm being optimistic about the chance of finding good candidates in unexplored areas. A negative value of "k" means that the algorithm is focus on exploitation rather than exploration. Finally, $k = 0$ means that the algorithm is using the plain prediction as the considered response value.

In order to find the best value for parameter "k", we are running multiple experiments using Hereditary Selection (HSel). As stated, Pool Selection (PSel) has the best performances, but it is also dependent on DM-RF. Tuning two parameters in the same time means performing a grid search that requires a high number of experiments. Since the GPs are working the same with both PSel and HSel, we prefer to isolate this problem by using HSel, because it is not dependent on the Distribution Metric Reduction Factor (DM-RF).

After finding the best value of "k" using the experiments with HSel, "k" will be fixed and used in experiments with PSel in order to tune DM-RF. In the early stages of the optimization, when there are only unfeasible candidates, the selection is made based on Constraint Violation (CV) and Distribution Metric (DM). The CV is calculated for all the offspring, then a few of them are selected solely with this metric. Then, for each of the other offspring, the DM is calculated based on its position in the design parameter's space and the position of the already selected offspring. This process is done iteratively, the DM being calculated again after each new selected offspring. To take into account both DM and CV, the two metrics are used as artificial objectives to compute the Hypervolume (HV). Then the solution with the best HV is selected. However, to find the best trade-off between exploitation and exploration, the impact of CV and DM can be different. Thus, we are using DM-RF as a weighing mechanism. DM-RF is a positive integer. A high value means that the impact of DM is low, so the algorithm is focused on exploitation. A lower value of DM-RF means a stronger exploratory behavior.

3.2. Performance-Timing Trade-off

The new candidate selection and survival policy that were proposed in MODEBI (i.e. PSel and ISv) use HV to compare solutions. The necessary time budget for HV computations is exponentially growing with the number of objectives of the optimization problem, when the algorithm starts finding feasible candidates. This is an important drawback. Thus, we are measuring the impact on the time budget, and we propose a cheaper solution that has similar performance.

When the algorithm starts producing enough feasible candidates, the most promising offspring are selected based on HV. The original approach of PSel is to select offspring iteratively, taking into account the feasible candidates in the current population and the already selected offspring. The problem is that the number of offspring predicted by the GPs to be feasible is growing fast, while the number of feasible candidates in the population is also growing. So, the number of HV computations required becomes important. Moreover, for problems with a high number of objectives, each HV computation is quite expensive (Table 1). Therefore, we propose to reduce the complexity of HV computations by drastically limiting the number of candidates considered from the current population. The disadvantage is that the selected offspring will be diverse compared to each other, but each of them can be close to candidates in the current population. However, this is a negligible performance loss for achieving a feasible timing budget.

The Improved Survival (ISv) mechanism starts from an extended population consisting of the previous population and the selected and simulated offspring. Its goal is to select a number of solutions equal to the population size. If the number of feasible solutions exceeds the population size, it uses an iteratively constructive approach based on the HV computed using the already selected candidates (Algorithm 1). Since the complexity of the HV computation grows with the number of solutions considered, after a number of candidates are selected, each HV computation becomes computationally expensive. To overcome this issue we propose a hybrid mechanism partially inspired by the pruning mechanism of GDE3. A number of the solutions will be selected based on expensive HV computations, ensuring that the best solutions in terms of HV are always selected. Then, when it comes to selecting the best solutions amongst the low performing ones, instead of using expensive HV computations, we use the cheap Crowdin Distance (CD) based pruning. Thus, the timing budget is drastically reduced, while the negative impact on solutions' quality is negligible.

