A Method Using Generative Adversarial Networks for Robustness Optimization

NICLAS FELDKAMP, SOEREN BERGMANN, FLORIAN CONRAD, and STEFFEN STRASSBURGER, Information Technology in Production and Logistics, TU Ilmenau

The evaluation of robustness is an important goal within simulation-based analysis, especially in production and logistics systems. Robustness refers to setting controllable factors of a system in such a way that variance in the uncontrollable factors (noise) has minimal effect on a given output. In this paper, we present an approach for optimizing robustness based on deep generative models, a special method of deep learning. We propose a method consisting of two Generative Adversarial Networks (GANs) to generate optimized experiment plans for the decision factors and the noise factors in a competitive, turn-based game. In a case study, the proposed method is tested and compared to traditional methods for robustness analysis including Taguchi method and Response Surface Method.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; **Machine learning**;

Additional Key Words and Phrases: Machine learning, deep learning, robustness optimization, Generative Adversarial Networks

ACM Reference format:
Niclas Feldkamp, Soeren Bergmann, Florian Conrad, and Steffen Strassburger. 2022. A Method Using Generative Adversarial Networks for Robustness Optimization. *ACM Trans. Model. Comput. Simul.* 32, 2, Article 12 (February 2022), 22 pages.
http://doi.org/10.1145/3503511

1 INTRODUCTION

Discrete event simulation is an established methodology for investigating the dynamic behavior of complex manufacturing and logistics systems. In this context, finding robust configurations is often a critical issue. Robustness refers to setting the controllable factors (decision factors) of a system in such a way that variance through uncontrollable noise on a given output is as low as possible [41]. To measure the robustness of system configurations, the quality-loss-formulas according to the commonly known Taguchi method can be used [46]. The Taguchi method provides certain experiment designs, but other approaches are also available. One of the key strengths of the Taguchi method is the ability to calculate and compare the robustness in a large and comprehensive

Authors' address: N. Feldkamp, S. Bergmann, F. Conrad, and S. Strassburger, Information Technology in Production and Logistics, Technische Universität Ilmenau, P.O. Box 100 565, 98684 Ilmenau, Germany; emails: {niclas.feldkamp, soeren.bergmann, florian.conrad, steffen.strassburger}@tu-ilmenau.de.

1049-3301/2022/02-ART12 $15.00
http://doi.org/10.1145/3503511

way. This gives the opportunity to not only explore the robust system configurations, but also allows for exploration of contributors against system robustness. Optimization algorithms on the other hand aim straight forward for the most robust solution, while Taguchi's loss function allows to investigate custom made experiment plans.

Robustness analysis and optimization in general is an ongoing topic in simulation methodology research [3, 12, 35]. Especially **Data Farming (DF)**, which is a concept that comprises broad scale simulation experimentation and the use of algorithmically enhanced analysis [11, 43], enables the finding of robust solutions due to a deep dive into the systems behavior. However, a main challenge is to have suitable experiment plans for decision and noise factors that offer a balance between good coverage of the input space and manageable runtime, while also maintaining decent experiment design quality criteria to avoid analysis bias [12]. In this paper, we examine how to use a machine learning method called **Generative Adversarial Networks (GANs)** to create experiment plans and to further use this approach for robustness optimization. In a GAN, two neural networks contest with each other in a zero-sum game framework. A generative network generates candidates while a discriminative network evaluates them. This technique has originally been used to generate images that look at least superficially authentic to human observers [16]. In fact, we propose a concept for robustness optimization that includes the use of two GANs, that are interconnected in a competitive, turn-based contest. This is done by using GANs for the generation of experiment plans. To measure the robustness of a decision configuration, the loss is calculated against a distinct number of noise configurations. We therefore build the optimization process so that the one GAN generates single decision configurations, while the other GAN generates complete experiment plans with multiple configurations for the noise. Both GANs then aim to subsequentially improve their generated configurations turn by turn. This means one GAN aims to improve the robustness of system by optimizing the decision factors, while the other GAN aims to worsen the robustness by generating noise factor configurations that are as disrupting as possible. For the calculation of the robustness, experiments (in terms of current decision configuration vs. current noise configurations) are carried out after each turn using a simulation model. The prerequisite for the robustness optimization process using two GANs is to evaluate how to use a GAN for the generation of experiment plans. Therefore, as a follow-up to [2], we provide an in-depth discussion of the generation of high quality experiment plans using GANs in this paper. This includes a discussion of multiple approaches for the training of the GAN as well as the determination of requirements and hyperparameter evaluation. Furthermore, we provide a more detailed presentation of our concept for robustness optimization using GANs regarding a more in-depth description of process, workflow, and details of technical implementation concerning the interconnection of simulator, AI, and optimization-algorithm. We also extended the case study by a comparison of robustness analysis using **Response Surface Method (RSM)**. The remainder of this paper is structured as follows. In Section 2 we introduce the related work on robustness analysis as well as some required basics on the design of experiments in this context. We also introduce the related work regarding Generative Adversarial Networks. Section 3 presents the concept for GAN-based robustness optimization. In Section 3.1, we give some preliminary considerations on generating experiment designs using a GAN, followed by introduction of the actual process of robust optimization using two GANs in Section 3.2. In Section 4, we present a case study for a proof of concept in which we compare our approach to traditional methods for robustness evaluation, including the original Taguchi method and the Response Surface Method. This is then concluded by summarizing remarks and a discussion of future work in Section 5.

Table 1. Loss Functions for Different Purposes [36]

Type of Quality Loss Function	Formula
Nominal-the-best	$\bar{L} = k[\sigma^2 + (\bar{y} - \tau)^2]$
Smaller-the-better	$\bar{L} = k\left[\sum_{i=1}^{n} {y_i}^2\right]$
Larger-the-better	$\bar{L} = k\left[\sum_{i=1}^{n} (1/{y_i}^2)\right]/n$

2 RELATED WORK

2.1 Revisiting Taguchi's Loss Function and Experiment Design Principles for Robustness Analysis

Robustness refers to setting the controllable parameters of a system in such a way that variance in uncontrollable noise has a minimal effect on a given output [41]. Variation through noise can emerge from various sources. For example, fluctuations in customer demand can lead to variation in the mixture of jobs that are dispatched in the system [17]. This effect can increase dramatically especially at the lower tiers of the supply chain, which is commonly known as the bullwhip effect [25]. Since we can control every factor including the noise factors in a simulation model, we can use simulation to assess the robustness of a system. A popular method for the measurement of robustness is the Taguchi method. Genichi Taguchi, who originally came from a quality engineering background, developed a methodology to assess decision alternatives not only based on their outcome value, but also on the variability around that outcome against noise. Taguchi created formulas (shown in Table 1) that calculate the quality loss caused by deviation from a desired value. Here, $\bar{L}$ is the average loss for a given system configuration over all noise configurations n, k is a fixed constant called the quality loss coefficient, $\bar{y}$ and σ^2 represent the mean and variance of an output value for each system configuration against all noise configurations (pure y representing the output for each experiment). In this context, a configuration represents one distinct set of factor values. The experiments that need to be conducted then ultimately consist of different configurations tested against each other. Depending on the characteristics of the desired output parameter, different loss functions have to be applied [36].

The nominal-the-best loss function aims to reduce the variability around a desired output target value τ and therefore sanctions output values above and below this target, for example the required output voltage of an electrical circuit. The smaller-the-better function aims to minimize a given output, for example cost, stress, or energy consumption. On the other hand, the larger-the-better loss function is used to maximize an output value like reliability, strength, or efficiency. The result of the loss function is usually converted to a logarithmic ratio ($-10log_{10}(\bar{L})$). This is called the signal-to-noise ratio and represents the magnitude of the mean compared to its variance in industrial processes according to Taguchi [36]. Besides the collection of formulas, the Taguchi method also provides experiment design patterns. Taguchi's work on robustness analysis had a great impact and also a lot of controversies among statisticians [32]. A more in-depth review on the subject of Taguchi method and other robust design concepts can be found in Park et al. [34]. Besides the Taguchi method, other optimization methods for system robustness evaluation exist, for example the Response Surface Method [31]. A great comprehensive review on robustness analysis can be found in [39, 41]. The Taguchi method recommends using crossed orthogonal fields, that are oriented towards minimizing experimental effort and therefore offer only a small degree of freedom. For example, having three factors with 4 levels each, the Taguchi method would recommend the L9-Array, which offers 9 rows of experiments. While criticism on the Taguchi method particularly

focused the aspects of experiment design and experiment plans [31], the Response Surface Method was proposed as an alternative for robustness optimization. RSM resembles a more traditional approach to experiment analysis. In a two-stage approach, statistical models are to be fitted onto the data to determine an optimal value. For simulation-based optimization, modern techniques like Genetic Algorithms and Swarm optimization are discussed to replace RSM [7, 33]. However, for robustness optimization, Taguchi's formulas, also in combination with other methods, still offer a great way to explore, visualize, and optimize the systems robustness by dividing the model's factors into decision and noise [38, 41]. This is why they still today are the subject of current research [19, 28], especially in the simulation community [12, 22, 44]. The strength of simulation-based robustness optimization using Taguchi's formulas is that a broad range of possible noise and decision configurations can be evaluated when large-scale experiment plans are used. This enables an in-depth analysis and investigation of the system behavior and the potential contributors and roadblocks to the system robustness. Several publications aim to improve on the experiment design aspects of the Taguchi method [12, 43, 47]. While Taguchi's formulas offer a great way to calculate a value of robustness, simulation-based approaches obviously can conduct many more experiments in a given time span than what could be achieved in a real-world experimental setup. New approaches that can make use of Taguchi's formulas but are also able to increase the information gain by leveraging the efficiency of a simulation model. A current approach in [12] combines the method of data farming with the Taguchi method. This allows to calculate the Taguchi values over a much larger collection of output data then the traditional orthogonal arrays that have been used for the Taguchi method. Data farming is a methodology for using a simulation model as data generator and an efficient experiment design next to automated output analysis to maximize data yield and therefore information gain [9, 21]. The farming metaphor describes how the data output can be maximized by efficient experiment designs, like a farmer that cultivates his land to maximize his crop yield [42]. This is, in particular, made possible by new approaches in the design of experiments that are highly efficient and manage a balance between broad scale factor combinations and manageable data volume [24]. For these kinds of experiments, having experiment plans that are of high quality is very important. The experiment plan defines the number of experiments that need to be conducted as well as the value of each factor for each individual experiment. In addition to the manageable number of individual experiments in the plan, the main criteria for quality concerns usually are orthogonality, the space-filling properties, and balance.

