Received 18 October 2022, accepted 2 November 2022, date of publication 4 November 2022, date of current version 10 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3219979

RESEARCH ARTICLE

Explicitly Constrained Black-Box Optimization With Disconnected Feasible Domains Using Deep Generative Models

NAOKI SAKAMOTO[1,3], **REI SATO**[1,3], **KAZUTO FUKUCHI**[2,3], **JUN SAKUMA**[2,3], **AND YOUHEI AKIMOTO**[2,3]

[1]Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
[2]Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
[3]RIKEN Center for Advanced Intelligence Project, Chuo-ku, Tokyo 103-0027, Japan

Corresponding author: Youhei Akimoto (akimoto@cs.tsukuba.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science (JSPS), Grants-in-Aid for Scientific Research, under Grant 19H04179 and Grant 19J21892.

ABSTRACT We tackle explicitly constrained black-box continuous optimization problems in which the feasible domain forms a union of disconnected feasible subdomains. The decoder-based constraint-handling technique is a promising approach when the feasible domain is disconnected. However, the design of a reasonable decoder requires deep prior knowledge of the optimization problem to be solved and, hence, human effort. In this study, we investigated the usefulness of a deep neural network as a decoder and developed a training scheme for a deep neural network without prior information, such as a training dataset consisting of feasible and infeasible solutions required by existing decoder approaches. To stabilize the training of the deep generative model as the decoder, we propose decomposing the decoder into sub-models, introducing skip connections to each sub-model, and training the sub-models sequentially with separate loss functions. Numerical experiments using a test problem and a topology optimization problem show that the proposed method can find feasible domains with better objective function values and higher probability than both conventional decoder-based constraint-handling methods and non-decoder-based constraint-handling methods.

INDEX TERMS Black-box optimization, constraint handling, deep learning, disconnected feasible domain, evolutionary computation, explicit constraint, generative models.

I. INTRODUCTION

Black-Box Optimization (BBO) is a class of mathematical optimization problems in which the objective function is a black-box function. When a simulation is required to evaluate the objective function value of a solution, such an optimization problem is often treated as BBO because the relation between a solution and its objective function value is nearly a black-box. BBO appears extensively in engineering fields such as material engineering [51], aeronautical engineering [5], [42], ocean engineering [41], [46], civil engineering [11], [45], mechanical engineering [3], [17], [18], [19], [20], and artificial intelligence [13], [23].

In this study, we consider the following black-box continuous optimization with explicit constraints:

$$\operatorname*{argmin}_{\boldsymbol{x}\in\mathbb{S}} f(\boldsymbol{x}) \quad \text{s.t. } g_j(\boldsymbol{x}) \leqslant 0,\ \forall j = 1, \ldots, m, \tag{1}$$

where $f, g_1, \ldots, g_m : \mathbb{S} \to \mathbb{R}$ are the objective function and m constraint functions, respectively, and $\mathbb{S} \subseteq \mathbb{R}^n$ is the search domain on which f is well defined. We assume that $\mathbb{S}$ is a rectangular region (e.g., $\mathbb{S} = [-1, 1]^n$). In BBO, the objective function usually requires computationally expensive simulations, and the gradients of the objective function, ∇f, are unavailable or unreliable. Constraint functions can be categorized as either *explicit* or non-explicit constraints. In this study, we focus on explicit constraints, where the constraint functions are computationally cheap to evaluate, and their gradients, ∇g_j, are often available.

The associate editor coordinating the review of this manuscript and approving it for publication was Huiyan Zhang.

This study focuses on explicitly constrained BBO problems where the feasible domain $\mathbb{X} := \{\boldsymbol{x} \in \mathbb{S} \mid g_j(x) \leqslant 0, \ \forall j\}$ is the union of non-connected feasible subdomains $\mathbb{X}_{l \ (l=1,\ldots,M)}$, $\mathbb{X}_i \cap \mathbb{X}_j = \emptyset \ (i \neq j)$, that is, $\mathbb{X} := \bigcup_{l=1}^{L} \mathbb{X}_l$. Such a problem is difficult to address because it is challenging to search across different disconnected domains.

This work originated in the field of topology optimization (TO) [8], [9], [54], which is an optimization that determines the optimal distribution of materials in a given design space. Several methods have been proposed, such as level-set methods [54], density-based methods [8], and NGnet methods [48]. In TO, a material configuration is parameterized using a real vector $\boldsymbol{x}$, and an objective function encompasses a boundary value problem of partial differential equations, which are computed using numerical solution methods such as the finite element method. Therefore, the evaluation of the objective function of TO is relatively computationally expensive, and the relationship between the solutions and objective function values is almost a black-box. Restrictions, such as volume and perimeter constraints [19], are often imposed to prevent undesirable material arrangements, such as checkerboard arrays [14]. The computational cost of restrictions is significantly lower than that of the objective function, and because the relationship between the constraint function values and solution is explicitly available, it is often possible to compute their gradients. However, the resulting feasible domain can be separated into several disconnected subdomains. Therefore, this problem can be considered an example of the aforementioned BBO problems.

Evolutionary computation has been widely used as a promising optimization method in BBO. This is because evolutionary computation does not require gradients or a priori knowledge of the characteristics of the objective function, such as smoothness, and has been empirically shown to have a low initial value dependence, which are properties required for BBO. In particular, the covariance matrix adaptation evolution strategy (CMA-ES) [2], [25], [29], [30], [32], a type of evolutionary computation, is a quasi-parameter-free method, and has been successfully applied to several real-world BBOs [10], [16], [36], [40], [41], [52], [53], including various TO applications [17], [18], [19], [20]. In this study, we also investigate optimization based on CMA-ES.