Algorithm 1: Improved Survival (for feasible solutions)

```
Input: Extended population ExPOP of size N+N/4
Input: Number of solutions by expensive computations Nsec
Output: Next population NextPOP of size N
NextPOP = []
for $i = 0; i < Nsec; i++$ do
    for $sol$ in ExPOP do
        computeHV(sol + NextPOP)
    end for
    best_sol = selectBestHV(ExPOP)
    NextPOP.append(best_sol)
    ExPOP.delete(best_sol)
end for
if NextPOP.len $< N$ then
    best_sols = pruneCD(ExPOP, N - NextPOP.len)
    NextPOP.extend(best_sols)
end if
```

4. Results

In this section we present the results obtained during parameter tuning and the findings related to the performance-timing trade-off when MODEBI is searching the feasible solutions' area. The parameter tuning was performed on two voltage regulator topologies, further referenced as Circuit 1 and Circuit 2. Circuit 1 has 27 Design Parameters (DP) and 11 circuit responses with associated constraints, 3 responses being also optimization objectives. For Circuit 1 the optimization process is performed in 8 Operating Corners (OC). Circuit 2 has 8 Design Parameters and 6 circuit responses with associated constraints, all 6 responses being also optimization objectives. For Circuit 2 the optimization process is performed in 10 Operating Corners.

4.1. Parameter Tuning

For parameter tuning we use the following procedure: First, we want to find the best value for parameter "k" using Hereditary Selection (HSel). Then, using the best "k", we want to find the best Distribution Metric Reduction Factor (DM-RF) using Pool Selection (PSel). As stated, PSel showed better results in [11] than HSel, so it is the main target the tuning procedure. However, to isolate the impact of "k" from the impact of DM-RF, we are searching for the best "k" using HSel, since this it is not using DM-RF.

Figure 3, Figure 4, Figure 5 and Figure 6 display the Constraint Violation (CV) of the best individual (solid line) in each epoch of evolution. The CV is calculated based on an aggregated individual having the worst responses amongst the OCs' simulations. The vertical dotted line represents the moment when the first feasible solution is found. Then, the Hypervolume (HV) is computed over all the feasible solutions in the current population. The Nadir point used for HV computations is the actual set of constraints, since it represents the worst possible feasible solution. If the dotted lines are missing, it means the optimization run did not find any feasible solution.

For parameter "k" there were 4 values considered for tuning ($-0.1, 0, 0.1$ and 0.3). It can be seen from Figure 3 that the impact of "k" on Circuit 1 is limited. All four experiments find individuals with similar performance. Thus, the value cannot be chosen based on these results.

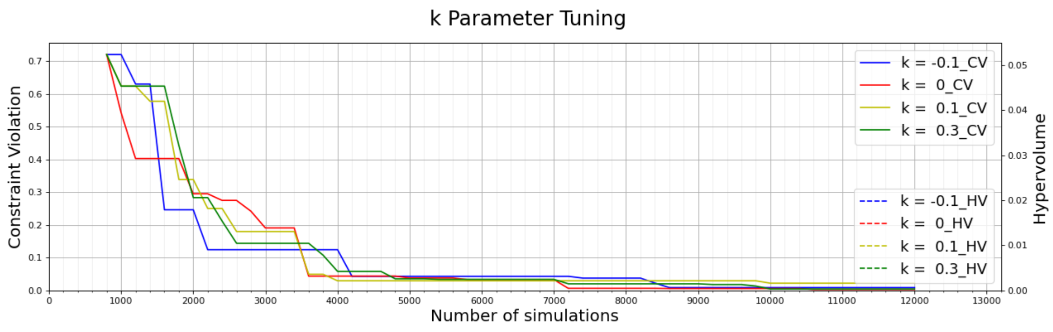


FIGURE 3. Parameter "k" tuning with MODEBI (HSel) on Circuit 1

In Figure 4 the "k" parameter tuning on Circuit 2 is presented. One important aspect for a designer is to find a circuit configuration that meets the specifications as fast as possible. On the other hand, it is important that at the end of the optimization the feasible solutions to approximate best the Pareto front, which means obtaining the biggest HV possible value. In terms