Orthogonality can be measured by simply calculating the maximum pairwise correlation between factor values, represented by the columns of an experiment design. Ideally, the pairwise correlation should be zero, but Cioppa and Lucas showed that in modern, efficient experiment design methods, it can be beneficial regarding the space filling properties to accept a minimal amount of correlation [5]. The space-filling properties represent the ability of an experiment design to sufficiently cover the input space defined by the factors. This input space coverage can be measured by the so-called discrepancy. Discrepancy measures the uniformity of the distribution of configurations within the input space [45]. A rather computation-intensive way to calculate this is the L_∞-discrepancy [10]. In this work, we use the L_2-discrepancy [5], which is more commonly used in practice to measure the space filling properties of a design [18]. The balance property accounts for the balanced distribution of factor values. This means that the values of a factor should occur with equal frequency. This can be measured by calculating the maximum imbalance measure Δ_k from all factors [49].

Efficient experiment designs are necessary since a full factorial design (n^k) that covers all mathematically possible combinations of factor values is usually not practical if more than just a few factors and factor values need to be addressed. In incomplete experiment designs, multiple factors are altered simultaneously between two subsequent experiments. In such cases it is important that

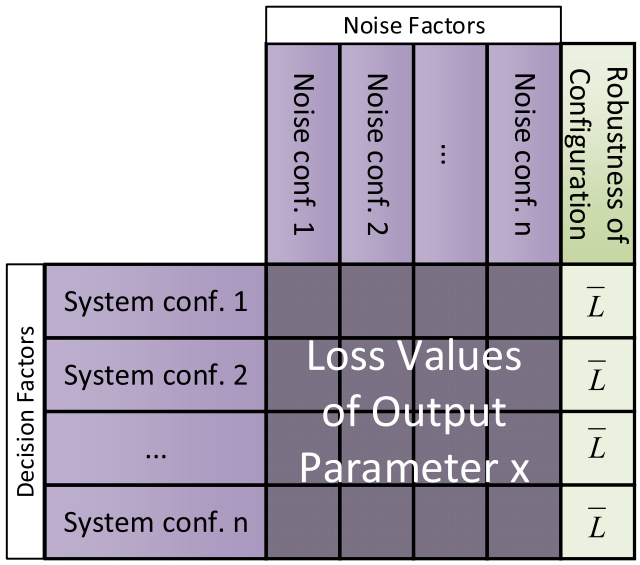

Fig. 1. Matrix of crossed arrays for robustness evaluation [12].

the quality criteria described above are kept at sufficient levels [5, 20, 40]. Many different design methods are available in the literature. Commonly used designs are the Latin hypercubes because of their good general-purpose applicability for exploring complex simulation models [5, 13, 40], as well as more optimized **nearly orthogonal Latin hypercubes (NOLH)** [15, 23]. For the robustness analysis based on data farming, one experiment plan for mapping the decision factors (system configurations) and one for representing the noise factors are generated and crossed. After the experiments are conducted, the results can be arranged into a matrix-like table that shows how each system configuration performs for each noise factor configuration, see Figure 1.

In the matrix, each cell represents one simulation experiment and can be filled with its corresponding robustness value of a selected output parameter x according to the chosen loss function. For each row, we can determine the average loss representing the robustness of each system configuration. We can then build various analysis methods around this robustness calculation. For example, in [12], a two-step process built around visually aided analysis was developed. The first step is grouping system configurations into classes of similar robustness groups using classification algorithms. Simulation experiments in the same class belong to system configurations that have similar values in their corresponding robustness dimensions. In the second step, different algorithms are usable for further analysis. The goal here is to provide insights on which input factor values lead to distinct classes, preferably those with decent robustness values. Knowing how to set input factor values accordingly to get to a desired class label allows conclusions on how to make the system robust.

2.2 Generative Adversarial Networks

Machine learning, as a sub-discipline of **artificial intelligence (AI)**, is currently a very popular topic. In particular, deep learning is used in a wide variety of applications, such as classifying images, audio waveforms containing speech, or natural language text corpora processing. The most popular and striking successes in deep learning have been achieved in regards to discriminative models. Those usually map a high-dimensional, rich sensory input to a class label [15].

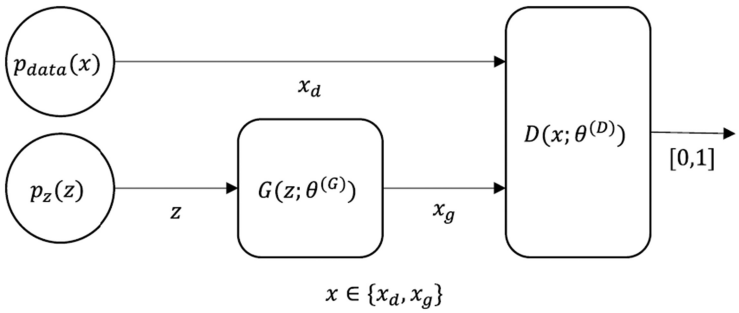

Fig. 2. Schematic basic structure of a GAN.

In contrast to classifying real world data, a comparatively new idea is to use neural networks to generate artificial data. This can be used for example for the generation of photorealistic images or videos [23]. One type of network used for this purpose is the Generative Adversarial Network (GAN), which is a class of machine learning systems invented by Goodfellow [16] in 2014. Given a training set, this technique learns to generate new data based on the features of the training set. A GAN (see Figure 2) consists of two main components, the actual generator (G) and a discriminator (D).

Both networks are implemented as feedforward networks. The basic idea is that the generative network generates candidates (χ_g), based on a noise vector (z), while the discriminative network evaluates them. Evaluation means that the discriminative network distinguishes candidates produced by the generator from the true data distribution (χ_d). The generator trains based on whether it succeeds in fooling the discriminator. The discriminator, on the other hand, is trained to distinguish generated from real data. The training therefore winds up in turn-by-turn zero-sum game, until the generator has improved in such a way that the discriminator can no longer distinguish between real and generated data samples. The results that have been achieved with GANs so far are impressive. For example, in 2018, a GAN-generated image was auctioned for $432,500 [6]. In the spring of 2019, researchers demonstrated a GAN-based tool that could generate videos of a person based on a single photograph [50].

3 CONCEPT FOR USING GENERATIVE ADVERSARIAL NETWORKS FOR ROBUSTNESS OPTIMIZATION

As already mentioned in Section 2.1, the strength of simulation-based robustness optimization using Taguchi's formulas is that a broad range of possible noise and decision configurations can be evaluated. This allows to explore the best decision configurations, and also the worst noise configurations. This is achieved by combining two experiment plans in a crossed array, one for the decision factors and one for the noise factors. As for every experiment-based analysis, the experiment plans need to meet certain quality criteria as explained in the previous section. These criteria ensure that the analysis of results is not biased in any way and that the input space is decently covered. In fact, creating experiment plans that meet those requirements and yet are efficient enough to be conducted in adequate runtime can be challenging. The idea of our approach is, that we implement a system of two GANs that can create experiment plans: One GAN generates single decision configurations that are evaluated against a complete experiment plan of noise configurations, generated by the second GAN. The robustness is evaluated using Taguchi's formulas in a turn-by-turn game. As a starting point, both GANs generate their experiments based on experiment design quality criteria as explained in the previous section to initially cover the

possible input space. During the optimization process, the training database is then altered based on the calculated robustness value, so that the first GAN is getting better and better at generating robust configurations over time, while the second GAN learns how to come up with disruptive noise configurations that can bring down the system robustness. Therefore, we first introduce some preliminary considerations on how to use a GAN for generating experiment plans that expose the desired features in Section 3.1. The actual process for robustness optimization using two GANs that compete against each other is then introduced in Section 3.2.

