COMPUTER RESEARCH AND MODELING
2023 VOL. 15 NO. 2 P. 281–300
DOI: 10.20537/2076-7633-2023-15-2-281-300

Ки&М

MATHEMATICAL MODELING AND NUMERICAL SIMULATION

UDC: 519.8

Application of discrete multicriteria optimization methods for the digital predistortion model design

A. Yu. Maslovskiy[1,a], O. Yu. Sumenkov[2,b], D. A. Vorkutov[2,c], S. V. Chukanov[3,d]

[1]Moscow Institute of Physics and Technology,
9 Institutskiy per., Dolgoprudny, Moscow region, 141701, Russia
[2]Sirius University of Science and Technology,
1 Olimpiyskii pr., pgt. Sirius, federal'naya territoriya Sirius, Krasnodarskii krai, 354340, Russia
[3]Federal Research Center "Computer Science and Control" of Russian Academy of Sciences,
44/2 Vavilova st., Moscow, 119333, Russia

E-mail: [a] aleksandr.maslovskiy@phystech.edu, [b] sumenkov.oy@learn.siriusuniversity.ru,
[c] vorkutov.da@learn.siriusuniversity.ru, [d] chukanov47@mail.ru

Received 19.02.2023.
Accepted for publication 23.02.2023.

In this paper, we investigate different alternative ideas for the design of digital predistortion models for radiofrequency power amplifiers. When compared to the greedy search algorithm, these algorithms allow a faster identification of the model parameters combination while still performing reasonably well. For the subsequent implementation, different metrics of model costs and score results in the process of optimization enable us to achieve sparse selections of the model, which balance the model accuracy and model resources (according to the complexity of implementation). The results achieved in the process of simulations show that combinations obtained with explored algorithms show the best performance after a lower number of simulations.

Keywords: digital predistortion, multicriteria optimization, model design, RF PA

Citation: *Computer Research and Modeling*, 2023, vol. 15, no. 2, pp. 281–300.

This research was funded by the Russian Science Foundation (project 21-71-30005), https://rscf.ru/en/project/21-71-30005/.

© 2023 Aleksandr Yu. Maslovskiy, Oleg Yu. Sumenkov, Denis A. Vorkutov, Sergey V. Chukanov
This work is licensed under the Creative Commons Attribution-NoDerivs 3.0 Unported License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nd/3.0/
or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2023 Т. 15 № 2 С. 281–300
DOI: 10.20537/2076-7633-2023-15-2-281-300

Ки&М

МАТЕМАТИЧЕСКИЕ ОСНОВЫ И ЧИСЛЕННЫЕ МЕТОДЫ МОДЕЛИРОВАНИЯ

УДК: 519.8

Применение дискретных методов многокритериальной оптимизации для построения модели цифрового предискажения сигнала усилителя мощности базовой станции

А. Ю. Масловский[1,a], О. Ю. Суменков[2,b], Д. А. Воркутов[2,c], С. В. Чуканов[3,d]

[1]Московский физико-технический институт (национальный исследовательский университет),
Россия, 141701, Московская обл., г. Долгопрудный, Институтский пер., 9
[2]Научно-технологический университет «Сириус»,
Россия, 354340, Краснодарский край, федеральная территория «Сириус»,
пгт. Сириус, Олимпийский пр., д. 1
[3]Федеральный исследовательский центр «Информатика и управление» РАН,
Россия, 119333, г. Москва, ул. Вавилова, 44/2

E-mail: [a] aleksandr.maslovskiy@phystech.edu, [b] sumenkov.oy@learn.siriusuniversity.ru,
[c] vorkutov.da@learn.siriusuniversity.ru, [d] chukanov47@mail.ru

Получено 19.02.2023.
Принято к публикации 23.02.2023.

Осуществление передачи сигналов сотовой связи — одна из ключевых задач современного мира. Для улучшения сигнала передаваемой информации необходимо чтобы сигнал не искажался при усилении мощности на базовой станции сотовой связи. Поставленную задачу можно решать самыми различными способами, однако одним из самых простых решений, которое широко используется в индустрии, является добавление нелинейных искажений, позволяющих линеаризовать работу усилителя и устранять интермодуляционные искажения в областях спектра, не используемых для передачи сигнала. В силу большой нагрузки и работы в реальном времени модель, осуществляющая данные искажения, не должна быть громоздкой и иметь большое количество адаптируемых параметров. В данной статье производится анализ современных работ по теме многокритериальной оптимизации и построения моделей для решения задачи предискажения сигнала при помощи данных методов. В статье показывается, что возможно найти структуру (сохранив производительность) и имеющую меньшее количество используемых ресурсов, быстрее, чем полный перебор по всему словарю из заданных параметров.

Ключевые слова: цифровое предискажение сигнала, многокритериальная оптимизация, построение модели, усилитель мощности

Исследование выполнено за счет гранта Российского научного фонда (проект № 21-71-30005), https://rscf.ru/en/project/21-71-30005/.

© 2023 Александр Юрьевич Масловский, Олег Юрьевич Суменков, Денис Андреевич Воркутов, Сергей Вячеславович Чуканов
Статья доступна по лицензии Creative Commons Attribution-NoDerivs 3.0 Unported License.
Чтобы получить текст лицензии, посетите веб-сайт http://creativecommons.org/licenses/by-nd/3.0/
или отправьте письмо в Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Introduction

At the moment, there are a lot of base stations which are very important for realizing a good wireless connection. For the time being, there is a need to generate additional high-speed signals that will allow us to download large amounts of data quickly. As a result, modern signals should have extremely complex modulation and a wide frequency bandwidth (for example, the third generation partnership project (3GPP) established frequency limits in the 5G new radio (NR) standard, namely, frequency range 1 (FR1: 0.4–7.1 GHz range) and frequency range 2 (FR2: 24–52.6 GHz range) [Hadi, Awais, Raza, 2020]. Wireless communication in the fifth generation is widely assumed to have a greatly increased capacity and communication rate. The envelope amplitude distortion is caused by variations in the input amplitude while amplifying the deviation from a straight line input-output transfer function in the cut-off and saturation regions [Briffa, 1996; Ghannouchi, Hammi, Helaoui, 2015].

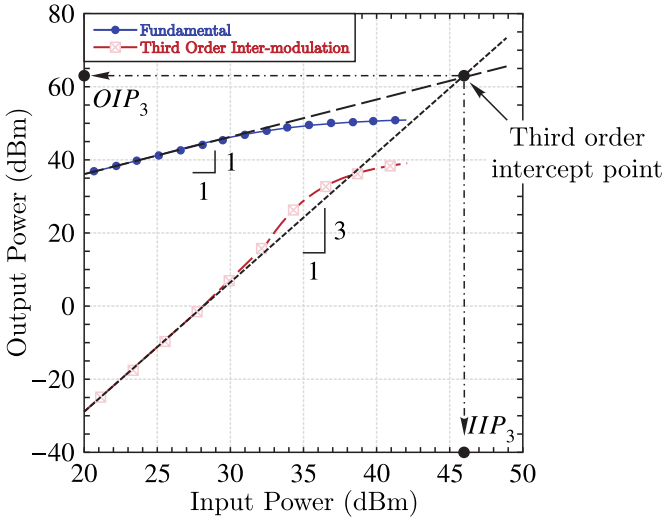

Figure 1. AM-AM amplitude characteristic of input/output signal

There are many approaches aiming to linearize the RF PA and keep a high efficiency at the same time, including feedback linearization [Kang, Sung, Hong, 2020], feedforward linearization [Katz, Wood, Chokola, 2016], analog predistortion and digital predistortion (DPD) [Guan, Zhu, 2014]. DPD is often regarded as the most powerful linearization technology because of its versatility and great performance. The correct modeling and linearization of wideband RF PA is now of great interest to academics and engineers. In this case we used the following approach of using the DPD technique (see Fig. 2).