Decoder-based techniques [12], [35] are considered promising approaches for BBO problems with disconnected feasible subdomains [43]. The idea underlying decoder-based approaches involves designing a map $G : \mathbb{Z} \to \mathbb{S}$, called the decoder, which is a mapping from a relatively simple and possibly convex region $\mathbb{Z}$, such as a hypercube, to the search domain $\mathbb{S}$. In this work, we consider setting $\mathbb{Z} = \mathbb{S} \subseteq \mathbb{R}^n$. Ideally, the image of G should be in the feasible domain, that is, $G(\mathbb{Z}) = \mathbb{X}$. Then, the original problem (Equation (1)) is translated into the following problem:

$$\operatorname*{argmin}_{z \in \mathbb{Z}} f(G(z)) \quad \text{s.t. } g_j(G(z)) \leqslant 0, \ \forall j = 1, \ldots, m. \qquad (2)$$

Once a reasonable decoder is obtained, the solver only needs to deal with a relatively simple boundary of $\mathbb{Z}$. However, existing works in decoder-based approaches [12], [35] require prior knowledge about the feasible domain or a dataset of feasible points to design or train a decoder.

The main contribution of this study is the automation of the decoder design while inheriting the decoder feature that transforms the original optimization problem with a disconnected feasible domain $\mathbb{X}$ (Equation (1)) into an optimization problem with a convex domain $\mathbb{Z}$ (Equation (2)), whose boundary is easy to handle. In practice, owing to the continuity and incomplete training of neural networks, $G(\mathbb{Z}) \subseteq \mathbb{X}$ cannot be guaranteed, and the constraints of the transformed optimization problem (Equation (2)) are violated in some regions of $\mathbb{Z}$. However, the fraction of the Lebesgue measure of the feasible domain on $\mathbb{Z}$ (regions of $z \in \mathbb{Z}$, such that $G(z) \in \mathbb{X}$) is much larger than the fraction of the Lebesgue measure of the feasible domain $\mathbb{X}$ on the search domain $\mathbb{S}$, as shown in the numerical experiments in Section IV. When the proportion of infeasible domains is small, a stochastic multipoint search method, such as CMA-ES, can ignore the infeasible domains because the probability of generating infeasible solutions is sufficiently low. This is expected to make it easier to identify feasible domains with globally superior objective function values.

The remainder of this paper is organized as follows: Section II reviews decoder-based constraint-handling techniques; Section III introduces the proposed method; Section IV quantitatively analyzes the proposed decoder G using a test problem in which the number of disconnected regions in the feasible domain can be adjusted; Section V verifies the effectiveness of the proposed method using a topology optimization problem by comparing it with existing decoder methods and typical constraint-handling methods; and finally, Section VI concludes the paper.

II. REVIEW OF DECODER-BASED CONSTRAINT-HANDLING

Many evolutionary computation methods, including CMA-ES, were designed by assuming unconstrained optimization or optimization with only rectangular constraints. Therefore, when applying them to constrained optimization, it is necessary to use them in conjunction with constraint-handling methods. The penalty function method is widely used in engineering as a constraint-handling method owing to its versatility and ease of implementation. However, the optimization becomes inefficient if the penalty coefficients are not well adjusted; if the penalty is too small, the constraints will be violated, and if the penalty is too large, the landscape of the penalized objective function will have a deceptive structure (i.e., a globally weak structure [27]), as shown in Figure 1. It is empirically known that CMA-ES can appropriately search on globally well-structured functions (i.e., functions with a big valley structure); however, it often fails to locate good local optimal solutions on globally

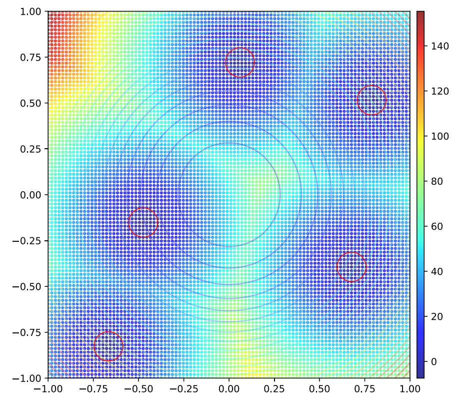


FIGURE 1. Landscape of the penalized objective function when the penalty coefficient is too large. The feasible subdomains are inside the red circles. The color map shows the penalized objective function values.

weakly structured functions (i.e., functions with a deceptive structure) [28], [39]. Therefore, such a constraint-handling method is not suitable for BBO problems with disconnected feasible subdomains.

The advantages of using a decoder include the following factors. First, we believe that optimizing the transformed problem in Equation (2) is expected to be much easier than optimizing the original problem Equation (1); if we have a decoder satisfying $G(\mathbb{Z}) \approx \mathbb{X}$, then the optimization method only needs to consider the simple boundary of the convex domain $\mathbb{Z}$, and effective constraint-handling methods have been proposed for such constraints (e.g., [47]). Second, we can utilize the fact that the computational cost of the constraint function g_j is significantly lower than that of the objective function f. In the case of the penalty function method, it is necessary to evaluate f, which is expensive even when the constraint violations are large and the objective function values are not necessary to guide the search method. By contrast, learning a decoder requires considerable access to constraint functions, but the cost is relatively small. Searching for the function defined in Equation (2) using the learned decoder on $\mathbb{Z}$ corresponds to searching only in the feasible domain $\mathbb{X}$; thus, the number of f-calls can be reduced. Therefore, when the measure of the feasible domain $\mathbb{X}$ is very small compared with the measure of the search domain $\mathbb{S}$, it is expected that optimization can be achieved in less time.