3.1 Preliminary Considerations on Using GANs for Experiment Plan Generation

Before using GANs for robustness optimization, we need to ensure that they are able to generate experiment plans in the first place. The underlying objective function that the GAN is generating samples of therefore should be based on some form of quality criteria for experiment plans. In Section 3.1.1, we first show that GANs are able to generate experiment plans by learning from a given distribution, which is the traditional way of using GANs. In Section 3.1.2, we present an improvement that we call directed GAN, where the database of the discriminator GAN is changing dynamically in order to further improve the training. When using GANs for the actual robustness optimization (Section 3.2), the objective function is then switched to a robustness measure. Because two GANs are competing against each other, one GAN aims to minimize the robustness measure and the other aims to maximize it.

3.1.1 Generating Experiments Using Traditional GAN-Training. Of particular interest here was to investigate if a GAN can generate single experiment configurations or even complete experiment plans. We tested both cases, as is described in the following section. In order to generate experiment configurations, a GAN must be able to generate points from the corresponding input space. When the input space is defined properly, meaning that lower and upper limits for each factor has been set, the input space represents the distribution P_{data} (see Figure 2 in previous section) that the GAN needs to learn from. Since the GAN has no insight into this original distribution, because its task is to learn it from examples, we need to generate m learning examples in the first step, where m is the size of a mini batch for training. A mini batch is a small sample drawn from the underlying data distribution. These examples are generated in such a way that every value stems from a uniform distribution between the lower and upper limit for each corresponding factor. This distribution can be either continuous or discrete, depending on the scale of the factor. These factor values are then combined to a complete sample configuration. Afterwards, the actual training can start. Figure 3 shows the schematic architecture for GAN-based experiment generation.

The goal of the generator is to generate configurations that are not predictable as synthetically generated by the discriminator. The discriminator, on the other hand, aims to distinguish original from generated configurations. Generator and discriminator compete in a contest, which is played over different rounds. In each round, the generator is trained first, and then the discriminator is trained on the current outputs. The generator always determines its error against the current version of the discriminator.

Individual configurations can then be combined to a complete experiment plan afterwards. This has some obvious drawbacks regarding experiment plan quality criteria that are calculated over multiple factors. So besides from generating individual configurations, generating a complete experiment plan at once is also possible in a fairly similar way. However, both methods are needed for our concept for robustness optimization that we present in the next section. For generating complete experiment plans, the original distribution does not represent the input space of a single factor but rather the set of all possible experiment plans in the entire input space of all factors. Therefore, the data to be generated corresponds to an experiment plan, represented by a matrix

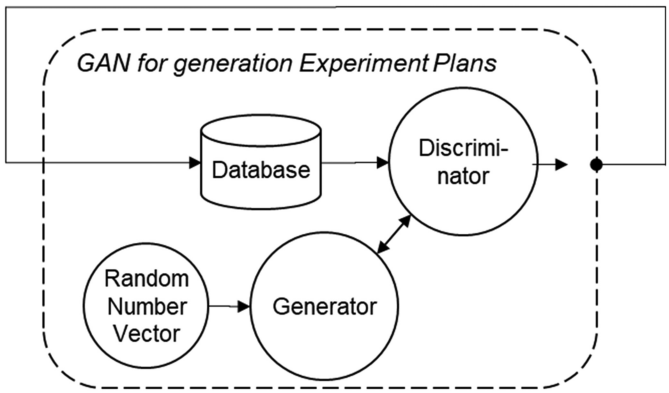

Fig. 3. Schematic architecture of GAN-based experiment plan generation.

M_{data} in the form of $n_{\mathfrak{x}}x$ K ($n_{\mathfrak{x}}$ rows/design points and K columns/factors), which needs to be transformed into a vector having $n_{\mathfrak{x}}x$ K entries for using it as an input for the discriminator. This can easily be done by simply linking together the rows of the matrix vertically. This means, that the output layer of the generator needs to include $n_{\mathfrak{x}}x$ K neurons as well in order to generate a similar vector. This vector can then be reconstructed into a matrix afterwards. Figure 4 shows the training of a GAN for a three-factor experiment plan in 3D scatter plots.

On the top left (in red), the original data (χ_{d}) is shown, which stems from a random sampling. For simplicity, all factors have a value range from 0 to 10. The blue dotted plots show the generator outputs after a corresponding number of training rounds. It can be seen that this is initially around the value 0 for each factor due to the initialization of the parameters (G). From here, with more training steps, the points are shifting in the direction of the desired factor space until it is hit by the generator after about 10,000 steps. At this point, the input space is covered decently according to the experiment quality criteria that the GAN was trained for. This is basically the process of traditional GAN training. It shows that the GAN is able to generate new samples that matches a given distribution and therefore is able to learn a desired distribution in the factor space in order to generate experiment plans.

On that note, actually two options exist for generating data through the generator network. The first one is, like shown in Figure 4, to generate the configurations directly through linear transformation of the output layer. The second option is to put a sigmoid function onto the output layer, limiting the interval of the generator output to the unit vector between 0 and 1 through a non-linear function embedded in the hidden layers of the generator. The output then needs to be scaled to the appropriate factor space. The advantage here is that the generator does not need to vary its value margin as high during training, which can reduce the time needed to converge. From here, we are able to create experiment plans that can recreate an original uniform distribution. The obvious problem here is that those experiment plans lack needed quality criteria like orthogonality, balance, and space-filling properties (as described in Section 2.1). Therefore, we implemented and tested two additional measures in order to improve on this. The first one was to use experiment plans that already exhibit those quality criteria for the original database P_{data} instead of uniformly distributed random numbers.

3.1.2 Generating Experiment Plans Using Directed GANs. The second measure we carried out was to let the database of the discriminator (χ_{d}) change dynamically. As a side note, it might be debatable whether the term GAN is technically correct when used in a way where the discriminators database is altered dynamically, so we propose to call this approach *directed GAN*. This means

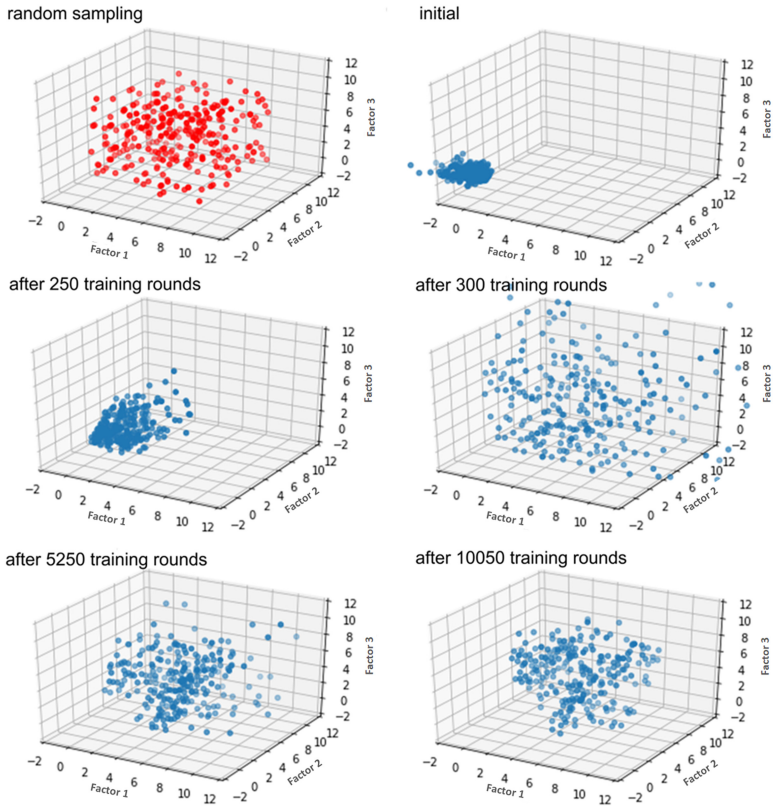

Fig. 4. Illustration of a training of a GAN (three-factor experiment plan).

that the GAN is directed towards a goal defined by an objective function. The idea here is that the GAN is given data samples drawn by a distribution that is improved over the course of the training towards the given objective function, rather than drawn from a fixed base distribution like in traditional GANs. After every training round, the best generated experiment plans are added to the database. The evaluation of the plans is done by the calculation of the quality criteria of the experiment plans itself. This way, the GAN could learn those quality criteria in an iterative process based on its own output. For this purpose, the first step is to train a GAN that is able to create an experiment plan from the set of all possible experiment plans. This means that the original data again stems from a uniform distribution. We let the GAN generate 1,000 sample experiment plans, which are then evaluated based on the following criteria: Maximum correlation $\rho_{\max}$, space-filling property measured by L_2-discrepancy ML_2, and maximum imbalance measure max Δ_k. From these samples, we choose the one that has the best value regarding one of these criteria. This step is repeated until we have enough candidates for a sufficient mini batch. This mini batch is then set as the new database $P_{\max}$ for the GAN training. This process contains 20 iterations and is repeated for each of the quality criteria. This strategy turned out to be very effective, as shown in Table 2. We can see that quality criteria for the generated configurations without these additional measures are equal to the uniformly distributed random sampling, both when generating single configurations as well as when generating the complete experiment plan at once. When we substitute the random sampling with a LHS template for the learning database, the quality criteria of the generated experiment plans start to improve slightly. When trained for a specific quality criteria by dynamically changing the discriminators database as described above, the value of the corresponding criterion

Table 2. Comparison of Quality Criteria (Smaller is Better)