From the mathematical point of view this approach can have the following formulation:

$$\frac{1}{2}\|PA(DPD(x)+x)-x\|_2^2 \to \min_{DPD(x)}.$$

All signal distortions could be presented as additive changes. So, the optimization problem could be reformulated in the following form:

$$\frac{1}{2}\|DPD_\theta(x)-d\|_2^2 \to \min_w, \tag{1}$$

where $d = y - x$.

A lot of DPD models have been proposed to compensate for the IMD's that are generated in the process of power amplification. For the time being, the most widely used Volterra-based models

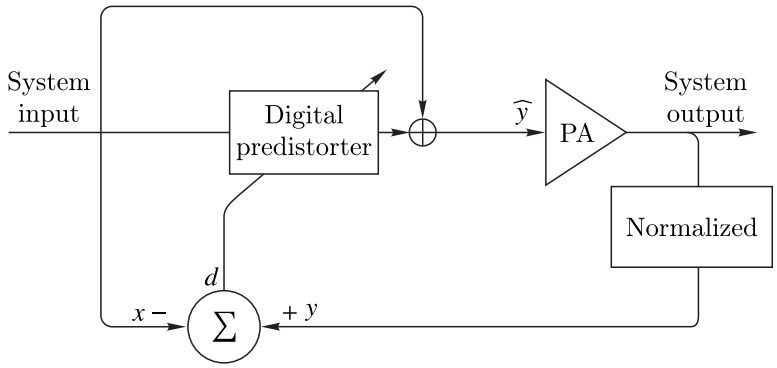

Figure 2. The technique of digital-predistortion

are memory polynomial (MP) [Ding et al., 2004], generalized memory polynomial (GMP) [Morgan et al., 2006], and dynamic deviation reduction Volterra (DDR) model [Zhu, Pedro, Brazil, 2006]. But for wideband signals, a one-layer model doesn't achieve the necessary performance. To accept best performance in hardware mostly it's use Wiener – Hammerstein (WH) model [Gilabert, Montoro, Albertí, 2006]. But in this particular situation, the ideas to stack more layers and use more coefficients to achieve the best results cannot be used because we cannot realize a really nonconvex model with a high count of model coefficients and because we shouldn't spend a lot of power consumption and area resources on a chip under high load in a power amplifier. It is necessary to choose these parameters correctly because of these resources and since the final performance of the model depends on the number of coefficients for various functions. Currently, state-of-the-art model design techniques are based on the matching pursuit approach [Tropp, Gilbert, 2007; Becerra et al., 2018], but this type of algorithm is applicable to a one-layer model, like GMP, because for this model, one can use least squares (LS) estimation. For WH model, LS estimation cannot be used. As a result, in this case there should be used least mean squares (LMS) estimation based on SGD like methods, as a result estimation of every model spend a lot of time. LS estimation cannot be applied with the WH model. As a result, least-mean-squares (LMS) estimation based on SGD-like methods should be used in this circumstance. As a result, estimating each model takes a long time. While LS estimation is faster than LMS, the greedy search process is not applicable in this scenario (the greedy search of the whole model will take a lot of time). Thus, it is difficult to optimize the hyperparameter (HP) search process. For some types of power amplifiers, it is not strictly required to use the whole set of model parameters; as a result, we can minimize the model's implementation area and power consumption in ASIC. As we mentioned above, it depends on the hyperparameters of subfunctions, like an adaptive FIR filter, a polynomial, etc. As stated above, the simulation process for this case is really long, and as a result, the count of possible candidates from whole sets of hyperparameters should be reduced. So that in this article we observe some traditional and state-of-the-art methods [Bardenet, Kegl, 2010; Auger, Hansen, 2012; Ozaki et al., 2022] that can be applicable for this task.

Idea description

As previously said, the primary goal of this research is to obtain great performance with little complexity automatically. This goal value is determined by the hyperparameters of several subfunctions that comprise the entire model.

Thus, the following steps are proposed:

- choose a model which can be applicable for these simulations, choose a possible candidate of dictionary;

- define a appropriate algorithms for HP search;
- search for HP with low complexity and acceptable (high) performance.

To determine the resources of a viable model, we can simplify the problem of model creation to a discrete optimization assignment. To calculate the model's score, we must compare its performance and complexity. To non-dimensionalize the model's score (dB) (*score*), divide it by a value of the same dimension (dB), so that the resulting value is more than zero and does not substantially exceed unity, but still contributes to the result obtained; it must depend on the best model's result (*worst_score*). In this example, the difference was recommended in both the numerator and the denominator $score(\theta) - best\_score$; $worst\_score - best\_score$, because otherwise the model does not change much in the achieved result, and it will have a value that does not alter the estimated model's result. The same idea was implemented for the valuation of the complexity part, but in this case it was decided to use the relax coefficient γ to reduce the influences of changes in the value of the coefficient, because it changes dramatically and mostly influences the score of this set of parameters. To solve this problem it was decided to use the following loss function:

$$J(score, \theta) = \frac{score(\theta) - best\_score}{worst\_score - best\_score} + \gamma \frac{complexity(\theta) - lowest\_complexity}{biggest\_model - lowest\_complexity}, \tag{2}$$

$$J \xrightarrow[\theta \in \mathcal{D}]{} \min, \tag{3}$$

where J is the objective function, θ is the set of hyper-parameters, $\mathcal{D}$ is the dictionary of all applicable candidates of hyper-parameters, *score* is the error reached by model with set θ after N iterations of optimization, *best_score* is the error reached by heavy model after n iterations, *biggest_model* is the size of heavy model, *complexity* is the size of model with current set of hyper-parameters (θ), $worst\_score = -20$ dBc, $lowest\_complexity = 70$ is the complex coefficient due to convenience reasons, γ is regularization coefficient ($\gamma = 0.1$), to reduce the influence of current model size.

Model design algorithms

It is now widely accepted that many product line models are based on a cascade of massive and extremely complex Wiener – Hammerstein [Gilabert, Montoro, Albertí, 2006] models with a massive number of parameters. As a result, there is an obvious problem: reducing the model's complexity while maintaining the original model's performance. Because the model's complexity is determined by the hyperparameters of different subblocks, the process of reducing the complexity could be automated into a multicriteria optimization process. In this study, we have verified state-of-the-art multicriteria optimization methods that were well proven in various deep learning tasks:

- Grid Search,
- Random Search,
- Quasi Monte Carlo Method (QMC),
- Tree-Structured Parzen Estimator (TPE),
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES),
- Nondominated Sorting Genetic Algorithm II (NSGAII).

Grid Search (GS) is a greedy hyperparameter optimization method that ensures an optimal solution in the given configuration space, however, it has an exponential time complexity. Since GS

admits paralleling and simple calculations, it is still used [Shekar, Dagnew, 2019] in low-dimensional configuration spaces.

Random Search is a stochastic optimization method that does not demand a gradient and can be applied even to nondifferentiable functions. It is based on the GS method, but outclasses GS in time performance due to a random choice of parameters instead of parameter enumeration, though its optimal solution guarantee is asymptotic.

Monte Carlo (MC) methods belong to the class of computational algorithms which also use random sampling under-the-hood, but they provide statistical estimates for the objective function by generating random independent samples, in other words, they allow one to simulate desired probability distribution. Many extensions of the MC, such as quasi-MC or Markov chain MC, allow an extension of the boundaries in applications or ensure a better accuracy compared to MC, respectively.

Tree-structured Parzen Estimator (TPE) is another representative of Bayesian-class optimization algorithms that is easy to use to calculate surrogate function as Sequential model-based optimization methods (SMBO). TPE uses mechanisms similar to those of the Gaussian process, but indirectly, usually comes up with the Expected Improvement (EI) acquisition function.