In [35], a method was proposed to create a quasi-isomorphic map between $\mathbb{X}$ and an n-dimensional hypercube $\mathbb{Z}$ as the decoder. This mapping was defined non-explicitly by a line search between the generated candidate solution z and a predefined reference point (a feasible solution). The disadvantage of this method was that the decoder tended to project an infeasible solution to the feasible domain in which the reference point was located. Therefore, domain knowledge of the problem and trial-and-error are required to determine a reasonable reference point. In [12], a method using a support vector data description (SVDD) was proposed to model and train a decoder using a dataset consisting of feasible and infeasible solutions. This method alleviated the tendency to select a single feasible domain, but it required the preparation of the training dataset in advance. The SVDD-based decoder method required expert knowledge of the problem and human effort to design and train the decoder. If we can obtain a decoder without domain knowledge or human effort, we can automatically transform a difficult problem (Equation (1)) into a relatively simple problem (Equation (2)); however, to the best of our knowledge, such a method has not yet been explored.

The purpose of this work is to automate the design of the decoder $G : \mathbb{Z} \to \mathbb{X}$ and facilitate the handling of explicit constraints in BBO, where the feasible domain is the union of several disconnected domains. As the feasible domain $\mathbb{X}$ is disconnected, the decoder must be able to represent highly nonlinear maps. Recent developments in deep generative models, such as generative adversarial networks (GANs) [21], variable auto-encoders (VAEs) [34], and flow-based deep generative models (flows) [15], have shown promising performance in terms of representing highly nonlinear mappings. However, unlike machine learning tasks, in which deep generative models are usually applied, the difficulty in using deep generative models in this research is that we do not have prior training data at hand. To address this difficulty, we propose a new training method that does not require the prior collection of training data.

In this study, we propose a method for training a decoder using a deep neural network. To represent a highly nonlinear mapping, it is necessary to use a deep network; however, the deeper the network, the more difficult it becomes to stably learn a disconnected feasible domain, and it is easy to obtain a model that can only represent a few disconnected domains. This phenomenon is similar to the well-known *mode collapse* observed in deep generative models. To solve this problem, we propose to divide the decoder network into sub-models, define a loss function for each sub-model, and train each sub-model sequentially. Numerical experiments show that the proposed method can stably learn many disconnected feasible domains without using training data.

III. PROPOSED METHOD

The primary research question in this study is that of how to train a decoder G without preparing the training data in advance, that is, without human effort. In GANs, we train a generator such that its output is indistinguishable from the training data by the discriminator. In the current context, the training data correspond to a feasible solution set in the feasible domain $\mathbb{X}$; however, we do not have such training data. Instead, we use the information that the constraint functions g_j that define $\mathbb{X}$ are explicit.

A. DESIGN PRINCIPLE

We consider that G should have the following properties: (a) G is a surjection from latent space $\mathbb{Z} \subseteq \mathbb{R}^n$ to feasible domain $\mathbb{X} \subseteq \mathbb{R}^n$; (b) in each feasible subdomain, $\mathbb{X}_\ell$,

$G^{-1} : \mathbb{X}_\ell \to \mathbb{Z}$ is continuous; and (c) in the latent space, two neighboring points $z_1, z_2 \in \mathbb{Z}$ are projected onto the same feasible subdomain $\mathbb{X}_\ell$ or onto neighboring feasible subdomains $\mathbb{X}_\ell$ and $\mathbb{X}'_\ell$. These properties are desirable for successful optimization by transforming the original optimization problem defined in Equation (1) into the problem in Equation (2). Requirement (a) is necessary because if $\boldsymbol{x}^* = \operatorname{argmin}_{\boldsymbol{x}\in\mathbb{S}} f(\boldsymbol{x}) \notin G(\mathbb{Z})$, then solving the transformed problem will not yield the optimal solution $\boldsymbol{x}^*$. The requirements (b) and (c) represent a type of continuity; if a small change in the latent space leads to a large change in its projection, a sudden change in the objective function value occurs. Consequently, even if the original objective function f is unimodal and has a good scale, the transformed objective function $f \circ G$ can either be multimodal or have a bad scale. The above requirements are intended to prevent these problems from occurring.

We represent a decoder G that satisfies the above requirements by using neural networks (NN) $G_\theta : \mathbb{R}^n \to \mathbb{R}^n$. In the following, we consider the latent space $\mathbb{Z}$ of the decoder to be equal to the search domain $\mathbb{S}$. In this case, we can consider learning G_θ by minimizing the loss function, as follows:

$$L(\theta) = \mathbb{E}_{z\sim P_z}\left[\|z - G_\theta(z)\|^2 + \gamma \cdot \phi\left(G_\theta(z)\right)\right], \qquad (3)$$

where P_z represents a probability distribution on $\mathbb{Z} \subseteq \mathbb{R}^n$ and $\phi : \mathbb{R}^n \to \mathbb{R}_{\geqslant 0}$ is an aggregation of constraint violations. For example, it can be defined as

$$\phi(\boldsymbol{x}) = \frac{1}{m}\sum_{j=1}^{m} \min\left\{g_j\left(\Pi_{\mathbb{Z}}(\boldsymbol{x})\right), 0\right\} + \|\boldsymbol{x} - \Pi_{\mathbb{Z}}(\boldsymbol{x})\|^2, \quad (4)$$