Method	ρ_{max}	ML_2	max Δ_k
Random sampling	0.626	20.602	1.861
GAN – Single configuration	0.677	21.314	2.084
GAN – Complete experiment plan	0.632	24.34	1.95
GAN – Using LHS template	0.628	18.264	1.656
GAN – Trained for orthogonality	0.406	36.001	3
GAN – Trained for input space coverage	0.622	11.653	1.5
GAN – Trained for balancing	0.797	37.397	1.341
LHS [29]	0.629	13.801	0.5
NOLH [18]	0.047	13.412	0.062

Table 3. Final Hyper-parameters/Training-parameters of the Neural Networks

hyper-parameter/training-parameter	generator	Discriminator
hidden layers	1	2
neurons per layer	128	128
training steps	10050	10050
batch-size	64	64
learn-rate	0.00005	0.0001
activation function	rectified linear unit (ReLU)	rectified linear unit (ReLU)

improves drastically, but in some cases worsens the value of the other criteria. For example, this is the case when training for a balanced distribution of factor values. Obviously, Table 2 shows that a pre-optimized NOLH experiment plan still remains superior. Note that we focused on single objectives for demonstration purposes, but a more complex, combined objective function based on linear combination of multiple quality criteria could also be carried out. However, in terms of first proof-of-concept, we can see that the GAN has potential to be trained into improving quality criteria fed by the discriminator. With a fine-tuned discriminator database that can manage a balance between multiple quality criteria, the GAN might be able to produce equally valuable experiment plans. This potential should be further evaluated in future work. On a side note, the GAN is obviously not able to generate discrete factor values since the output is always continuous. Therefore, is it necessary to discretize the continuous output. This can be done by rounding, which can reduce the quality of the experiment plan in regard to orthogonality. However, increasing the number of experiments contains this effect [49].

Regarding hyperparameters, we were able to limit the range of possible parameter values due to background knowledge and upfront testing. However, we aimed to further narrow down the best hyperparameters for the GAN-training. For that purpose, we set up an experiment design for the hyperparameters and trained the GANS multiple times. Afterwards, we were able to analyze the most influential hyperparameters. After additional testing, we could then identify the best value for each parameter. The results are shown in Table 3. However, note that these settings were evaluated using a three-factor test case. For larger experiment plans, the parameters may need to be adjusted.

3.2 Using experiment-generating GANs for Robustness Optimization

In the previous section, we showed that a directed GAN can generate samples from a distribution by learning its factors, and that this can be leveraged to generate factor configurations and

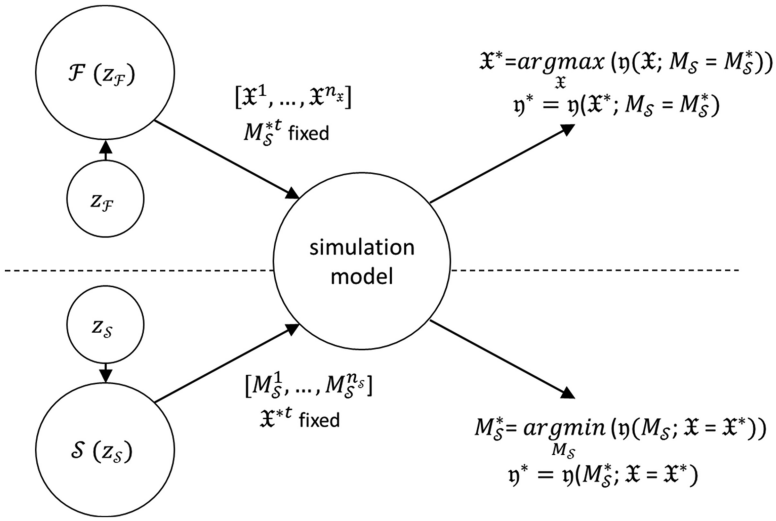

Fig. 5. Schematic structure of GAN-based robustness optimization framework.

experiment plans. This can therefore be used for a robustness optimization mechanism. The objective function that the GAN is directed to therefore needs to be based on a robustness measure. As already described in Section 1, two experiment plans are needed for the analysis or the optimization of the system robustness, one for the decision factors and one for the noise factors. Both experiment plans can be generated with the outlined approach shown in the previous section. The goal of one GAN is to maximize the robustness given a certain plan of noise, while the goal of the other GAN is to minimize the robustness given a certain system configuration. So technically, each GAN is used to improve its output regarding a given criteria as shown in Section 3.1, which in this case is the criteria of robustness. This approach is shown conceptually in Figure 5.

In the upper part of the figure, the GAN $\mathcal{F}$ (Decision GAN) is for generating configurations $\mathfrak{X}$ of the decision factors is shown. In the lower part, the GAN $\mathcal{S}$ (Noise GAN) for generating the experiment plans for the noise factors M_s is shown. Both use random numbers (z) as an input vector.

In order to be able to achieve even better robustness, the particularly robust configurations are included in the corresponding experiment plan for the decision factors $\mathfrak{X}$. On the other hand, the most unfavorable environmental conditions need to be discovered, so they must be part of the corresponding experiment plan for the noise factors (M_s). The aim of the concept is searching for a configuration $\mathfrak{X}$ which maximizes a robustness measure η under a given noise M_s. The training of the GANs is conducted alternately. This means, when the decision factors are optimized, the noise remains fixed and vice versa. The actual evaluation of the robustness is then done by conducting simulation runs and calculating the robustness values using a loss function, as shown in Section 2.1. Therefore, the two GANs need to be connected via the simulation model to compete with each other. Figure 6 shows the schematic architecture of the framework for robustness optimization, using two GANs and a simulation model. Both GANs can start the simulation model using their experiment plans and update their database for their respective discriminator based on the results of the robustness measurement.

The actual process for the optimization of robustness is shown in Figure 7. By definition, robustness is the average loss of a single decision configuration against the noise (see Section 2.1).

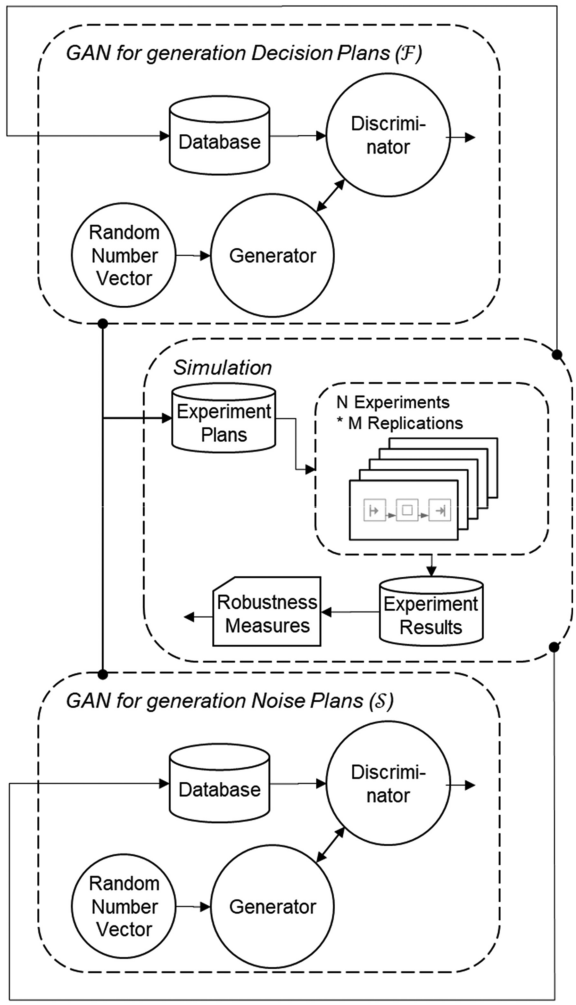

Fig. 6. Schematic architecture of GAN-based robustness optimization framework.

Thus, the value of one single decision configuration is calculated against a complete noise plan. We therefore build the optimization process so that the decision GAN generates single decision configurations, while the noise GAN generates experiment plans. The process starts with the decision GAN generating several initial decision configurations. and a randomly initialized noise plan. Those are used for the first calculation of the initial robustness values using the simulation model. Both initializations are based on the previously trained GANs that were optimized for experiment plan quality criteria. Further on, the decision GAN only generates one new decision configuration, while the noise GAN generates complete experiment plans. The process starts by training the decision GAN, which then generates a new decision configuration. After that, the simulation starts once again (running the new and best decision configurations vs. the current noise plan) followed by the calculation of the robustness values.