CMA-ES is a state-of-the-art nongradient evolutionary algorithm for optimization on continuous domains, which has been shown to outperform the Gaussian search EDA. Notice that such a gradient-free approach allows nondifferentiable kernels for the GP regression. We do make use of mixtures in [Bardenet, Kegl, 2010], but rather restart the local searches several times, starting from promising places.

Genetic Algorithm (GA) is a search heuristic strategy based on the natural selection principle. GA evolves from the initial population to obtain the fittest candidate for the objective by selection, crossover and mutation mechanisms similar to biological evolution tools. The ability of the strategy to solve NP hard nonlinear optimization problems due to its fundamentals allows the authors [Young et al., 2015] to apply GA to the hyperparameter optimization problem.

In the paper [Ozaki, Nomura, Onishi, 2020], the following recommendations were proposed for the case we are considering — there is an upper limit of sequential calculations, a low level of calculations parallelism, no model conditional parameters: GP-EI (Gaussian Process – Expected improvement), SMAC (Sequential Model-based Algorithm Configuration), TPE, and in the case of implementation of parallel computing, GA and CMA-ES. Researchers in [Shekhar, Bansode, Salim, 2021] compared not only sampling algorithms, but also their implementations in different frameworks — as a result, Optuna with TPE algorithm demonstrated the most balanced results on submitted data sets. In turn, [Balázs et al., 2021] provides experimental results showing that CMA-ES deals well with a global minimum problem.

Grid Search and Random Search

The simplest of the above methods is a grid search when we iterate through all possible sets of hyperparameters and choose the one that provides the smallest value of the objective function. The main flaw is that the increase in hyperparameter dimension causes significant (exponentially) computational complexity growth.

On the other hand, one can improve performance by choosing a set of hyperparameters manually or using both manual search and grid search [LeCun et al., 1998; Larochelle et al., 2007]. However, in this case reproducing results seems to be a real problem. In their work, the authors [Bergstra, Bengio, 2012] went further by suggesting an algorithm based on a random selection of a set of hyperparameters. Namely, this technique for estimating and evidence of random search technique high performance compare to grid search in [Bergstra, Bengio, 2012] for spaces that have high dimensions, but lower effective dimensions — when the desired function is more sensitive to changes in one dimension than the other. In this case it is possible, for example, to approximate a function of two variables by a function

of one variable. The results of [Bergstra, Bengio, 2012] were shown in comparison with [Larochelle et al., 2007] on the MNIST-based (basic, background, rotated and etc) and rectangles data sets.

Quasi Monte Carlo Method

Unlike the previous methods, Bayesian optimization (BO) algorithms [Mockus, Tiesis, Zilinskas, 1978] use the a posteriori knowledge obtained as a result of performing the previous steps to generate the current step. As the name suggests, this method is based on the Bayesian theorem:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$

A distinctive feature of BO is the construction of be a so-called surrogate function approximating the target function in order to make calculations faster and cheaper. The Monte Carlo method is one of the most well-known and extensive tool families in Bayesian optimization. The Monte Carlo method is indeed a universal approach for solving applied computational problems: in engineering, for instance, navigation and localization problems [Enayati et al., 2016; Ortiz, Yu, Li, 2019] or signal processing [Martino, Elvira, 2021]; in natural sciences, for trajectory design in astrophysics [Nickel et al., 2016], thermal effect analysis [Molina, Finzi, 2006]; in machine learning in the hyperparameter optimization problem [Snoek, Larochelle, Adams, 2012; Rakotoarison, Schoenauer, Sebag, 2019; Campbell et al., 2021].

In general, the key steps of MC-family algorithms are:

1) determine possible input variables and their interconnection,

2) generate input elements from the predefined distribution,

3) perform deterministic transformations with sampled elements from p. 2,

4) calculate distribution estimations from p. 3.

Mathematically, the concept of MC integration can be formulated as evaluation of the integral in the form:

$$I = \int f(x)\pi(x)\,dx$$

for an arbitrary pdf $\pi(x)$ with $\int x\pi(x)\,dx$ that does not have a closed solution. Drawing samples x from $\pi(x)$, we can approximate the integral as the average of $f(x)$ — let $x^{i}{}_{i=1}^{N}$ be N distributed independently and identically from $\pi(x)$, and assign the sample mean of f:

$$I_N = \frac{1}{N}\sum_{i=1}^{N} f(x^i).$$

Based on the theory, we get

$$I_n \to I \quad \text{as} \quad N \to \infty.$$

As follows from step 2 of the Monte Carlo algorithm, we select elements from a predetermined random distribution, while in quasi Monte Carlo, quasi-random sequences are used, which are distributed more evenly in the area where the integral is required to be calculated. In other words, quasi-random variables have the property of low discrepancy. This improves the accuracy and slightly the speed of calculations.

Covariance Matrix Adaptation Evolution Strategy

As mentioned above, CMA-ES is a powerful tool for nonlinear optimization, which successfully deals with nonconvex, nondifferentiable problems even with local optima. Moreover, unlike other members of the evolutionary family, CMA-ES does not have, according to [Auger, Hansen, 2012]:

1) large population sizes necessity,

2) early convergence of the population,

3) poor performance on badly scaled objective functions.

Since CMA-ES is an evolution algorithm, it includes principles of natural selection, namely, selection, mutation and crossover to maximize the probability. Then the population of new particles (candidates, individuals and etc) is slightly similar to the Markov model:

$$x_n^{(g+1)} \sim m^{(g)} + \sigma \mathcal{N}\left(0,\, C^{(g)}\right), \quad n = 1, \ldots, \lambda, \tag{4}$$

where $\mathcal{N}$ is a multivariate normal distribution, $x - n^{\text{th}}$ offspring from generation $g+1$, m — mean value of the search distribution at generation g, σ — overall standard deviation (step-size) at generation g, C — covariance matrix at generation g. This is fundamental equation of CMA-ES. In general, it consists of 4 main steps:

- new particles sampling,
- particle selection and recombination (moving the mean),
- adapting the covariance matrix,
- step-size control.

It can be seen from (4), the aim of remaining steps is to calculate m, C, σ for the next step (generation) $g + 1$. The new mean is the weighed average of n chosen points from x:

$$m^{(g+1)} = \omega^T \mathbf{x}, \quad \omega, \mathbf{x} \in \mathbb{R}^n, \quad \sum_{i=1}^{n} \omega_i = 1, \quad \omega_1 \geqslant \omega_2 \geqslant \ldots \geqslant \omega > 0. \tag{5}$$

Then the variance effective selection formula is introduced and equations (5) are rewritten. The covariance matrix can be calculated both empirically with sampled particles

$$C_{emp}^{(g+1)} = \frac{1}{\lambda - 1} \sum_{i=1}^{\lambda} \left(x_i^{g+1} - \frac{1}{\lambda} \sum_{i=1}^{\lambda} x_j^{g+1} \right) \left(x_i^{g+1} - \frac{1}{\lambda} \sum_{i=1}^{\lambda} x_j^{g+1} \right)^T$$

and using the definition:

$$C_{\lambda}^{(g+1)} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left(x_i^{g+1} - m^{(g)} \right) \left(x_i^{g+1} - m^{(g)} \right)^T.$$

Then the weighed selection mechanism transforms the equation above, and as the result of comparing it with the Estimation of Multivariate Normal Algorithm (EMNA) in [Larrañaga, 2002] we have the effective selection mass coefficient μ that has to be updated. Thus, it can be placed here. A full description of the CMA-ES according to [Hansen, 2016] is given in appendix.

Nondominated Sorting Genetic Algorithm II

The genetic algorithm (GA), like the CMA-ES, refers to evolutionary algorithms solving constrained and unconstrained optimization and search problems. So far, this algorithm has been successfully used in various fields: financial markets [Francescomarino et al., 2018], image processing [Sheta, Braik, Aljahdali, 2012], computer science [Alibrahim, Ludwig, 2021], control engineering [Belyaev, Sumenkov, 2021], physics [Tani et al., 2021] and so on.