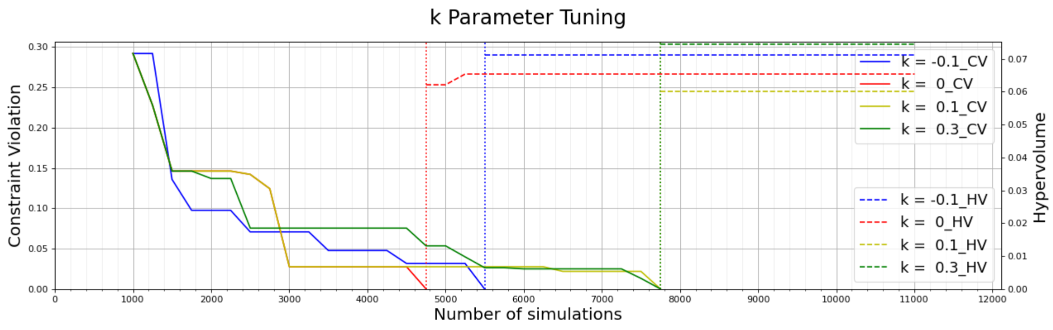


FIGURE 4. Parameter "k" tuning with MODEBI (HSel) on Circuit 2

of convergence rate it is quite clear that the experiment with $k = 0$ obtains better results than the other 3 values. Since, it is the faster in generating feasible solutions. In terms of HV, all the experiments have a flat or nearly flat evolution. After they generate the first feasible solutions, they do not manage to significantly increase the HV by generating better ones in terms of performance or diversity. Thus, it is hard to compare the four optimization runs in this regard, because the quality of the first feasible solutions generated is highly impacted by randomness. Considering the faster convergence rate, we consider the experiment with $k = 0$ to be the most promising one. This means that MODEBI algorithm works best with the prediction from the GP, without using the uncertainty measure provided. So, "k" was fixed to 0 for the following experiments concerning DM-RF tuning.

For DM-RF the results are easier to interpret. In Figure 5, 6 experiments with different DM-RF on Circuit 1 are compared. There might not be a direct correlation between the value of DM-RF and the performance of MODEBI. For example the experiment with DM-RF = 4 is able to find feasible solutions faster than the one with DM-RF = 7. However, this may be just the randomness impact over the evolution, because there is a clear difference in performance in the extreme values. The experiment with DM-RF = 10 performs best, while the experiment with DM-RF = 1 performs worst.

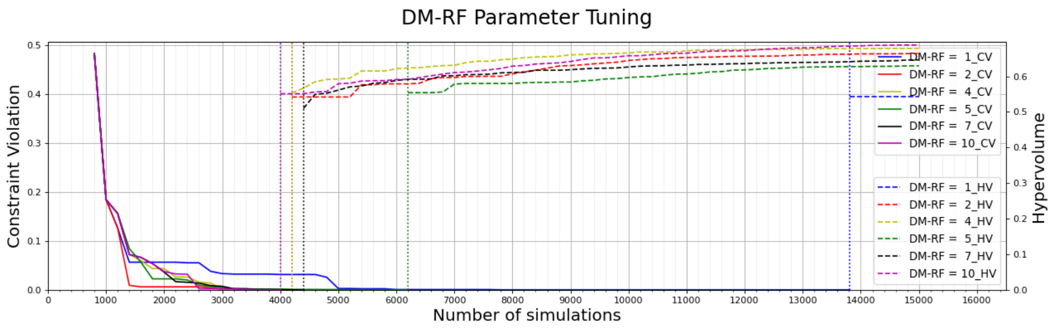


Figure 5. Parameter "DM-RF" tuning with MODEBI (PSel) on Circuit 1

In Figure 6, 4 experiments with different DM-RF on Circuit 2 are compared. Also in this case, the experiment with DM-RF = 10 obtains the best performances. Moreover, it is the only experiment that produces feasible individuals. Selecting higher values (10) for DM-RF means a much lower weight to the Distribution Metric (DM). Prioritizing CV over DM seems to have a good impact on the convergence of the algorithm.

These experiments show the importance of hyperparameters tuning in the context of Artificial Intelligence based circuit sizing methods. While a good value for a hyperparameter can lead to an important speed-up, around 1.6 for "k" and 3.4 for DM-RF in our case. It is important to mention that sometimes (as in Figure 6) the value of the hyperparameter can make the difference between obtaining a circuit configuration that meets the specifications or not. In practice, reducing the design time of the analog blocks lead to smaller time-to-market of semiconductors products. Also, obtaining circuit