We use the same strategy for dynamically changing the database of the GANs as described in the previous section. This means, if a new most robust configuration was found, it is added to the

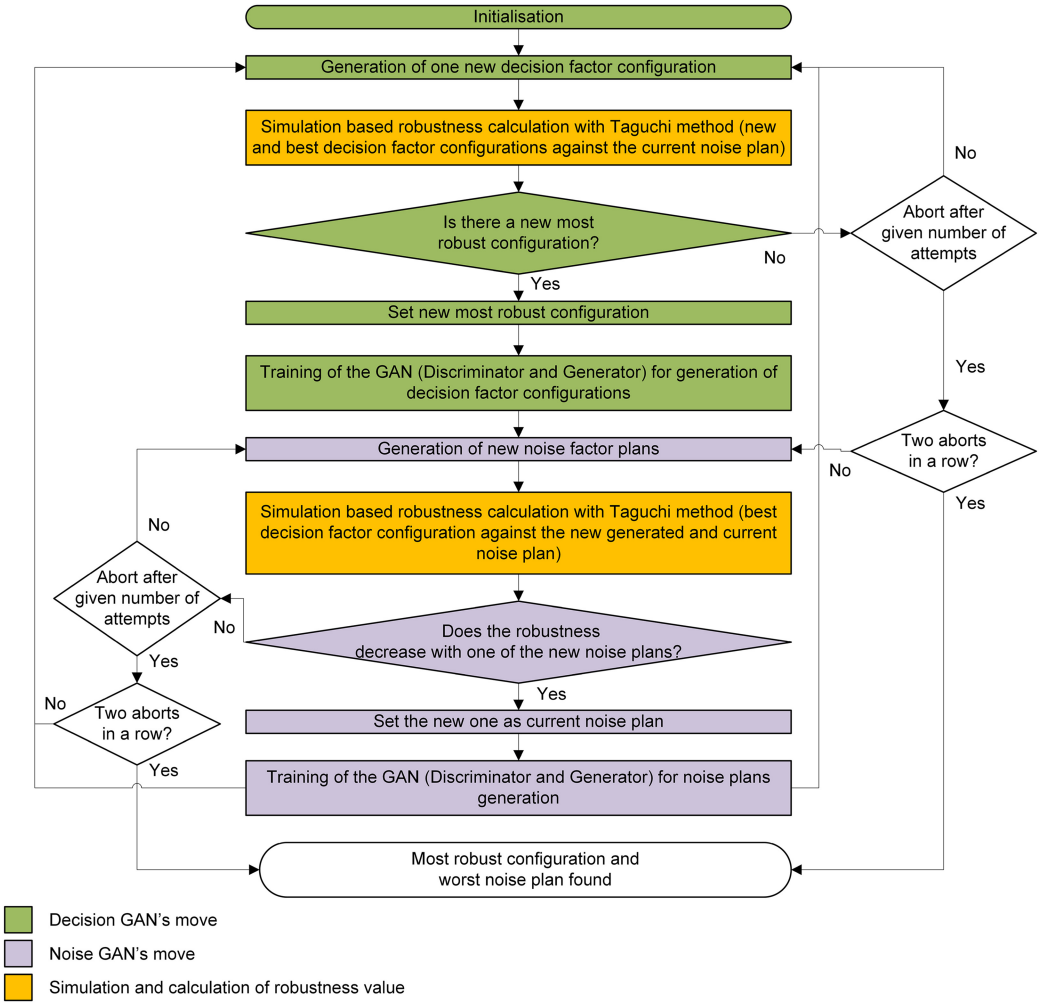

Fig. 7. Process for robustness optimization using GANs.

database of the decision GAN, and the GAN training based on the new database is carried out. The next step then is the training of the noise GAN, which is analog to the previous step. The noise GAN generates a new noise plan that is then tested against the current best decision configurations. After both GANs are trained, one iteration is completed. They can then generate new configurations based on their training of their refreshed databases. When no further improvements in the robustness can be achieved, neither in the decision GAN nor in the noise GAN, the process stops and the most robust decision configuration as well as the worst noise plan was found.

The decision, whether or not an improved configuration and/or experiment plan was found, is based on the results of the simulation runs. These are evaluated by the robustness index $\eta = -10\ log_{10}(\bar{L})$, where $\bar{L}$ is defined as in Table 1. The S/N-ratio is subsequently communicated to the GANs and is used for further training rounds.

When testing the decision configurations against the noise plan, the noise plan is kept constant. Therefore, the calculation of the S/N-ratio of these configurations is done against the current best (or worst in terms of robustness) noise plan. For testing noise plans, it goes the other way around

$$\begin{pmatrix} \eta_{1,1} & \cdots & \eta_{1,r} & \cdots & \eta_{1,R} \\ \vdots & & \vdots & & \vdots \\ \eta_{i,1} & \cdots & \eta_{i,r} & \cdots & \eta_{i,R} \\ \vdots & & \vdots & & \vdots \\ \eta_{n_{\mathfrak{X}},1} & \cdots & \eta_{n_{\mathfrak{X}},r} & \cdots & \eta_{n_{\mathfrak{X}},R} \end{pmatrix}$$

Fig. 8. Matrix of S/N-ratios.

by testing the current best decision configuration against the new noise plan. In order to get significant results, we perform R replications for each configuration against the noise plan. This results in R S/N-ratios, which are the basis for the evaluation. Thus, replications are not calculated for each single element $\mathcal{y}i$ of the crossed target matrix (see Figure 1 in Section 2), but rather for each configuration over the entire noise plan. This is because we are not optimizing for individual target values, but for S/N-ratios, which are calculated by means of the rows of the crossed target matrix. Thus, which each replication, the regarding row is simulated again. By using the robustness formula, we get a corresponding S/N-ratio for each replication of a factor configuration, which results in a matrix of S/N-ratios seen in Figure 8.

From this matrix, we can then calculate a mean value for each decision configuration. The similar strategy applies for testing multiple noise plans against a single decision configuration. Note that S/N-ratios are already a form of expectancy values themselves, since they are calculated over varying conditions in the model. Because there are usually a variety of noise factors in a discrete-event simulation model in the context of production and logistics, it is not feasible to consider all of them in the noise factor plan. When potential noise factors are not considered in the noise plan, their values have to be determined by pseudorandom numbers. But since the robustness of each configuration needs to be calculated for a given noise plan, we must ensure that those noise factors, that are not considered in the noise plan, do not add a bias to the S/N-ratio. This is important because the Noise-GAN ultimately aims to find the worst-case-scenario of noise plans and therefore relies on the S/N-ratio as a measure of assessment. This should therefore not be dependent on noise factors that are excluded from the noise plan. The problem would be that those can disrupt the signal to the noise GAN, whose noise plan currently minimizes the configurations robustness. Therefore, initial values for the random number generators for each replication of different configurations should be constant to ensure that only noise factors included in the noise plan are influential to the variance of the systems target value and the resulting S/N-ratios. This can be done by using common random numbers [27].

Furthermore, since we use multiple replications, we need to validate the selection of the best candidates statistically. The goal here is to find the true best robustness measure within the tested configurations under an error probability α. Therefore, we use classical methods of ranking and selection [14]. Using the matrix of S/N-ratios, we conduct a ranking-and-selection-screening by selection of the configuration with the best expected performance measure with indifference zone = 3 and $a = 0.05$. This way, definitively inferior configurations are weeded out. Ranking and selection also determines the number of additional replications that needs to be conducted. The same principle goes for finding noise plans. In the next step, if the best solution cannot be identified, a new set of replications is determined and conducted.

4 CASE STUDY

4.1 Implementation and Simulation Model

To verify the concept, we implemented a prototype. Figure 9 shows the prototype's main architecture. The central component is implemented in Python 3.6 (developed with Jupyter Notebook

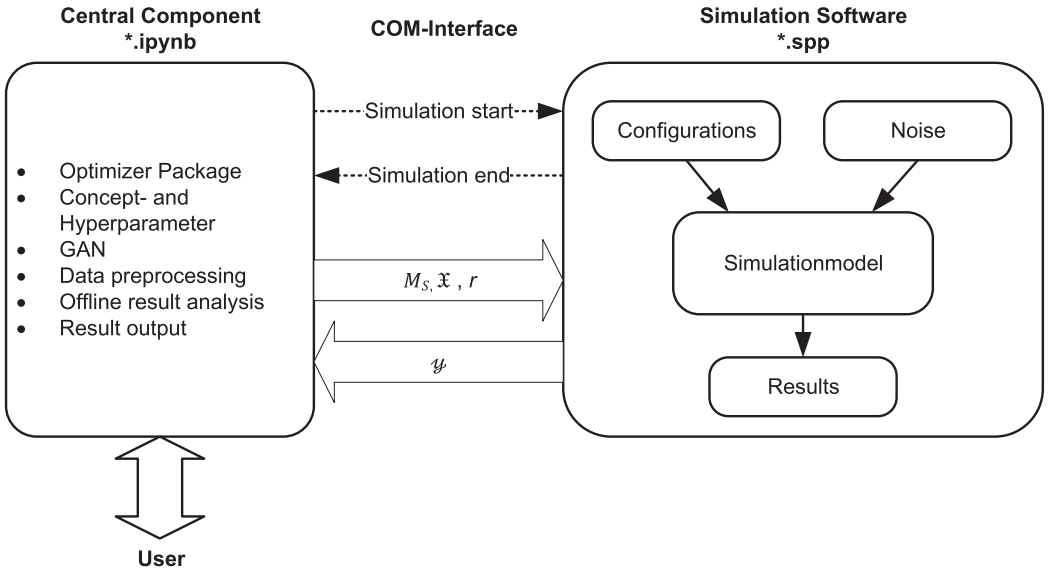

Fig. 9. Components of the implementation.