GA is well-fitted for black-box optimization since it does not require derivatives, its basic working principles provides capability to both parallel calculations and continuous performance improvement GA is not suitable for solving simple problems with an excess of input data, the fitness function must be chosen correctly, because it is calculated repeatedly, it is guaranteed to improve the result of work, but not to achieve an optimal solution.

We have a population whose members, at each step, cross, breed to form new representatives, from which the most suitable for the task at hand is selected. After that everything happens all over again. A more detailed description of GA is shown below.

- Initialization,
- Selection,
- Crossover and mutation,
- Evaluation.

At the initialization step, the initial population is set randomly from the entire search space (random initialization) or predefined (heuristic initialization).

During the selection stage, individuals are specially selected and then paired to reproduce offspring in order to improve their performance through a new combination of genes. This stage is important in order to determine the area of the prostrate where the population is more likely to improve.

One of the essential things that should be designed is the measure of how well the offspring meets the optimality of the solution in the genetic algorithm — the fitness function. It should be calculated fast enough because it is called many times during GA epoch.

Hence, the question whether the population will not degenerate if we constantly select the most suitable offspring, but in GA this problem is solved.

Then there is a crossover stage, when the genes from at least of two parental sequences (chromosomes) are selected and then swapped to generate offspring population of the same size. The way it happened depends on the type of crossover — one point, uniform or some other crossover type.

There is one more important stage — mutation, which serves to introduce diversity into the population, because by selecting individuals with the best fitness function, our population can degenerate. Mutations are random in nature with a relatively low probability. With a high degree of mutation, however, the GA in a sense is simplified into a random search algorithm. In essence, the crossover and mutation phase can be combined in meaning as the generation phase of a new and improved generation.

The next step, not reflected in the list, is when the old generation is replaced by the new one with a better fitness function value.

Now let's move directly from the GA to NSGAII, which was introduced in 2002 by [Deb et al., 2002] to improve and outperform existed multiobjective evolutionary algorithms (MOEA) that have several disadvantages:

- complexity $O\left(MN^3\right)$, where M is number of objectives and N is the population size;

- nonelitism approach is the lower computational speed;
- user determined sharing parameter.

Omitting the details, NSGAII design solved all of these problems, overall complexity was reduced to $O\left(MN^2\right)$, including the introduction of a special approach called crowd-comparison, which replaced the sharing function. This is why this algorithm is one of most standard varieties of evolutionary algorithms and can be successfully applied to the hyperparameter optimization problem [Alibrahim, Ludwig, 2021; Binder et al., 2019; Morales-Hernández, Van Nieuwenhuyse, 2022].

Tree-Structured Parzen Estimator

In many application tasks such as control theory problems, model dynamics simulation, hyperparameters search and adjustment, it is necessary to deal with situations where the object under study f has a black-box structure. In this case calculation of f is expensive in machine time terms, and we often do not have complete information about f (for instance, high-order derivatives).

The model-based approach to tackle such problems has been known for a long time and has been improved in recent years. The essence of model-based algorithms is to describe f using some approximation $\widehat{f}$ called surrogate function, which is much easier to calculate and determines the point at f that should be calculated. More specifically, the task is to find $x^* \leftarrow \underset{x \in X}{\operatorname{argmin}}\, \widehat{f}(x)$. The Expected Improvement (EI) criterion is often used as a functional when solving this problem. More specifically, EI is the expectation of f of model M:

$$EI(x) = \int_{-\infty}^{\infty} \max\left(y^* - y,\, 0\right) p_M(y|x)\, dy,$$

where y^* is some threshold. While GP-based approaches calculate the density $p(y|x)$ directly, the Tree-Parzen Estimator simulates $p(x|y)p(y)$. If we take into account the fact that the configuration space is represented as a generating process in the form of a graph. Thus, X is the tree-structured configuration space, $f\colon X \to \mathbb{R}$ is an objective function to be minimized and x^* has to be found. TPE models $p(x|y)$ with two pdfs in the form:

$$p(x|y) = \begin{cases} l(x) & \text{if} \quad y < y^*, \\ g(x) & \text{if} \quad y \geqslant y^*. \end{cases}$$

As for the objective function (SMBO), namely EI, according to [Bergstra et al., 2011], it transforms into:

$$EI(x) = \int_{-\infty}^{\infty} \max\left(y^* - y,\, 0\right) p_M(y|x)\, dy = \int_{-\infty}^{y^*} \left(y^* - y\right) p_M(y|x)\, dy \propto \left(\gamma + (1-\gamma)\frac{g(x)}{l(x)}\right)^{-1},$$

where $\gamma = p\left(y \leqslant y^*\right)$. Thus, the authors [Ozaki et al., 2022] have developed a TPE algorithm based on the formulas above, a block diagram of which is shown in the appendix.

Experiments

Model Description

As we mentioned above, the Wiener – Hammerstein model was used in this experiment, and specifically an instance of the Wiener – Hammerstein cascade model [Ghannouchi, Hammi, Helaoui, 2015], those structure is presented in Fig. 3, was chosen to refine robustness and enhance performance.