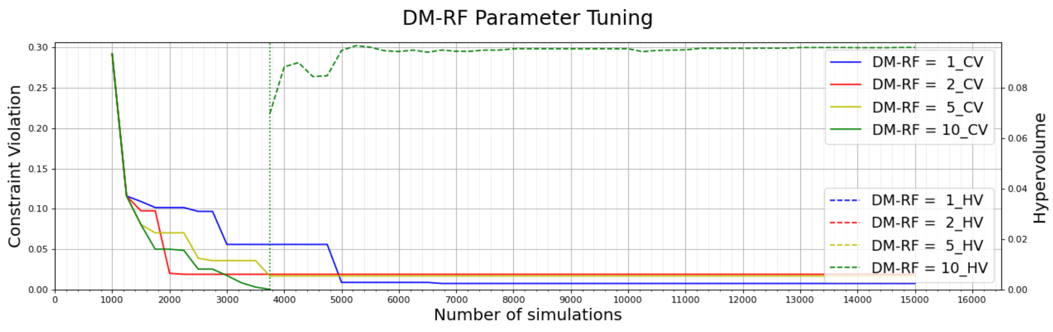


Figure 6. Parameter "DM-RF" tuning with MODEBI Scenario 2 on Circuit 2

configurations that meet the specifications from a single optimization run is essential for achieving a lower design time.

4.2. Performance-Timing Trade-off

During the experiments performed on the two circuits, we have observed an important difference between the timing budgets. While the candidate selection and survival mechanisms were negligible in terms of time budget for the first circuit, they were an important aspect for the second circuit. We observed this difference during the evolution in the hyperspace of feasible solutions. Since the metric used in this case is the Hypervolume, we analyzed its computation time based on the number of individuals involved and the number of objectives of the target circuit. The results are presented in Table 1.

Objectives\Solutions	30	60	80	90	100
3	0.0013	0.0063	0.0103	0.0121	0.0143
4	0.0031	0.0195	0.0312	0.0382	0.0455
5	0.0173	0.1394	0.3600	0.4600	0.5337
6	0.2055	2.2422	6.9643	10.4896	13.1571
7	1.0876	13.7714	55.0963	74.7566	87.6359

Table 1. Hypervolume timing based on the number of objectives and the number of feasible solutions [seconds]

It is quite clear that the computation time of one Hypervolume is growing fast with both the number of objectives and the number of considered solutions. As a numeric example, if the population (100 solutions) includes just feasible solutions, and the GP predicts 500 offspring to be feasible, there will be about 12500 HV computations (101 to 125 solutions each). This means that for Circuit 1 (3 objectives), the time budget of selection has to be around 5 minutes, while for the Circuit 2 (6 objectives) it has to be around 2 days.

This is completely unfeasible, so we proposed two mechanisms to limit the computation time while keeping the original ideas behind MODEBI candidate selection and survival procedures.

For the candidate selection procedure, the original approach of MODEBI was to consider the already selected candidates and the feasible solutions in the current population to select the next candidates. The proposed trade-off is to consider the already selected candidates and just *the best* feasible solutions from the current population. The number of the *best* feasible solutions is variable from 0 to the population size (original approach).

In Table 2 there is the computing time for each solution selected in 3 cases (0, 5 and 10 *best* solutions considered) on Circuit 2. For example, to select the 5th solution in case 0 there will be 4 already selected solutions considered and no solution from the current population, in comparison to 4 already selected solutions and 100 solutions from the current population (original approach). Thus, the total computation time is reduced drastically, from around 2 days to around 15 minutes.

size sol #	0	5	10	**size sol #**	0	5	10
1	0.07	0.9	5.22	14	16.1	42.82	93.97
2	0.1	1.55	7.94	15	21.29	50.07	101.47
3	0.17	2.46	10.93	16	26.04	70.9	121.64
4	0.32	3.84	15.78	17	33.29	80	152.67
5	0.57	6.1	18.81	18	40.17	99.08	184.52
6	1.24	8.19	24.55	19	49.24	116.79	210.36
7	1.79	10.89	25.92	20	57.36	135.33	225.36
8	2.97	11.52	32.87	21	65.05	142.4	233
9	4.26	13.63	35.49	22	72.54	151.84	271.54
10	4.49	20.78	42.98	23	91.55	182.06	345.48
11	5.55	24.37	52.36	24	121.68	236.73	382.7
12	8.58	27.53	62.11	25	130.5	270.28	427.06
13	11.85	35.57	86.03	**Total:**	**766.77**	**1745.63**	**3170.76**

Table 2. Solution selection timing (25 out of ∼500) depending on 0, 5 and 10 considered solutions for Circuit 2 [seconds]

The original survival procedure of MODEBI involves constructing the next population one by one always considering the previously selected candidates. This procedure ensures that the population have the maximum HV possible. However, the time budget needed for Circuit 2 was around 10 hours, which is unfeasible. The proposed trade-off in this case is to select just a fraction of the population with the original approach and the rest using the Crowding Distance (GDE3 approach).