where $\Pi_{\mathbb{Z}} : \mathbb{R}^n \to \mathbb{Z}$ is the projection onto $\mathbb{Z}$. As we assume that $\mathbb{Z} = \mathbb{S}$ is a hyper-rectangle and it is easy to force the decoder to output solutions in $\mathbb{Z}$, e.g., by using `Tanh` output activation, we can often ignore the projection. By training the NN according to Equation (3), we expect $z = G_\theta(z)$ if $z \in \mathbb{X}$, and $G_\theta(z)$ to be a projection from z to the nearest feasible domain if $z \notin \mathbb{X}$. When training the decoder G_θ, we generate M samples, $z_1, \ldots, z_M$ (called *mini-batch*), from P_z independently every iteration and apply mini-batch training. In other words, the expected value $\mathbb{E}_{z\sim P_z}$ is minimized by replacing it with the sample mean $\frac{1}{M}\sum_{i=1}^{M}$ for the newly generated mini-batches in each iteration. Note that this loss function does not require the preparation of training data, such as feasible solution sets, in advance.

By training G_θ using Equation (3) as the loss function, we expect to obtain G_θ, such that $G_\theta(z) \in \mathbb{X}$ for any $z \in \mathbb{Z}$. However, as shown in Section IV, $G_\theta(\mathbb{Z})$ trained using the loss function above tends to be biased toward some regions of $\mathbb{X}$. In standard generative models, such as GANs, the phenomenon of output bias toward some training data is notorious as a mode collapse. In this study, we did not use training data; therefore, the phenomenon described above is not necessarily the same as general mode decay; however, it is similar. Here, we refer to mode collapse as the state in which the NN model learns only a portion of the disconnected feasible domain. In the next section, we present the proposed structure and loss function of an NN that avoids mode collapse.

B. PROPOSED STRUCTURE AND TRAINING SCHEME FOR DEEP DECODER NETWORK

One reason for mode collapse is that the decoder network has insufficient representational capability. To learn mapping to a disconnected region, such as the one targeted in this study, it is necessary to use a network with sufficient expressive power, that is, a sufficiently deep network. However, the deeper the network, the more unstable its learning is.

The first idea is to introduce a *skip connection* into the decoder network. When the loss function is defined as in Equation (3), the first term requires that G_θ is close to an isomorphic map. However, it is challenging to realize isomorphic mapping in deep neural networks. To reduce the difficulty, we incorporated the skip connection proposed by ResNet [31] into the model structure.

A skip connection is a structure in which an input to the model is added to the output. That is, we have $G_\theta(z) = z + G_\theta^{\text{res}}(z)$, where the model to be trained is the residual block $G_\theta^{\text{res}} : \mathbb{Z} \to \mathbb{R}^n$. By adopting a skip connection, only the difference between z and the feasible domain $\mathbb{X}$ must be represented by G_θ^{res}. This is expected to reduce the learning difficulty compared with simply learning the mapping from $\mathbb{Z}$ to $\mathbb{X}$.

Consequent upon preliminary experiments, the number of feasible domains that decoder G_θ can learn was improved by introducing a skip connection. However, in tasks with numerous disconnected feasible domains, we have not been able to prevent mode collapse, in which $G_\theta(\mathbb{Z})$ is biased toward some regions. This may be due to the difficulty of training G_θ such that all $z \notin \mathbb{X}$ are projected to the nearest feasible domain $\mathbb{X}_\ell$. If both the mini-batch size and the capacity of G_θ are large, it would be feasible. However, in reality, the mini-batch size is finite, and when θ is updated to project a finite number of latent vectors to $\mathbb{X} = \bigcup_\ell \mathbb{X}_\ell$, the latent vectors that are not in the mini-batch are mapped to $\mathbb{X}_\ell$. Once such a mapping is learned, it is difficult to correct it because when θ is updated such that an input z is mapped to $\mathbb{X}_\ell$, the constraint term is dominant in Equation (3). If z is incorrectly projected to a distant $\mathbb{X}_\ell$, and we try to move it to a neighboring $\mathbb{X}_\ell$ by the effect of homogeneous mapping, a single parameter update will only take z out of the constraint momentarily, and it will be quickly pushed back to the original $\mathbb{X}_\ell$ when the constraint term is dominant.

The second innovation represents the decoder G_θ as a composite $G_\theta = G_{\theta_K} \circ \cdots \circ G_{\theta_1}$ of several sub-models $G_{\theta_1}, \ldots, G_{\theta_K}$, and trains each sub-model G_{θ_k} independently. For input z, we define $\boldsymbol{x}_0 = z$ and $\boldsymbol{x}_k = G_{\theta_k}(\boldsymbol{x}_{k-1})$. In this case, $G_\theta : z \mapsto \boldsymbol{x}_K$ and $\boldsymbol{x}_K$ are the final output. The idea is to gradually move the input data closer to $\mathbb{X}$ each time it passes through each sub-model G_{θ_k}; thus, the difference in the mapping of G_{θ_k} is small and mode collapse is prevented

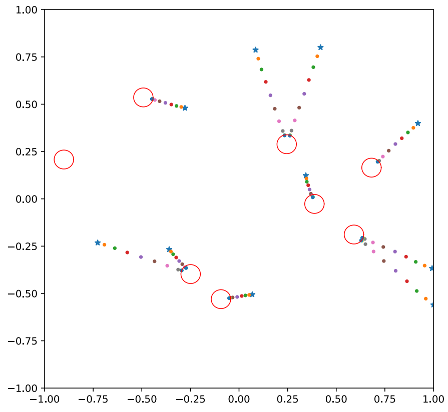


FIGURE 2. Visualization of the sequences $x_0 = z, x_1 = G_{\theta_1}(x_0), \ldots, x_K = G_{\theta_{K-1}}(x_{K-1})$ generated by inputting several latent vectors z (star) to the decoder obtained by SeqTrain (the proposed approach).