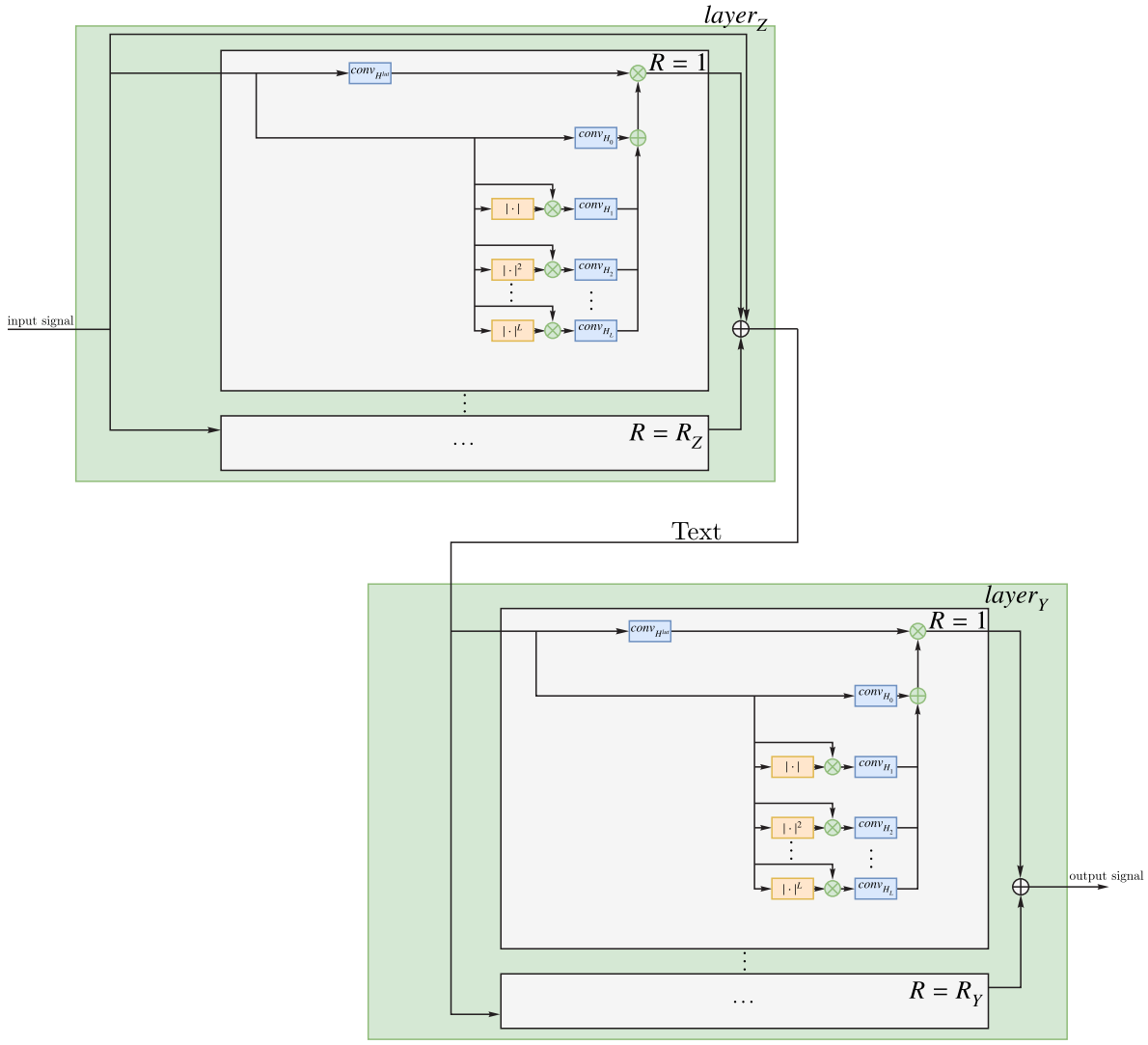

Figure 3. Two-layer block model of the Wiener – Hammerstein type

The following is a formal mathematical model for the case of two layers: the outputs of the first z-layer and the second y-layer are the sum of the results returned from R_z and R_y, respectively, identical blocks. Each of these blocks is described as a combination of convolution, polynomials and lookup table functions applied to the input signal:

$$\text{block}_{H_{CS},H^{lut},C,k}(x) := \text{conv}_{k,H^{lut}}(x) + \sum_{l=0}^{P} C_l \cdot \text{conv}_{k,H_l}\left(x \cdot |x|^l\right), \tag{6}$$

where $H^{lut} \in \mathbb{C}^M$, $H_{CS} \in \mathbb{C}^{K\times L} = \left\{H_l \in \mathbb{C}^K \mid l \in \{1, \ldots, L\}\right\}$ denote weights of convolutions, $C \in \mathbb{C}^P$ are weighting coefficients of gains in lookup table functions, ϕ_p is a polynomial function of arbitrary order applied to the input vector to activate special gain for quantized amplitude of the complex input, $\text{conv}_{k,H}$ is the convolution of the input vector with a vector of weights H and shift $k \in \mathbb{N}$ [Ghannouchi, Hammi, Helaoui, 2015]:

$$\text{conv}_{k,H}(x) := \sum_{n=1}^{N} H_n x_{k-n+1}.$$

Thus, the presented two-layer model is described as follows:

$$\begin{aligned} z_k(x) &= \sum_{r=1}^{R_z} \text{block}_{H_{CS,z},H_z^{lut},C_z,k}(x) + x, \\ y_k(x) &= \sum_{r=1}^{R_y} \text{block}_{H_{CS,y},H_y^{lut},C_y,k}(z(x)). \end{aligned} \tag{7}$$

As a result, we get a computational graph characterized by the following hyperparameters θ (such a set for each layer):

1) M, L is the width of applied convolution,

2) P is the number of polynomial degree,

3) R is the number of blocks in a layer (5 for all experiments);

and having the following set of weights:

$$w := \left(H_z^{lut},\, H_{CS,z},\, H_y^{lut},\, H_{CS,y},\, C_z,\, C_y\right),$$

where

$$\left.\begin{aligned} H_{CS,z} &\in \mathbb{C}^{R_z \times N_z}, \quad H_z^{lut} \in \mathbb{C}^{R_z \times M_z} \\ H_{CS,y} &\in \mathbb{C}^{R_y \times N_y}, \quad H_y^{lut} \in \mathbb{C}^{R_y \times M_y} \end{aligned}\right\} \text{ convolution weights,}$$

$$\left.\begin{aligned} C_z &\in \mathbb{C}^{R_z \times P_z}, \quad C_y \in \mathbb{C}^{R_y \times P_y} \end{aligned}\right\} \text{ polynomial weights.}$$

The total number of model parameters can be calculated as follows: $N = R_z(M_z \times L + P_z) + R_y(M_y \times L + P_y)$.

Optimization task

Closed-loop hyperparameter optimization algorithm

As described in the first part of the chapter, the problem of reducing the complexity of the model while maintaining the performance can be reduced to the problem of multicriterion optimization task "Idea description" using the presented algorithms "Covariance Matrix Adaptation Evolution Strategy", "Quasi Monte Carlo Method", "Nondominated Sorting Genetic Algorithm II", "Tree-Structured Parzen Estimator", using a certain method of changing the error (3). Then, this process can be reduced to the following sequence of actions.

Details of internal process optimization

In the following experiments we use 100 MHz training signals. Because in "Model Description" there are less than 300 coefficients it was decided to use the whole signal for training and validation.

Based on θ, the set of parameters model initializes the whole polynomials as zeros values and ones in the middle coefficients of FIR to get the performance of the model, an optimization process based on (1) problem was started.

According to the previous research, one of the best SGD like optimization methods for DPD task is Adam [Maslovskiy et al., 2021]. So, in these simulations we used Adam optimizer with initial learning rate 0.01 and StepLR scheduler with factor $\gamma = 0.95$ and step size 5.

In the process of internal optimization each model is trained 20 epochs. The batch size is equal to 32 and the length of signal sequance is 1024 (4). The random seed is fixed.

Algorithm 1. Closed-loop hyperparameter optimization algorithm

```
Require: set of HP X
  while J do
    if Termination condition satisfied then
      break
    else
      Choose particular θ ⊂ 𝒟
      Initialization NN model with θ
      Internal optimization process
      Calculating objective function (2) J (score, complexity)
    end if
```

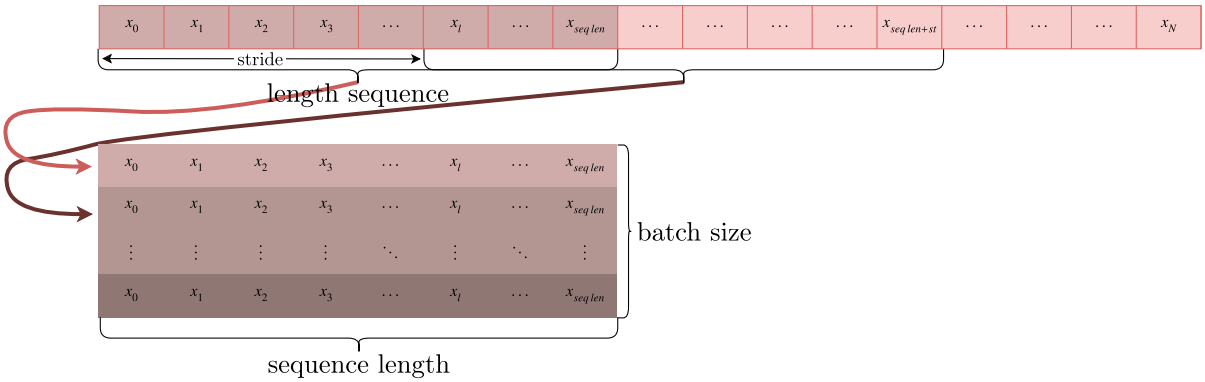

Figure 4. The tensor for model optimization process

To check the performance of the whole model in our simulations we decided to use two types of metrics **Normalized Mean Squared Error (NMSE)** and **Adjacent Channel Leakage Ratio (ACLR)** as a quality measure:

$$NMSE(y, \overline{y}) = 10\log_{10}\left(\frac{\sum\limits_{k=1}^{m}|y_k(x) - \overline{y}_k|^2}{\sum\limits_{k=1}^{m}|x_k|^2}\right) \quad \text{dB}.$$

This metric allows one to check the performance model in the whole spectrum area.

$$ACLR = 10\log_{10}\frac{Power_{A\,jacent\,Chanel}}{Power_{Main\,Channel}}.$$

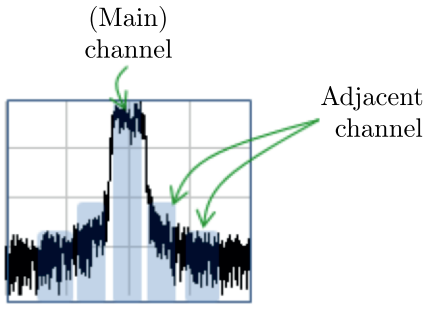

Figure 5. Adjacent Channel Leakage Ratio (ACLR) description

Simulation results

To assess the performance of the proposed models in comparison to existing models, several experimental tests were conducted. Models was created in Python 3.8.12 using an open source machine learning framework PyTorch 1.13.0 and Python. Finally, the model parameters were extracted and trained on a server with a 2.60 GHz Intel Xeon Gold 6240 processor with 72 CPU threads and graphics processing unit Nvidia tesla v100.