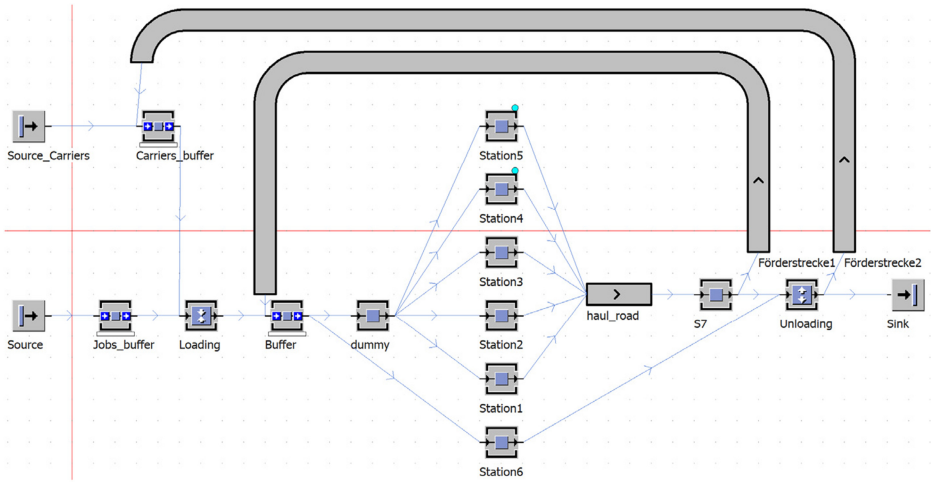

Fig. 10. Screenshot of the used Plant Simulation Model.

as a web-based interactive computational environment). Various Python libraries have been used including numpy, matplotlib, scipy, and win32.client. This Python component includes the user interaction, visualization of the results as well as the general controls. For the realization of the GANs, we used the popular open-source platform for machine learning Tensorflow, which can be addressed directly from the Python application. For implementing the simulation, we used the commercial simulator Siemens Plant Simulation (see Figure 10).

The simulator is connected via a **Component Object Model (COM)** interface. The interface allows the optimizer package to have access to the event controller of the simulation. The optimizer package can therefore start the execution of the experiments based on the generated plans by crossing decision configurations $\mathfrak{X}$ and noise plans M_s in regard to the defined number of

Table 4. Decision Factors for the Simulation Experiments

Factor name	Scale	Description	Margins
#Stations	Discrete	Number of parallel stations	1–6
S7_ProcTime	Continuous	process time of station S7	10–60s
#Carriers	Discrete	Number of work piece carriers	1–100

replications r. Once all experiments have been completed, corresponding robustness measures y are returned. Other than that, no further interactions between optimizer and simulator take place. Due to these interactions, the implementation of the conjunction of simulator and optimizer technically classifies as an optimization with integrated simulation [48].

For a proof of concept, we developed a simulation model of an assembly line. Figure 8 shows a screenshot of this model. In this model, three different part types are loaded onto carriers that are then transported via a conveyor. The parts are processed on up to six stations (Station 1-6). At the end of the line, there is a quality inspection (S7) before parts get unloaded from their carrier and leave the system. The mixture of parts can vary, but arriving parts are kept in a buffer until they are cleared to get mounted on a free carrier. Some stochastic effects arise through machine reliability and a small proportion of parts that fail the quality assurance and are rescheduled for manufacturing.

The goal here is to make the line robust against variations in the product mixture. For the actual output that should be optimized for robustness, we chose the total throughput of the line. Because we expect the throughput to be preferably large, the larger-the-better loss function was used to calculate the system robustness. The decision factors that we used for this model are shown in Table 4.

For the example scenario, we compared the traditional Taguchi based method presented in Section 2.1 with the GAN based approach presented in the previous section. Additionally, we carried out a robustness optimization using the **Response Surface Method (RMS)** for comparison.

In the traditional Taguchi approach, two different experiment design methods were used. The decision factor plan was created by using the L9 design method [36] recommended by Taguchi. The experiment design for the product mix requires a so-called mixture design. Here, the simplex lattice design [26] was used, which is a standard design method for the creation of experiments for mixture problems. In terms of noise factor values, a product mixture is obviously limited by 0%–100% per product. When choosing other factors for the noise, of course every system could possibly be made unstable with just enough noise. Therefore, the range and limits for the noise factors should be chosen reasonably and realistically according to the noise in the real system. We ran the resulting crossed experiment plan with five replications for each configuration. Using an F-Test, we confirmed that all factors have a significant effect on the S/N-ratio.

RSM, on the other hand, is a set of procedures that combine design of experiments with numerical optimization in order to describe the relation between a target value and corresponding factors. For more information on RSM see [30]. Using the RSM, an experiment plan that mutually considers both decision and noise factors as recommended in the literature [31] is not possible due to the mixture problem that needs to be addressed in the noise. The creation of such an experiment plan that covers both decision factors and mixture proportions and furthermore enables to calculate main effects, interaction effects, and quadratic effects is not feasible. Therefore, we again used a crossed design, but one that allows to determine interaction and quadratic effects in terms of the RSM. Instead of calculating an S/N-ratio, we created two metamodels: one for the target value and one for the variance of the target value. Following the guidelines in [41], we used the logarithm of the variance. For the factors, we used a custom experiment plan created through statistical

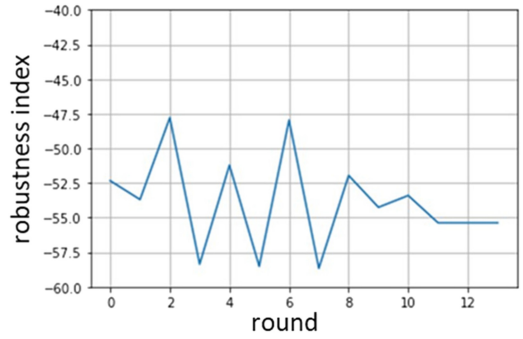

Fig. 11. Robustness measured over the 13 learning rounds.

software. In the first phase of the RSM, the complete input space is searched. It is then narrowed down in the second phase of the RSM. With the results of the second phase, we can estimate an equation that defines a decision configuration. In terms of robustness optimization, this is the stationary point of minimum variance. For our GAN concept, we applied the parameters defined in Section 3.1 (Table 3). In addition, we set a number of five replications per experiment. For the ranking and selection, an error probability (α) of 0.05 was defined.

4.2 Results

Figure 11 shows the progression of the robustness index (as calculated according to Equation 1) over the learning rounds. Note that in even rounds the GAN for the decision factors was trained, and in the odd rounds the GAN for the noise experiment plan was trained, resulting in the up-and-down-curve that we see in the figure. When the decision GAN finds a new most robust configuration, the noise GAN adapts in next iteration and brings down the robustness index in turn.

After 13 rounds of training, no further improvements could be achieved since both GANS are in a stable equilibrium by having found both the most robust configuration as well as the worst/most disrupting noise plan. Figure 12 shows a visualization of the experiment plans of the decision factors as 3D scatter plots, after the initialization and the training rounds 2, 4, 6, 8, and 10. We can see how the GAN quickly adapts from a randomized distribution to a very specific distribution of factor values that yield the highest robustness for the underlying system.

Regarding experiment plans for the noise, Figure 13 shows the processing of the product mix plans during the training of the noise GAN in comparison to the simplex lattice design (bottom right) that aims to cover the input space uniformly. In the figure, each point represents the proportion of the three products (A, B, C) in a mixture. At the beginning of the training, the noise plans represent mixtures with roughly balanced proportions for each product. During training, the points quickly move to the sides, which means that mixtures that exhibit a more dominant product seem to be preferred. Therefore, mixtures with one dominant product obviously exhibit much worse robustness than balanced mixtures. By analyzing the actual robustness measures, we found out that the increase of the proportion of product B seemed to decrease the robustness the most. Consequently, the noise GAN detected this effect correctly and adapted to it.

Table 5 shows the results of GAN-based optimization in comparison to the traditional Taguchi method and RSM that we conducted for reference. Since the robustness of throughput was investigated here, the larger-the-better loss function was used. A direct comparison of the results of all three methods is not useful since different noise plans and robustness measures are used. In order to compare the three methods, we tested the best configuration found by each of three methods against both a GAN generated noise design and the simplex lattice and calculated the

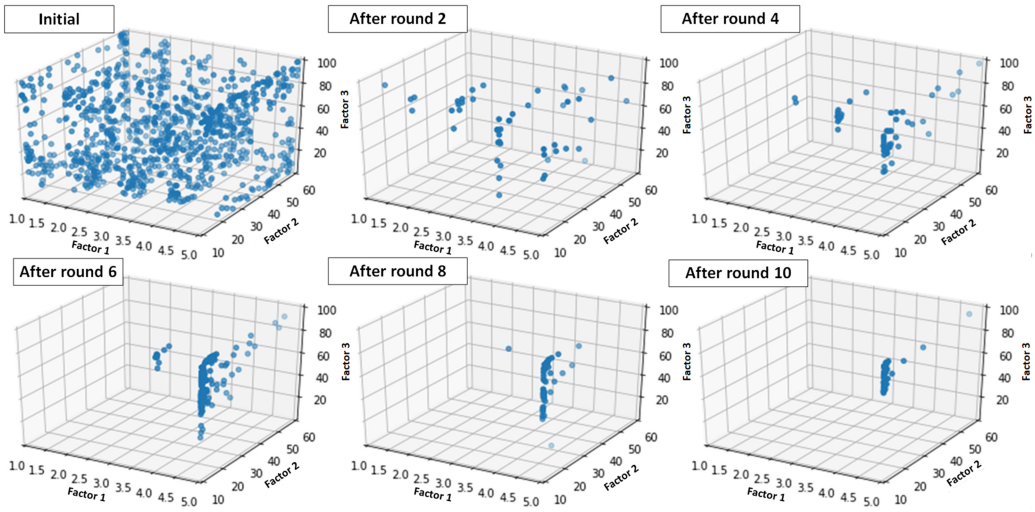

Fig. 12. Progress of the training of decision factors; from top left to bottom right initial and after training round 2, 4, 6, 8, and 10.