In Table 3 two cases are analyzed: 10/30 solutions selected with the MODEBI approach, the rest being selected using GDE3 approach. The computation time is 2 minutes and 10 minutes respectively. HV100 is the HV obtained with the original MODEBI approach, while HV10/HV30 is the HV of the first 10/30 solutions. During the 12 epochs considered the best 10 solutions represent more than 92% of the HV obtained with the original MODEBI approach, while the best 30 solutions represent more than 97%. So, the performance is good enough just taking the fraction of the population with the original MODEBI approach. In addition, using the GDE3 approach for the rest produces populations that differ just a little in terms of HV from the population produced with the original MODEBI approach. The error is calculated as follows $error[\%] = \frac{HV(original\_pop) - HV(cheap\_pop)}{HV(original\_pop)}$. In all epochs, the error is less than 1%; in 11 out of 12 epochs, the error is less than 0.1%; and in 9 out of 12 epochs, the error is less than 0.01%.

Epoch	1	2	3	4	5	6
Considered solutions	121	118	106	109	117	118
HV10 / HV100 [%]	92.45	91.97	92.78	92.42	92.42	92.86
error [%] (10 sol)	0.0289	0.3670	0.0003	0.0978	0.0018	0.0031
HV30 / HV100 [%]	97.23	97.80	97.79	97.70	97.71	97.71
error [%] (30 sol)	0.0291	0.3673	0.0003	0.0980	0.0021	0.0027
Epoch	7	8	9	10	11	12
Considered solutions	117	120	113	119	119	121
HV10 / HV 100 [%]	92.86	93.39	93.39	93.38	93.38	93.38
error [%] (10 sol)	0.0018	0.0028	0.0021	0.0054	0.0012	0.0027
HV30 / HV 100 [%]	97.71	97.70	97.70	97.70	97.70	97.70
error [%] (30 sol)	0.0010	0.0025	0.0012	0.0081	0.0010	0.0016

TABLE 3. Cheaper survival (10 and 30 solutions chosen based on HV, the rest being chosen based on CD) compared to the original MODEBI survival (100 solutions based on HV)

The two proposed mechanisms extend the usability of MODEBI algorithm, giving the circuit designer more freedom in defining the problem. As described, in the situations where the number of optimization objectives is high, the original candidate selection and survival mechanisms of MODEBI are too computationally expensive. So, the circuit designer has some restrictions while defining the problem (e.g. limited number of objectives). Using the improved versions proposed in this paper, the designer can set the trade-offs such that it fits his needs.

5. Conclusions

The focus of this paper is to extensively present the improvements over the State-of-the-art MODEBI algorithm. It describes the methodology used for parameters tuning and the corresponding results. Also, it introduces and motivates an enhancement to MODEBI which shows a big improvement in computation speed with negligible costs in performance.

The tuning of parameter ”k” shows that MODEBI works best with the direct prediction of its Gaussian Processes (GPs), without taking into account the level of uncertainty of the prediction. Further research in this area should analyze the possibility of integrating different acquisition functions for the GPs. For Distribution Metric Reduction Factor (DM-RF) it seems that MODEBI needs a high value that promotes an exploiting behavior.

The proposed candidate selection mechanism, that does not take into consideration the feasible candidates in the current population, significantly reduces the time budget for each optimization. For sizing tasks with a high number of optimization objectives this makes MODEBI a feasible option. The trade-off is that in later stages of the optimization, the offspring can be close to their parents. The impact of this drawback can be analyzed in future work. Anyhow, it is a low price for making the algorithm usable.