After a series of tests it was found that with fixed and the same for all parameters, the variant number five copes with the task best. It is possible to achieve the best result by training the cells of the selected architecture with different hyperparameters to further combine them into an ensemble. This will significantly improve the results of the model. Each cell has 2 selectable parameters, the ensemble has 5 cells, which means the model has 10 hyperparameters. To select the best coefficients, we used a minimization algorithm and a dictionary of 6 AFIR parameters and 4 polynomial parameters used in formulas (6), (7), which describes the plant:

$$D: \begin{cases} [3,\ 5,\ 7,\ 9] \text{ — convolution parameters,} \\ [4,\ 5,\ 6,\ 7,\ 8,\ 9] \text{ — number of polynomial degree,} \\ [-2,\ -1,\ 0,\ 1,\ 2] \text{ — shift convolution.} \end{cases}$$

By enumerating the model parameters from the above dictionary, thereby changing its complexity and affecting its score. Attempting to minimize the objective function J complexity and estimation is constantly changing, but improvement on one parameter does not mean a better solution in general and the best value can be obtained before the enumeration is completed. Each algorithm except Baseline(GridSearch) had the same number of iterations of the external optimizer to fit the parameters. The Figures 6, 7 and 8 show an example of how the TPE algorithm works, with three graphs showing how the result changes with each iteration

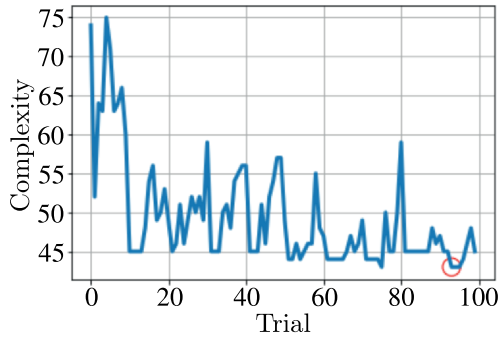

Figure 6. Model complexity variation during the optimization procedure with functional J. Complexity value of chosen model is red circled

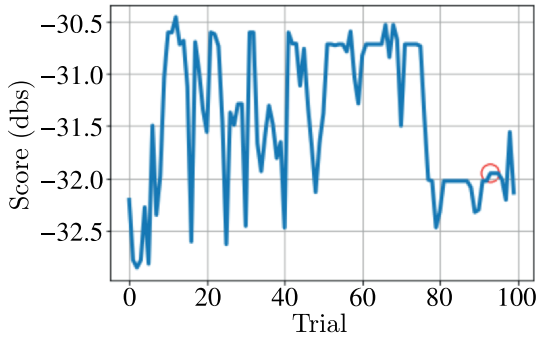

Figure 7. Model performance score on each trial during the optimization procedure with functional J. Score of chosen model is red circled

The models found and trained using the above methods show different rates of learning and score. The convergence plots of the models are shown in Figure 9. When analysing the data, it can be seen that almost every algorithm has had moments of upward and downward movement, which is natural because of the nature of model learning.

The TPE and CMA-ES algorithms, according to the graphs, showed the best results over the entire time, significantly different from the others, while not being affected by sharp jumps and almost always adhering to the downward trend. It also seems from the graph that the TPE algorithm was the winner, having converged to a minimum literally in the last iterations, but this is not entirely true.

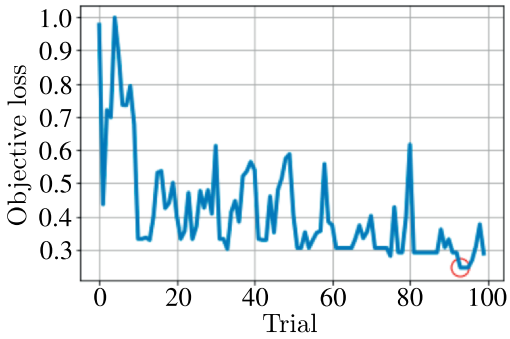

Figure 8. Functional J convergence with the Tree-Structured Parzen Estimator algorithm. The red circle corresponds to the minimum value found — the chosen model

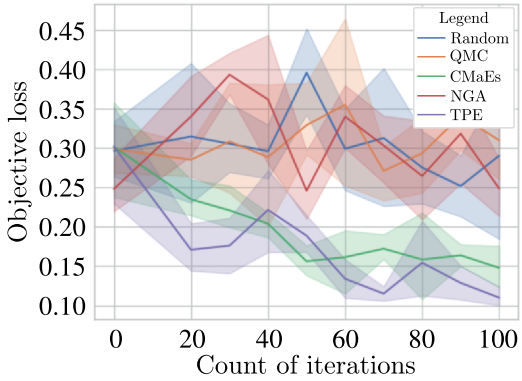

Figure 9. Comparison of different strategies of the optimization process. The x-axis is the number of optimizer iteration, the y-axis is the NMSE value. The figure shows: TPE — Tree-structured Parzen Estimator algorithm, CMA-E — Covariance Matrix Adaptation Evolution Strategy, Random Search, QMC — Quasi Monte Carlo Method, GA — Nondominated Sorting Genetic Algorithm II. It shows an estimate of the central tendency and a confidence interval for that estimate

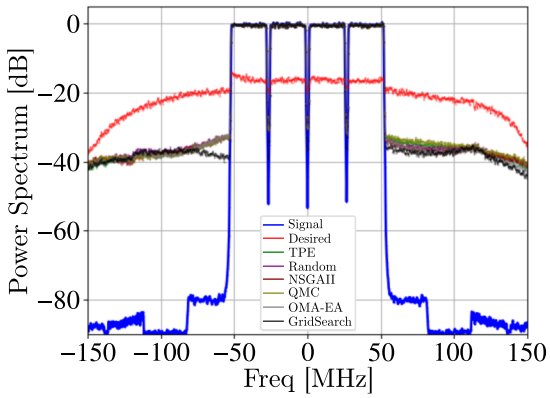

Figure 10. Normalized power spectral densities signals (PSDs). The original signal is highlighted in blue, while the other colors are the result of the models. The figure shows: TPE — Tree-structured Parzen Estimator algorithm, CMA-ES — Covariance Matrix Adaptation Evolution Strategy, Random search algorithm, QMC — Quasi Monte Carlo Method, NSGAII — Nondominated Sorting Genetic Algorithm II, Grid Search algorithm

The Tree-Structured Parzen Estimator algorithm did indeed perform very well, but the minimum object function reached CMA-ES, as will be shown numerically later.

QMC, Random and NGAII algorithms showed quite frequent changes in the up and down trend of the target function. The Random algorithm shows the largest scatter in neighboring iterations of the optimizer.

And figure 10 shows comparison of the normalized power spectral density of the original signal and the signal obtained after the trained models have been run.