(see Figure 2). Each sub-model G_{θ_k} is represented using a skip connection, $G_{\theta_k}(\boldsymbol{x}) = \boldsymbol{x} + G^{\mathrm{res}}_{\theta_k}(\boldsymbol{x})$, as shown in the first device, and is trained according to the loss function

$$L_k(\theta_k) := \mathbb{E}_{z\sim P_z}\Big[\|\boldsymbol{x}_{k-1} - G_{\theta_k}(\boldsymbol{x}_{k-1})\|^2 + \gamma_k \cdot \phi(G_{\theta_k}(\boldsymbol{x}_{k-1}))\Big], \quad (5)$$

where $\gamma_k = \gamma_0\alpha^{k-1}$ are the penalty coefficients. The reason for increasing the coefficients γ_k exponentially is to align the differences between the mappings of each sub-model G_{θ_k} as much as possible. Because the output $\boldsymbol{x}_k$ approaches $\mathbb{X}$ as k increases, the constraint term decreases as k increases. In other words, to equalize the weight of the constraint term with the first term for each loss function L_k, it is appropriate to increase the coefficient γ_k as k increases. This idea is expected to alleviate the aforementioned problems.

The third innovation is to train the coupled model sequentially. The simplest way to train the coupled model $G_\theta = G_{\theta_K} \circ \cdots \circ G_{\theta_1}$ described above is to update the training parameters of all the sub-models G_{θ_k} simultaneously at every iteration. By contrast, in the proposed method, we train the connected sub-models G_{θ_k} starting from $k = 1$. Initially, only the model G_{θ_1} is trained using Equation (5). After a certain number of training iterations, the parameter θ_1 is fixed. Next, the model G_{θ_2} is trained. In this case, the input to the summodel G_{θ_2} is $G_{\theta_1}(z)$ for each input z. This process is repeated until $k = K$. In other words, there is always only a single sub-model G_{θ_k} being trained, and its input is the output of model $G_{\theta_{k-1}} \circ \cdots \circ G_{\theta_1}$ with fixed parameters. The advantage of sequential training is that the number of models can be increased indefinitely by repeating the aforementioned operations; therefore, there is no need to determine the number of models K in advance. In addition, by starting with a small value of the coefficient γ_k of the constraint term and increasing the number of models while observing the

Algorithm 1 Sequential Decoder Training

```
for $k = 1, \ldots, K$ do
    $\gamma_k \leftarrow \gamma_0\alpha^{k-1}$
    for Itr $= 1, \ldots,$ Itr$_{\max}$ do
        $z_i \sim P_z$ for $i \in \{1, \ldots, M\}$
        $\boldsymbol{x}_{k-1,i} \leftarrow G_{\theta_{k-1}} \circ \cdots \circ G_{\theta_1}(z_i)$ for $i \in \{1, \ldots, M\}$
        Approx. (5) with mini-batch loss $\widehat{L_k}(\theta_k)$
        Train $\theta_k$ with $\nabla\widehat{L_k}(\theta_k)$
    end for
end for
return $G = G_{\theta_K} \circ \cdots \circ G_{\theta_1}$
```

situation, we can reduce the risk of setting γ_k too large and causing mode collapse.

The proposed training scheme for the decoder network is summarized in Algorithm 1.

IV. QUANTITATIVE EVALUATION OF TRAINED DECODER ON TEST PROBLEM

This experiment was performed to quantitatively evaluate the effectiveness of the proposed decoder training method in learning disconnected feasible domains. We focused on determining the number of regions that could be learned when the feasible domain was the union of several disconnected regions; that is, $\mathbb{X} = \bigcup_{\ell=1}^{L}\mathbb{X}_\ell$ and $\mathbb{X}_\ell \cap \mathbb{X}_{\ell'} = \emptyset$ for any $\ell \neq \ell'$.

First, we designed a test problem to quantitatively evaluate the number of learned feasible domains $\mathbb{X}$ in model G_θ. Using this test problem, we focused on (*i*) the coverage of disconnected regions (see below) and (*ii*) the search performance using a trained decoder. Using these two metrics, we quantitatively measured the quality of the decoder training method.

A. TEST PROBLEM

To evaluate indices (*i*) and (*ii*), we define the following test problem on $\mathbb{S} = [-1, 1]^n$:

$$f(\boldsymbol{x}) := \|\boldsymbol{x} - \boldsymbol{x}_f^*\|^2$$
$$g(\boldsymbol{x}) := \min_{\ell=1,\ldots,L}\{\max\{0, \|\boldsymbol{x} - \boldsymbol{x}_\ell^*\|^2 - \varepsilon^2\}\},$$

where $\boldsymbol{x}_\ell^* \in \mathbb{R}^n$ denotes a uniform randomly generated vector from $[-1+\varepsilon, 1-\varepsilon]^n$. The feasible domain is the union of L disconnected closed hyperspheres, $\bar{B}(\boldsymbol{x}_\ell^*, \varepsilon)$. In other words, $\mathbb{X} = \bigcup_{\ell=1}^{L}\mathbb{X}_\ell$, where $\mathbb{X}_\ell = \bar{B}(\boldsymbol{x}_\ell^*, \varepsilon)$. The number of disconnected feasible subdomains and their volumes are easily controllable by changing L and ϵ. Moreover, because the location of each feasible subdomain is clear, it is suitable for analyzing the quality of trained decoders.