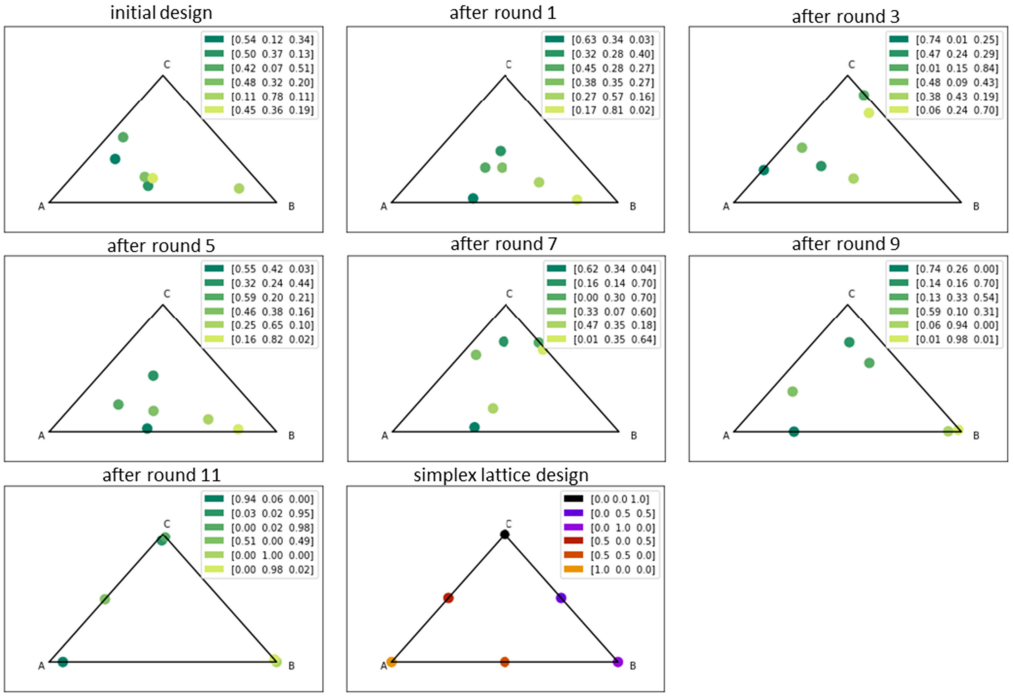

Fig. 13. Progress of the training of product mix (noise factors); from top left to bottom right: initial, after training round 1, 3, 5, 7, 9, 11, and the simplex lattice design.

Table 5. Comparison of Methods for Robustness Optimization

	Best configuration found	Robustness index (closer to 0 is better)	
		Best configuration against GAN-generated noise design	Best configuration against simplex lattice design
Taguchi method	• #Stations = 3 • S7_ProcTime =10.0s • #Carriers = 100	−64,65	−65,15
RSM	• #Stations = 5 • S7_ProcTime = 10.0s • #Carriers = 63	−57,807	−60,31
GAN approach	• #Stations = 5 • S7_ProcTime = 10.0s • #Carriers = 82	−54,72	−55,37

corresponding S/N-ratio. We can see that the three methods found different configurations to be most robust, but in all solutions, the process time on the station S7 is 10 seconds. The method based on traditional Taguchi recommends three stations and 100 carriers, whereas RSM and the solution based on GAN identified five stations as most robust. On the other hand, RSM and the GAN based solutions differ in the number of recommended carriers. Regarding the robustness measure, our GAN-based approach delivers the best results in both scenarios with a significantly better robustness. The reason why the Taguchi method performs poorly in comparison to RSM presumably is associated with the factor plan that was used here. This plan only accounts for the estimation of main effects, so interaction effects cannot be determined. However, the second phase of the RSM optimization showed that there is an interaction between the number of stations and the number of carriers in regards to their effect on the throughput. Since the calculated S/N-ratio is dependent on the mean target value, this effect was not accounted for in the solution based on the Taguchi method. In the GAN-based approach, we assume that the internal model of the generator was able to pick up this effect. The implications of this simple case study are not generalizable, yet they demonstrate the potential of the GAN-based robustness optimization in terms of a proof-of-concept. More complex case studies are needed to verify and improve the method.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we demonstrated a method for robustness optimization using two Generative Adversarial Networks that are connected via a simulation model. In a simple case study, we were able to show that with the method presented here, robust solutions can be found that beat traditional robustness optimization methods. It should be noted though that the method presented does not automatically qualify as an optimization from a purely mathematical standpoint. If we define the term optimization more broadly, the method presented provides the incremental improvement of a target value due to purposefully testing different configurations. That means, although winning against the Taguchi method and RSM, our approach does not necessarily guarantee to find a global optimum. In addition, hardly any insights into the internal model of the neural networks' behavior are possible, so that no knowledge is generated regarding why the determined solution is so robust. This is a common problem with machine learning and AI-based systems and is usually referred to as the problem of Explainability/Explainable AI [1]. The presented case study acts as proof of concept that can keep up with the traditional robustness evaluation methods. The results show that the presented method has a lot of potential but needs to be elaborated and validated

in additional case studies with more factors and larger model complexity. Further research investigating the applicability of GAN variants like InfoGAN [4], BiGAN [8], or DCGAN [37] is also conceivable. In addition, having tested our approach against more traditional approaches like Taguchi method and RSM, future work should investigate how the performance and efficiency of this approach holds up against large scale LHS or NOLH experiment designs, and against modern optimization methods like genetic algorithms and swarm optimization. This work accounts for the current trend in the simulation community to incorporate machine learning and AI-based systems into simulation workflows for optimization and therefore contributes to expanding of the portfolio of useable methods by GANs. We also showed that in a more generic way, GANs can also be used for generating experiment plans that exhibit desirable design quality criteria. This was done by setting up the GAN-training in a way that the GAN is able to learn those features. Although being only a byproduct of this work, we see a lot of potential in this approach as well, which should be investigated more in-depth in future work.

REFERENCES

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* 6, 52138–52160. DOI: https://doi.org/10.1109/ACCESS.2018.2870052.

[2] Soeren Bergmann, Niclas Feldkamp, Florian Conrad, and Steffen Strassburger. 2020. A method for robustness optimization using generative adversarial networks. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, New York, NY, USA, 1–10. DOI: https://doi.org/10.1145/3384441.3395981

[3] Dimitris Bertsimas, Omid Nohadani, and Kwong M. Teo. 2010. Robust optimization for unconstrained simulation-based problems. *Operations Research* 58, 1 (2010), 161–178. DOI: https://doi.org/10.1287/opre.1090.0715

[4] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. *InfoGAN. Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets*.

[5] Thomas M. Cioppa and Thomas W. Lucas. 2007. Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics* 49, 1 (2007), 45–55. DOI: https://doi.org/10.1198/004017006000000453

[6] Gabe Cohn. Oct. 25. AI Art at Christie's Sells for $432,500. *The New York Times* 2018 (Oct. 25.).

[7] Davi S. Correia, Cristiene V. Gonçalves, Sebastião S. Da Cunha, and Valtair A. Ferraresi. 2005. Comparison between genetic algorithms and response surface methodology in GMAW welding optimization. *Journal of Materials Processing Technology* 160, 1 (2005), 70–76. DOI: https://doi.org/10.1016/j.jmatprotec.2004.04.243

[8] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *Proceeding of the ICLR 2017*.

[9] Bruce G. Elmegreen, Susan M. Sanchez, and Alexander S. Szalay. 2014. The future of computerized decision making. In *Proceedings of the 2014 Winter Simulation Conference. Institute of Electrical and Electronics Engineers*, Piscataway, New Jersey, 943–949. DOI: https://doi.org/10.1109/WSC.2014.7019954

[10] Kai-Tai Fang, Dennis K. J. Lin, Peter Winker, and Yong Zhang. 2000. Uniform design: Theory and application. *Technometrics* 42, 3 (2000), 237–248. DOI: https://doi.org/10.2307/1271079

[11] Niclas Feldkamp, Soeren Bergmann, and Steffen Strassburger. 2020. Knowledge discovery in simulation data. *ACM Trans. Model. Comput. Simul.* 30, 4 (2020), 1–25. DOI: https://doi.org/10.1145/3391299

[12] Niclas Feldkamp, Soeren Bergmann, Steffen Strassburger, and Thomas Schulze. 2017. Knowledge discovery and robustness analysis in manufacturing simulations. In *Proceedings of the 2017 Winter Simulation Conference*. IEEE Inc.

[13] Niclas Feldkamp, Sören Bergmann, Steffen Strassburger, and Thomas Schulze. 2016. Knowledge discovery in simulation data: A case study of a gold mining facility. In *Proceedings of the 2016 Winter Simulation Conference*. IEEE Inc, Piscataway, N.J., 1607–1618. DOI: https://doi.org/10.1109/WSC.2016.7822210

[14] Jean D. Gibbons, Ingram Olkin, and Milton Sobel. 1979. An introduction to ranking and selection. *The American Statistician* 33, 4 (1979), 185–195. DOI: https://doi.org/10.1080/00031305.1979.10482690

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press, Cambridge, Massachusetts, London, England.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the International Conference on Neural Information Processing Systems* 2014.