The process of finding model hyperparameters, the models were trained for 20 epochs each to speed up the loss function calculations. The data on NMSE, the optimization time, the complexity of the found models, and the final value of the entered functional of the best found model for the given parameters are presented in Table 1. As expected, the best result was achieved by a complete overrun of the parameters. Grid Search algorithm was able to excel in functional value, showing a better MNSE value while maintaining a fairly low complexity. However, this approach takes significantly longer to find the optimal parameters than any other algorithm. At the same time, the Covariance

Table 1. Simulation results with 20 epoch model training

Algorithm	Complexity	Objective loss	NMSE dBc (20 Epoch)	t, sec
Random	92	0.1064	−27.419	10598.6
TPE	90	0.0987	−27.259	8001.0
NSGAII	106	0.192	−27.324	7651.5
QMC	98	0.1465	−27.254	7609.3
CMA-ES	88	0.08	−27.491	7644.9
Baseline(GridSearch)	94	0.0797	−27.518	18263.7

Table 2. Simulation results with 500 epoch model training

Algorithm	NMSE dBc $_{(500\ \text{Epoch})}$	ACLR $_{(500\ \text{Epoch})}$		Complexity
Random	−27.99	−36.221	−37.554	92
TPE	−28.253	−36.617	−36.689	90
NSGAII	−28.254	−36.623	−37.969	106
QMC	−27.828	−36.5074	−36.067	98
CMA-ES	−28.205	−36.461	−37.811	88
Baseline(GridSearch)	−28.456	−36.621	−36.948	94

matrix adaptation evolution strategy showed a similar MNSE model score, less complexity, but took significantly less time to select the hyperparameters, almost identical to the best time.

This is enough to evaluate the performance of the resulting model, but not enough to fully utilize the model in real-world conditions. Therefore, the best models found by different algorithms were additionally trained for 500 epochs, in order to evaluate the efficiency change. The evaluation of these models using NMSE and ACLR is presented in Table 2. When you increase the number of epochs, the results are not much different. The QMC algorithm still shows the worst NMSE result, GridSearch the best. However, the gain of the CMA-ES algorithm was less than that of the other models, which may be a consequence of the fact that this model has the lowest complexity, which means that the gain with increasing the number of epochs of training is also the lowest.

Based on the ACLR data from the simulations, we can conclude that the model built genetic algorithm NSGAII showed the best performance in signal suppression at low and high frequencies, as well as the greatest efficiency gain with increasing learning epochs. This may also be directly related to the complexity of the model, which is the maximum, among the others.

As a result, we were able to achieve a reduction in the complexity of the model, as well as a decrease in the search time for hyperparameters, with little decrease in performance.

Conclusion

In this article various method for solving the problem of synthesizing a digital pre-distortion model of a power amplifier signal were reviewed and discussed.

According to the theoretical and practical results, the most suitable algorithms for multi-criteria optimization are TPE and CMA-ES. The concept of a closed-loop optimization process based on the Bayesian optimization method of the Tree-Structured Parzen Estimator (TPE) algorithm shows the best median curve of convergence during the whole optimization process for different simulations. Another concept of closed loop multi-criteria optimization, based on the Covariance matrix adaptation evolution strategy (CMA-ES) algorithm, shows the best simulation results on the full convergence of the found parameters of the model.

This strategy was compared with greedy algorithms like grid search and the non-dominated sorting genetic algorithm with elite strategy (NSGAII). This type of algorithm allows for finding the best parameters for the explained structure but requires significant computational resources because of its structure.

Lack of balance can lead to models that are either too simple or too complicated, which can result in incorrect predictions and inefficient usage of valuable resources. A simple model may not have enough sophistication to fully interpret the data, while an overly complex model may expend excessive resources in the training process, resulting in high computational overheads. Therefore, the best algorithms, which are presented in this article, can be used to solve optimization problems with decreasing model resources.

References

Alibrahim H., Ludwig S. A. Hyperparameter optimization: comparing genetic algorithm against grid search and Bayesian optimization // 2021 IEEE Congress on Evolutionary Computation (CEC). — 2021. — P. 1551–1559.

Auger A., Hansen N. Tutorial CMA-ES: evolution strategies and covariance matrix adaptation // Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation. — 2012. — P. 827–848.

Balázs C., van Beekveld M., Caron S., Dillon A. S. HA comparison of optimisation algorithms for high-dimensional particle and astrophysics applications // Journal of High Energy Physics. — 2021. — Vol. 2021, No. 5.

Bardenet R., Kégl B. Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm // International Conference on Machine Learning. — 2010.

Becerra J., Madero-Ayora M., Reina-Tosina J., Crespo-Cadenas C., Garcia-Frias J., Arce G. A doubly orthogonal matching pursuit algorithm for sparse predistortion of power amplifiers // IEEE Microwave and Wireless Components Letters. — 2018. — Vol. 28, No. 8. — P. 726–728.

Belyaev A. S., Sumenkov O. Yu. Hybrid control algorithm based on LQR and genetic algorithm for active support weight compensation system // 20th IFAC Conference on Technology, Culture, and International Stability TECIS 2021. — 2021. — Vol. 54, No. 13. — P. 431–436.

Bergstra J., Bardenet R., Bengio Y., Kégl B. Algorithms for hyper-parameter optimization // Advances in Neural Information Processing Systems. — 2011. — Vol. 24.

Bergstra J., Bengio Y. Random search for hyper-parameter optimization // J. Mach. Learn. Res. — 2012. — Vol. 13. — P. 281–305.

Binder M., Moosbauer J., Thomas J., Bischl B. Multi-objective hyperparameter tuning and feature selection using filter ensembles // Proceedings of the 2020 Genetic and Evolutionary Computation Conference. — 2019.

Briffa M. A. Linearization of RF power amplifiers // Victoria University. — 1996.

Campbell A., Chen W., Stimper V., Hernandez-Labato J. M., Zhang Y. A gradient based strategy for Hamiltonian Monte Carlo hyperparameter optimization // Proceedings of the 38th International Conference on Machine Learning. — 2021. — Vol. 139. — P. 1238–1248.

Deb K., Pratap A., Agarwal S., Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation. — 2002. — Vol. 6, No. 2. — P. 182–197.

Ding L., Zhou G. T., Morgan D. R., Ma Z., Kenney J. S., Kim J., Giardina C. R. A robust digital baseband predistorter constructed using memory polynomials // IEEE Transactions on communications. — 2004. — Vol. 52, No. 1. — P. 159–165.

Enayati J., Sarhadi P., Rad M. P., Zarini M. Monte Carlo simulation method for behavior analysis of an autonomous underwater vehicle // Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment — 2016. — Vol. 230, No. 3. — P. 481–490.

Francescomarino C. D., Dumas M., Federici M., Ghidini C., Maggi F. M., Rizzi W., Simonetto L. Genetic algorithms for hyperparameter optimization in predictive business process monitoring // Information Systems. — 2018. — Vol. 74. — P. 67–83.

Ghannouchi F. M., Hammi O., Helaoui M. Behavioral modeling and predistortion of wideband wireless transmitters. — John Wiley & Sons, 2015.

Gilabert P., Montoro G., Albertí E. On the Wiener and Hammerstein models for power amplifier predistortion // CICYT-TIC2002-04084-C03-01. — 2006. — Vol. 2. — P. 1–4.

Guan L., Zhu A. Green communications: Digital predistortion for wideband RF power amplifiers // IEEE Microwave Magazine. — 2014. — Vol. 15, No. 7. — P. 84–99.

Hadi M. U., Awais M., Raza M. Multiband 5G NR-over-fiber system using analog front haul // 2020 International Topical Meeting on Microwave Photonics (MWP). — 2020. — P. 136–139.

Hansen N. The CMA evolution strategy: A tutorial. — 2016. — 39 p.

Kang S., Sung E. T., Hong S. Dynamic feedback linearizer of RF CMOS power amplifier // IEEE Microwave and Wireless Components Letters. — 2018. — Vol. 28, No. 10. — P. 915–917.

Katz A., Wood J., Chokola D. The evolution of PA linearization: From classic feedforward and feedback through analog and digital predistortion // IEEE Microwave Magazine. — 2016. — Vol. 17, No. 2. — P. 32–40.