The optimal solution of the objective function $\boldsymbol{x}_f^* \in [-1, 1]^n$ is randomly sampled such that $\boldsymbol{x}_f^* \notin \mathbb{X}$, and the optimal solution to the test problem is at the boundary of the nearest feasible domain from $\boldsymbol{x}_f^*$, that is, $\boldsymbol{x}^* = \mathrm{argmin}_{\boldsymbol{y}\in\mathbb{X}}\|\boldsymbol{x}_f^* - \boldsymbol{y}\|$. The number of dimensions was $n = 10$, the number of feasible domains was $L = 100$, and the radius was $\varepsilon = 0.25$.

TABLE 1. Network architecture specifications. The decoder comprised K sub-models, $G_\theta = G_{\theta_K} \circ \cdots \circ G_{\theta_1}$, and each sub-model was $G_{\theta_k}(x) = x + G^{\mathrm{res}}_{\theta_k}(x)$. Batch normalization (BN) and an activation function were applied before each dense layer (i.e., fully connected layer) if they are indicated in the table.

Kernel	BN	Activation	Resample	Output Shape
Residual block $G^{\mathrm{res}}_{\theta_k} : \mathbb{R}^n \to \mathbb{R}^n$				
Dense	✓	Mish	-	128
Dense	✓	Mish	-	128
Dense	✓	Mish	-	128
Dense	✓	Mish	-	n
-	-	Tanh	-	n

The ratio of the feasible domain $\mathbb{X}$ to the search domain $\mathbb{S} = [-1, 1]^n$ was $L \cdot [\pi^{\frac{n}{2}} / \Gamma(\frac{n}{2} + 1)] \cdot (\varepsilon/2)^n \approx 2.38 \times 10^{-7}$; the feasible domain of the test problem was very small compared with that of the search domain.

B. EXPERIMENTAL SETTING

The structure of each sub-model G_{θ_k} ($k = 1, \ldots, K$) is described in Table 1. Mish [44] is similar to ReLU, which is widely used as an activation function; however, Mish is non-flat and smooth everywhere, and the loss function is also smooth, making it easier to optimize than ReLU. The number of sub-models in the coupled model was set to $K = 10$, the initial value of the coefficients of the constraint term was $\gamma_0 = 0.1$, and the increasing coefficient was $\alpha = 1.3$. The ADAM optimizer [33] was used for training, with a mini-batch size of 128, a learning rate of 0.001, and $\beta_1 = 0.3$. A uniform distribution $U(\mathbb{S})$ was used as the generating distribution P_z of z. The maximum number of training iterations was set to 20×10^4, and the number of training iterations for each sub-model during sequential training was set to $20 \times 10^4 / K = 2 \times 10^4$.

The data used by the chosen existing decoder methods, a decoder with Homomorphous Mapping (HM) [35] and decoder with SVDD [12], that is, the set of feasible solutions, were collected by minimizing the constraint violations as the objective function by CMA-ES and saving the feasible solutions obtained in the search process. When g was minimized and the search distribution enters the feasible domain, most of the candidate solutions become feasible solutions, their objective function values all become zero, and the search distribution starts a random search within the feasible domain. The search distribution was restarted every time 100 feasible solutions were collected, and feasible solutions spanning as many regions as possible were collected.

We define the following two evaluation indices for the model G after training:

$$\text{feasibility} := \Pr[G(z) \in \mathbb{X} : z \sim U(\mathbb{S})].$$
$$\text{coverage} := \#\{\ell \in [\![1, L]\!] : \Pr[G(z) \in \mathbb{X}_\ell \leqslant P_{\mathrm{cov}}/L]\}/L.$$

Here, feasibility denotes the probability that $G(\mathbb{Z})$ will enter $\mathbb{X}$ and coverage denotes the fraction of the disconnected partially executable region $\mathbb{X}_\ell$ that G has learned. In our experiments, we set $P_{\mathrm{cov}} = 0.01$, and the probability Pr was estimated using the Monte Carlo method separately from the mini-batch, with 2^{15} samples $z \sim P_z$. By examining these two metrics, we can evaluate how well model G is learning $\mathbb{X}$.

Optimization of the test problem using CMA-ES with the trained decoder model was conducted for 20 trials. For each trial, the mean vector, step size, and covariance matrix were initialized to $\boldsymbol{m}^{(0)} \sim U(\mathbb{S})$, $\sigma^{(0)} = 2/5 = 0.4$, and $\boldsymbol{C}^{(0)} = \boldsymbol{I}_n$, respectively. The true optimal solution of the objective function was set to $\boldsymbol{x}_f^* \sim U(\mathbb{S})$ for each trial, and the optimal solution of the problem $\boldsymbol{x}^*$ was initialized accordingly. Each trial was terminated when the maximum number of evaluations of the objective function reached 15,000 or when $\|G(\boldsymbol{m}) - \boldsymbol{x}^*\|^2 \leqslant 10^{-2}$ was satisfied. If the latter termination condition was satisfied, it was considered a successful trial as this condition is satisfied only when the feasible subdomain where $\boldsymbol{x}^*$ exists is located.