In terms of survival, it seems that using Crowding Distance (CD) to select most of the solutions does not impact the performance considerably. Same as for the candidate selection, the gain in terms of speed-up is tremendous. Thus, the dimension of the fraction of Hypervolume-based chosen solutions has to prioritize speed-up, since the performance loss is negligible.

In future work we plan to design a more robust algorithm that is less dependent on complex solutions' selection mechanisms. This can also help to reduce the number of hyperparameters that require tuning, such as DM-RF. A promising track is to use the surrogate model as a synthetic circuit simulator and to run ”virtual” epochs of optimization to reduce the number of real simulations. Consequently, the optimizer can work exactly as the original version of an evolutionary algorithm, without the need of offspring preselection. In addition, we plan to explore the multiple Hypervolume computation algorithms [14] to further reduce the necessary time budget. Specifically, for current population survival mechanism of MODEBI, it is beneficial to compute the Hypervolume contributions of the solutions in a set [15] and to find the least Hypervolume contributor [16].

Acknowledgement

This work was supported by a grant of the Romanian Ministry of Education and Research, CCCDI - UEFISCDI, project number PN-III-P2-2.1-PTE-2019-0861, within PNCDI III.

REFERENCES

[1] R. Phelps, M. Krasnicki, R. Rutenbar, L. Carley, and J. Hellums, "Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 6, pp. 703–717, 2000.

[2] J. Panerati, D. Sciuto, and G. Beltrame, *Optimization Strategies in Design Space Exploration*, pp. 189–216. 2017.

[3] V. K. Mishra and A. Sengupta, "Mo-pse: Adaptive multi-objective particle swarm optimization based design space exploration in architectural synthesis for application specific processor design," *Advances in Engineering Software*, vol. 67, pp. 111–124, 2014.

[4] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 65, pp. 1954–1967, June 2018.

[5] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for Analog/RF circuit synthesis," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2018.

[6] M. Stanescu, C. Visan, G. Sandu, H. Cucu, C. Diaconu, A. Buzo, and G. Pelz, "Multi-objective optimization algorithms for automated circuit sizing of analog/ mixed-signal circuits," in *2021 International Semiconductor Conference (CAS)*, pp. 1–4, IEEE, 2021.

[7] C. Visan, O. Pascu, M. Stanescu, H. Cucu, C. Diaconu, A. Buzo, and G. Pelz, "Versatility and population diversity of evolutionary algorithms in automated circuit sizing applications," in *2021 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, pp. 1–6, IEEE, Oct. 2021.

[8] S. Kukkonen and J. Lampinen, "GDE3: the third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 443–450 Vol.1, Sept. 2005.

[9] B. Liu, H. Aliakbarian, S. Radiom, G. A. E. Vandenbosch, and G. Gielen, "Efficient multi-objective synthesis for microwave components based on computational intelligence techniques," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 542–548, ACM, 2012.

[10] B. Liu, D. Zhao, P. Reynaert, and G. G. E. Gielen, "Gaspad: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 2, pp. 169–182, 2014.

[11] C. Visan, O. Pascu, M. Stanescu, E.-D. Sandru, C. Diaconu, A. Buzo, G. Pelz, and H. Cucu, "Automated circuit sizing with multi-objective optimization based on differential evolution and bayesian inference," *arXiv*, 2022.

[12] A. P. Guerreiro, C. M. Fonseca, and L. Paquete, "The hypervolume indicator," *ACM Computing Surveys*, vol. 54, pp. 1–42, jul 2021.

[13] K. Zheng, R.-J. Yang, H. Xu, and J. Hu, "A new distribution metric for comparing pareto optimal solutions," *Struct. Multidiscip. Optim.*, vol. 55, pp. 53–62, Jan. 2017.

[14] K. Shang, H. Ishibuchi, L. He, and L. M. Pang, "A survey on the hypervolume indicator in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 1–20, 2021.

[15] K. Bringmann and T. Friedrich, "An efficient algorithm for computing hypervolume contributions," *Evolutionary Computation*, vol. 18, no. 3, pp. 383–402, 2010.

[16] K. Bringmann and T. Friedrich, "Approximating the least hypervolume contributor: Np-hard in general, but fast in practice," *Theoretical Computer Science*, vol. 425, pp. 104–116, 2012.