Larochelle H., Erhan D., Courville A., Bergstra J., Bengio Y. An empirical evaluation of deep architectures on problems with many factors of variation // Proceedings of the 24th International Conference on Machine Learning. — 2007. — P. 473–480.

Larrañaga P. A review on estimation of distribution algorithms // Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. — Boston, MA: Springer US, 2002. — P. 57–100.

LeCun Y., Bottou L., Orr G. B., Müller K.-R. Efficient BackProp // Neural Networks: Tricks of the Trade. — 1998. — P. 9–50.

Martino L., Elvira V. Compressed Monte Carlo with application in particle filtering // Information Sciences. — 2021. — Vol. 553. — P. 331–352.

Maslovskiy A., Pasechnyuk D., Gasnikov A., Anikin A., Rogozin A., Gornov A., Antonov L., Vlasov R., Nikolaeva A., Begicheva M. Non-convex optimization in digital pre-distortion of the signal // Mathematical Optimization Theory and Operations Research: Recent Trends. — 2021. — P. 54–70.

Mockus J., Tiesis V., Zilinskas A. The application of Bayesian methods for seeking the extremum // Towards Global Optimization. — 1978. — Vol. 2, No. 2. — P. 117–129.

Molina M., Finzi A. E. Monte Carlo techniques in thermal analysis — design margins determination using reduced models and experimental data // Journal of Aerospace. — 2016. — Vol. 115. — P. 304–311.

Morales-Hernández A., Van Nieuwenhuyse I. A survey on multi-objective hyperparameter optimization algorithms for machine learning // Artificial Intelligence Review. — 2022.

Morgan D. R., Ma Z., Kim J., Zierdt M. G., Pastalan J. A generalized memory polynomial model for digital predistortion of RF power amplifiers // IEEE Transactions on signal processing. — 2006. — Vol. 54, No. 10. — P. 3852–3860.

Nickel C., Lebois R., Lutz S., Dichmann D. J., Parker J. J. K. Monte Carlo analysis as a trajectory design driver for the TESS mission. — 2016.

Ortiz S., Yu W., Li X. Autonomous navigation in unknown environments using robust SLAM // IECON 2019 — 45th Annual Conference of the IEEE Industrial Electronics Society. — 2019. — Vol. 1. — P. 5590–5595.

Ozaki Y., Nomura M., Onishi M. Hyperparameter optimization method in machine learning: Overview and features // IEICE Transactions on Information and Systems. — 2020. — Vol. J.103-D, No. 9. — P. 615–631.

Ozaki Y., Tanigaki Y., Watanabe S., Nomura M., Onishi M. Multiobjective tree-structured parzen estimator // J. Artif. Int. Res. — 2022. — Vol. 73. — 42 p.

Rakotoarison H., Schoenauer M., Sebag M. Automated machine learning with Monte Carlo tree search // Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. — 2019. — P. 3296–3303.

Shekar B. H., Dagnew G. Grid search-based hyperparameter tuning and classification of microarray cancer data // 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP). — 2019. — P. 1–8.

Shekhar S., Bansode A., Salim A. A comparative study of hyper-parameter optimization tools // 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). — 2021. — P. 1–6.

Sheta Alaa., Braik M., Aljahdali S. Genetic algorithms: A tool for image segmentation // Proceedings of 2012 International Conference on Multimedia Computing and Systems, ICMCS 2012. — 2012. — P. 84–90.

Snoek J., Larochelle H., Adams R. P. Practical Bayesian optimization of machine learning algorithms // Proceedings of the 25th International Conference on Neural Information Processing Systems. — 2012. — Vol. 2. — P. 2951–2959.

Tani L., Rand D., Leelken C., Kadastik M. Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics // The European Physical Journal C. — 2021. — Vol. 81, No. 2. — P. 67–83.

Tropp J. A., Gilbert A. C. Signal recovery from random measurements via orthogonal matching pursuit // IEEE Transactions on Information Theory. — 2007. — Vol. 53, No. 12. — P. 4655–4666.

Young S., Rose D., Karnowski T., Lim S.-H., Patton R. Optimizing deep learning hyper-parameters through an evolutionary algorithm // Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. — 2015. — P. 1–5.

Zhu A., Pedro J. C., Brazil T. J. Dynamic deviation reduction-based Volterra behavioral modeling of RF power amplifiers // IEEE Transactions on microwave theory and techniques. — 2006. — Vol. 54, No. 12. — P. 4323–4332.

Appendix

Algorithm 2. Tree-structured parzen estimator

Require:
$D = \left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(k)}, y^{(k)}\right)$: observations
$n_t \in \mathbb{N}$: number of iterations
$n_c \in \mathbb{N}$: number of candidates
$\gamma \in (0, 1)$: quantile
for $t = 1, \ldots, n_t$ do
  $D_l = (x, y) \in D | y$ is included in the best — $[\gamma |D|]$ objective values in D
  $D_g = D \setminus D_t$
  repeat
    $i = i \in [1, n]$ such that x_i is active and x_i^* has not been sampled
    construct $l(x_i)$ with $x_i | (x, y) \in D_l$ and $g(x_i)$ with $x_i | (x, y) \in D_g$
    $C_i = x_i^{(j)} \sim l(x_i) | j = 1, \ldots, n_c$
    $x_i^* = \underset{x_i \in C_i}{\operatorname{argmax}} \frac{l(x_i)}{g(x_i)}$
  until all active parameters have been sampled
  $D = D \cup (x^*, f(x^*))$
end for
return x with the minimum y value in D

Algorithm 3. Covariance matrix adaptation evolution strategy algorithm

Set parameters

Set parameters λ, $\omega_{i=1,\ldots,\lambda}$, c_σ, d_σ, c_c, c_1 and c_μ

Initialization

Set evolution paths $p_\sigma = 0$, $p_c = 0$, covariance matrix $C = I$, and $g = 0$

Choose distribution mean $\mathbf{m} \in \mathbb{R}^n$ and step-size $\sigma > 0 \in \mathbb{R}$

Until termination criterion met, $g = g + 1$

Sample new population of search points, for $k = 1, \ldots, \lambda$

$$z_k \sim \mathcal{N}(0, I),$$
$$y_k = BDz_k \sim \mathcal{N}(0, C),$$
$$x_k = m + \sigma y_k \sim \mathcal{N}\left(m, \sigma^2 C\right)$$

Selection and recombination

$$\langle y \rangle_w = \sum_{i=1}^{\mu} \omega_i \cdot y_{i:\lambda}, \quad \text{where} \quad \sum_{i=1}^{\mu} \omega_i = 1, \ \omega_i > 0, \quad \text{for} \quad i = 1, \ldots, \mu,$$

$$m \leftarrow m + c_m \sigma \langle y \rangle_w, \quad \text{equals} \quad \sum_{i=1}^{\mu} \omega_i \cdot x_{i:\lambda} \quad \text{if} \quad c_m = 1$$

Step-size control

$$p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_{eff}}\, C^{-1/2} \langle y \rangle_w,$$
$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0, I)\|} - 1\right)\right)$$

Covariance matrix adaptation

$$p_c \leftarrow (1 - c_c) p_c + h_\sigma \sqrt{c_c (2 - c_c) \mu_{eff}} \langle y \rangle_w,$$
$$\omega_i^\circ = \omega_i \times \left(1 \quad \text{if} \quad \omega_i \geqslant 0 \quad \text{else} \quad n / \|C^{-1/2} y_{i:\lambda}\|^2\right),$$
$$C \leftarrow \Bigg(1 + \underbrace{c_1 \delta(h_\sigma) - c_1 - c_\mu \sum \omega_j}_{\text{usually equals to 0}}\Bigg) C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\lambda} \omega_i^\circ \cdot y_{i:\lambda} \cdot y_{i:\lambda}^T$$