Because infeasible solutions may be generated even with a trained decoder G, we combined CMA-ES and the constraint-handling method ε-ordering [50] to handle such solutions and optimize them. ε-ordering is based on the constraint violation $\phi(\boldsymbol{x}) = \sum_{j=1}^m \max\left(0, g_j(\boldsymbol{x})\right)$, and ranks solutions using the ε-level comparison defined by

$$\{f(\boldsymbol{x}_1), \phi(\boldsymbol{x}_1)\} \leqslant_\varepsilon \{f(\boldsymbol{x}_2), \phi(\boldsymbol{x}_2)\}$$
$$\Leftrightarrow \begin{cases} f(\boldsymbol{x}_1) \leqslant f(\boldsymbol{x}_2) & \text{if } \phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2) \leqslant \varepsilon(t) \\ f(\boldsymbol{x}_1) \leqslant f(\boldsymbol{x}_2) & \text{if } \phi(\boldsymbol{x}_1) = \phi(\boldsymbol{x}_2) \\ \phi(\boldsymbol{x}_1) < \phi(x_2) & \text{otherwise,} \end{cases}$$

where $\varepsilon(t)$ is defined as

$$\varepsilon(t) = \begin{cases} \varepsilon(0)\left(1 - \frac{t}{T_c}\right)^{\mathrm{cp}} & \text{if } 0 < t < T_c \\ 0 & \text{otherwise} \end{cases}$$
$$\varepsilon(0) = \phi(\boldsymbol{x}_\theta).$$

Here, $\phi(\boldsymbol{x}_\theta)$ ($\theta = \lceil 0.2N \rceil$) is the value of 20% of the constraint violation amount of infeasible solutions out of the $N = 100$ solutions generated from the initial search distribution of the CMA-ES,[1] T_c is the maximum number of iterations ($T_c = 0.8\, T_{\max}$) and cp $= 5$. The algorithm prioritizes the objective function while allowing constraint violations in the early stages of the optimization, but gradually increases the weight of the constraint function and emphasizes it in the final stages to eventually converge to the feasible domain. The top three methods in the CEC2020 Competitions on Real-World Single-Objective Constrained Optimization that dealt with single-objective constrained problems based on real problems [22], [37], [38] employ ε-ordering and its improved versions. In addition, they are particularly effective for problems where obtaining a feasible solution is difficult. As a baseline, we also show the results of optimization using

[1]When using a decoder, many of the generated solutions are feasible. To calculate the constraint violation amount, it is desirable to generate a sufficient number of solutions. Even if numerous solutions are generated, the computational cost is assumed to be sufficiently low because only the constraint functions are evaluated.

CMA-ES with ε-ordering without using a decoder, i.e., $G : \boldsymbol{x} \mapsto \boldsymbol{x}$ (denoted as BruteForce).

C. RESULTS AND DISCUSSION

We first check the effectiveness of the proposed training method, and then we examine the effect of transforming the search domain using the decoder on the optimization results.

1) EFFECT OF COMPONENT-WISE LOSS FUNCTION

In the proposed method, we confirmed that training each sub-model using the loss function defined in Equation (5) leads to the suppression of mode collapse; consequently, a large number of feasible domains $\mathbb{X}$ can be trained. For this purpose, we compared the following three approaches: (1) SeqTrain (sequential training) is a method in which sub-models are trained sequentially; (2) SimulTrain (simultaneous training) trains all sub-models simultaneously, but with the same loss function and architecture as SeqTrain; and (3) SingleFix uses the same architecture as SeqTrain and SimulTrain, but trains the entire model using the single loss function defined in Equation (3). By comparing SingleFix with the other two, we can determine the effect of training each sub-model with multiple loss functions.

The feasibility and coverage of the decoder G obtained after training are summarized in Table 2. The SingleFix result shows that feasibility was improved by increasing the coefficient of the penalty term γ, but the coverage was low in both cases, indicating that only a small portion of the feasible domain can be trained. By contrast, the proposed methods, SeqTrain and SimulTrain, which use multiple loss functions, show a significant improvement in coverage compared to SingleFix in most settings. The fact that there is a large difference in coverage between SingleFix and the proposed method, even though the final model size is exactly the same, indicates that the use of multiple loss functions facilitates the learning of complex distributions.

2) EFFECT OF SEQUENTIAL TRAINING

Next, we confirmed the effectiveness of the proposed sequential training. The coverage of SimulTrain is remarkably low when $\alpha = 1.6$, which is the rate of increase in the penalty coefficient γ_k. This is attributable to the fact that the coefficient of the loss function γ_k is too large for the models after the middle of the connected models. The coefficients γ_k are designed based on the assumption that the value of the penalty term decreases as the method progresses to later models. However, there is no guarantee that the penalty term of later models will be smaller because the mapping "moving the input slightly in the direction of $\mathbb{X}$" is not learned in the initial stages of SimulTrain training. In other words, the penalty term of the latter model is likely to be dominant, and mode collapse will occur at this time, as described in Section III. After this occurs, because it cannot be repaired, the coverage converges to 0.02 until the end without recovering.