[17] Guido Guizzi, Silvetro Vespoli, Andrea Grassi, and Liberatina C. Santillo. 2020. Simulation-based performance assessment of a new job-shop dispatching rule for the semi-heterarchical industry 4.0 architecture. In *Proceedings of the 2020 Winter Simulation Conference. Institute of Electrical and Electronics Engineers*, Piscataway, New Jersey, 1664–1675.

[18] Alejandro S. Hernandez, Thomas W. Lucas, and Matthew Carlyle. 2012. Constructing nearly orthogonal Latin hypercubes for any nonsaturated run-variable combination. *ACM Trans. Model. Comput. Simul.* 22, 4 (2012), 1–17.

[19] Humberto Hijar-Rivera, Jaime Sanchez-Leal, and Adan Valles-Chavez. 2009. Improving a soldering process applying the dual response approach to a Taguchi's orthogonal array. In *2009 International Conference on Computers & Industrial Engineering*. 1174–1178. DOI: https://doi.org/10.1109/ICCIE.2009.5223949

[20] Gary Horne, Bernt Åkesson, Ted Meyer, and Steve Anderson. 2014. *Data Farming in Support of NATO. Final Report of Task Group MSG-088.* STO technical report, TR-MSG-088. North Atlantic Treaty Organisation, Neuilly-sur-Seine Cedex.

[21] Gary E. Horne and Ted E. Meyer. 2005. Data farming: Discovering surprise. In *Proceedings of the 2005 Winter Simulation Conference. Institute of Electrical and Electronics Engineers*, Piscataway, New Jersey, 1082–1087.

[22] Tamara Jaber, Rana Nazzal, Alaa Horani, and Sameh Al-Shihabi. 2011. Selecting the best supplier based on a multi-criteria Taguchi loss function: A simulation optimization approach. In *Proceedings of the 2011 Winter Simulation Conference*. IEEE Inc, Piscataway, NJ, 4280–4288.

[23] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks.*

[24] Jack P. Kleijnen, Susan M. Sanchez, Thomas W. Lucas, and Thomas M. Cioppa. 2005. State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing* 17, 3 (2005), 263–289.

[25] Florian Klug. 2013. The internal bullwhip effect in car manufacturing. *International Journal of Production Research* 51, 1 (2013), 303–322. DOI: https://doi.org/10.1080/00207543.2012.677551

[26] D. P. Lambrakis. 1968. Experiments with mixtures. A generalization of the simplex-lattice design. *Journal of the Royal Statistical Society: Series B (Methodological)* 30, 1 (1968), 123–136. DOI: https://doi.org/10.1111/j.2517-6161.1968.tb01511.x

[27] Averill M. Law and W. D. Kelton. 2000. *Simulation Modeling and Analysis* (3. ed., international ed.). *McGraw-Hill Series in Industrial Engineering and Management Science.* McGraw-Hill, Boston.

[28] Gang Lei, Chengcheng Liu, Yanbin Li, Dezhi Chen, Youguang Guo, and Jianguo Zhu. 2019. Robust design optimization of a high-temperature superconducting linear synchronous motor based on Taguchi method. *IEEE Trans. Appl. Supercond.* 29, 2 (2019), 1–6. DOI: https://doi.org/10.1109/TASC.2018.2882426

[29] M. D. McKay, R. J. Beckman, and W. J. Conover. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 42, 1 (2000), 55. DOI: https://doi.org/10.2307/1271432

[30] Douglas C. Montgomery. 2017. *Design and Analysis of Experiments* (Ninth edition). John Wiley & Sons, Inc, Hoboken.

[31] Raymond H. Myers, Andre I. Khuri, and Geoffrey Vining. 1992. Response surface alternatives to the Taguchi robust parameter design approach. *The American Statistician* 46, 2 (1992), 131. DOI: https://doi.org/10.2307/2684183

[32] Vijayan N. Nair, Bovas Abraham, Jock MacKay, John A. Nelder, George Box, Madhav S. Phadke, Raghu N. Kacker, Jerome Sacks, William J. Welch, Thomas J. Lorenzen, Anne C. Shoemaker, Kwok L. Tsui, James M. Lucas, Shin Taguchi, Raymond H. Myers, G. G. Vining, and C. F. J. Wu. 1992. Taguchi's parameter design. A panel discussion. *Technometrics* 34, 2 (1992). 127. DOI: https://doi.org/10.2307/1269231

[33] Sidhartha Panda and Narayana P. Padhy. 2008. Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design. *Applied Soft Computing* 8, 4 (2008), 1418–1427. DOI: https://doi.org/10.1016/j.asoc.2007.10.009

[34] Gyung-Jin Park, Tae-Hee Lee, Kwon H. Lee, and Kwang-Hyeon Hwang. 2006. Robust design. An Overview. *AIAA Journal* 44, 1 (2006), 181–191. DOI: https://doi.org/10.2514/1.13639

[35] Amir Parnianifard, A. S. Azfanizam, M. K. A. Ariffin, and M. I. S. Ismail. 2018. An overview on robust design hybrid metamodeling: Advanced methodology in process optimization under uncertainty. *10.5267/j.ijiec* 9, 1–32. DOI: https://doi.org/10.5267/j.ijiec.2017.5.003

[36] Madhav S. Phadke. 1989. *Quality Engineering Using Robust Design.* Prentice Hall, Englewood Cliffs, N.J.

[37] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceeding of the ICLR* 2016.

[38] Mohd S. Said, Jaharah A. Ghani, Mohd S. Kassim, Siti H. Tomadi, and Che H. Haron. 2013. Comparison between Taguchi Method and Response Surface Methodology (RSM) In Optimizing Machining Condition. In *International Conference on Robust Quality Engineering*. 60–64.

[39] S. M. Sanchez. 2000. Robust design: Seeking the best of all possible worlds. In *Proceedings of the 2000 Winter Simulation Conference*. IEEE Inc, Piscataway, N.J. 69–76. DOI: https://doi.org/10.1109/WSC.2000.899700

[40] Susan Sanchez and Paul J. Sanchez. 2017. Better big data via data farming experiments. In *Advances in Modeling and Simulation. Seminal Research from 50 Years of Winter Simulation Conferences*, Andreas Tolk, John Fowler, Guodong Shao and Enver Yücesan, Eds. Springer International Publishing, 159–179.

[41] Susan M. Sanchez. 1994. A robust design tutorial. In *Proceedings of the 1994 Winter Simulation Conference*. 106–113. DOI: https://doi.org/10.1109/WSC.1994.717084

[42] Susan M. Sanchez. 2014. Simulation experiments: Better data, not just big data. In *Proceedings of the 2014 Winter Simulation Conference. Institute of Electrical and Electronics Engineers*, Piscataway, New Jersey, 805–816.

[43] Susan M. Sanchez. 2020. Data farming: Methods for the present, opportunities for the future. *ACM Trans. Model. Comput. Simul.* 30, 4 (2020), 1–30. DOI: https://doi.org/10.1145/3425398

[44] Susan M. Sanchez and Paul J. Sanchez. 2020. Robustness revisited: Simulation optimization viewed through a different lens. In *Proceedings of the 2020 Winter Simulation Conference. Institute of Electrical and Electronics Engineers*, Piscataway, New Jersey, 60–74. DOI: https://doi.org/10.1109/WSC48552.2020.9383880

[45] Karl Siebertz, David van Bebber, and Thomas Hochkirchen. 2010. Versuchspläne. In *Statistische Versuchsplanung - Design of Experiments (DoE)*, Karl Siebertz, David van Bebber and Thomas Hochkirchen, Eds. Springer, Berlin. 25–56, DOI: https://doi.org/10.1007/978-3-642-05493-8_2

[46] Genichi Taguchi. 1995. Quality engineering (Taguchi methods) for the development of electronic circuit technology. *IEEE Trans. Rel.* 44, 2 (1995), 225–229. DOI: https://doi.org/10.1109/24.387375

[47] Wa-Muzemba A. Tshibangu. 2018. Taguchi loss function to minimize variance and optimize a flexible manufacturing system (FMS): A Six Sigma Approach Framework. In *Proceedings of the 15th International Conference on Informatics in Control, Automation and Robotics*. SCITEPRESS - Science and Technology Publications, 602–609. DOI: https://doi.org/10.5220/0006868706020609

[48] Verein Deutscher Ingenieure. 2020. *Simulation of Systems in Materials Handling, Logistics, and Production - Simulation and Optimisation, VDI 3633 Page 12*. Beuth-Verlag, Berlin. Retrieved from.

[49] H. Vieira, Susan M. Sanchez, K. H. Kienitz, and M. C. N. Belderrain. 2013. Efficient, nearly orthogonal-and-balanced, mixed designs. An effective way to conduct trade-off analyses via simulation. *Journal of Simulation* 7, S4 (2013), 264–275. DOI: https://doi.org/10.1057/jos.2013.14

[50] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. *Few-Shot Adversarial Learning of Realistic Neural Talking Head Models*.

Received December 2020; revised November 2021; accepted December 2021