However, because SeqTrain trains each model sequentially, the possibility of such a problem occurring is low. The results

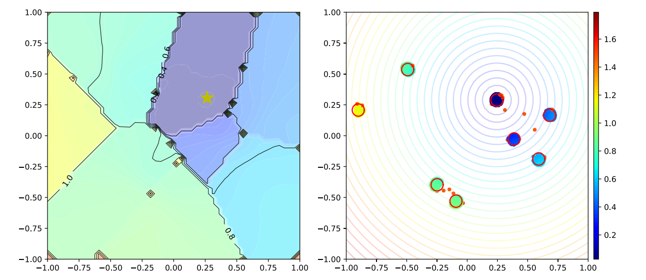
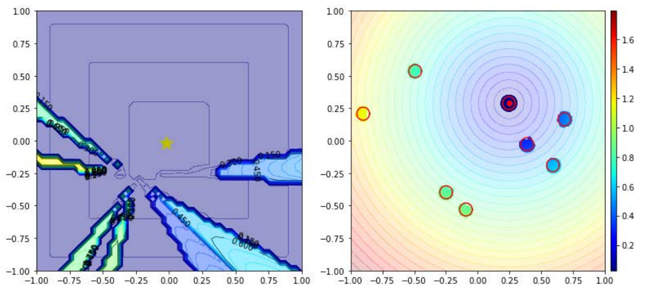
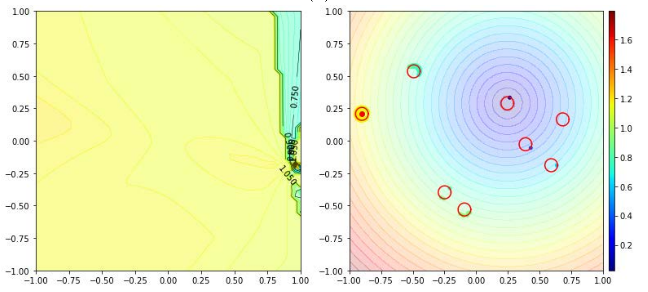


FIGURE 3. Visualization of the decoder trained using SeqTrain. Left: Contours and heat map of $f(G(z))$ on $\mathbb{Z}$. Right: Image of $G(z)$ when z is sampled evenly over $\mathbb{Z}$ at $50 \times 50 = 2500$ points. The interior of the red circle represents the feasible domain and the orange points represent the points projected onto the infeasible domain. The color bars represent the objective function values of the points projected onto the feasible domain.



(a)



(b)

FIGURE 4. Visualization of decoders obtained by HM: (a) reference point is the same as the optimal solution; (b) reference point is located in a feasible subdomain where the optimal solution does not exist. (Please see the caption of Figure 3 for details.).

of SeqTrain confirm that SimulTrain can train approximately 80% of the regions when the coverage is set to 0.02. However, there is a decrease in the coverage compared to $\alpha = 1.3$; therefore, although it is relatively robust to the increasing rate α, there is a possibility of mode collapse if it is too large.

3) EFFECT ON OPTIMIZATION RESULTS

The feasibility and coverage of the models trained using each method and the success rate after 20 optimizations of the problem using the trained models are shown in Table 3.

The success rate of BruteForce was 0.18, indicating that optimizing the test problem directly was difficult.[2] On the

[2]Although the performance of CMA-ES with ε-ordering (BruteForce) was not competitive in this experiment, it was better than that of other penalty function-based approaches, such as augmented Lagrangian constraint handling [6] and the penalty function method with manually tuned penalty coefficient.

TABLE 2. Results of a typical single trial obtained during three trials in each setting. A similar trend was observed in all trials. α and γ_0 in SeqTrain and SimulTrain, respectively, are parameters that determine the penalty coefficient $\gamma_k = \gamma_0 \alpha^{k-1}$ in the loss function L_k (Equation (5)) of each sub-model G_{θ_k}. The parameter γ in SingleFix represents the penalty coefficient in the loss function (Equation (3)).

Method	**SeqTrain**				**SimulTrain**				**SingleFix**		
α	1.3		1.6		1.3		1.6		$\gamma = 5$	$\gamma = 10$	$\gamma = 20$
γ_0	0.03	0.1	0.03	0.1	0.03	0.1	0.03	0.1			
feasibility	0.95	0.98	0.96	0.98	0.79	0.89	0.92	0.92	0.09	0.39	0.60
coverage	1.0	0.99	0.81	0.73	1.0	1.0	0.02	0.02	0.06	0.01	0.01

other hand, the proposed method achieved a success rate as high as 0.7, indicating that the search domain was transformed using decoder G, which made the optimization process easier.

The comparison of the success rates between the methods confirms that the proposed method achieves the highest value. This means that the search domain transformed using the proposed decoder G can be transformed into a space that is easiest (among the compared methods) for the optimization method to search. A visualization of the acquired G is shown in Figure 3, which shows that the proposed method acquires a Voronoi-like map, where the center of each disconnected feasible domain is the representative point. This indicates that the design principle of the proposed method, which states that the nearest z should be projected onto the nearest $\boldsymbol{x}$, has been realized.

Although the feasibility and coverage of the model obtained by the decoder with SVDD [12] are comparable to those of the proposed method, the success rate is zero. This is because the output $G(\mathbb{Z})$ of the decoder G is degenerate. When the decoder has numerous regions to learn, the output of each region tends to converge to a point in exchange for learning the numerous regions. Consequently, even if it learns the region where the optimal solution is located, it cannot reach the area around the optimal solution.

The results of the decoder with HM [35] show that while the feasibility is 1, the coverage is 0.01, meaning that only one disconnected region of the feasible domain has been learned. In this method, $G(z)$ is calculated via a line search between the feasible solution and latent variable z, which is called the reference point; thus, a feasible solution is always obtained. However, when the feasible domain is disconnected and the measure of the feasible domain is very small compared to the search domain, as in this experiment, it is extremely difficult to find a feasible domain other than the region with the reference point using a line search. This phenomenon is illustrated in Figure 4. Figure 4 shows a heat map of $f(G(\mathbb{Z}))$ for different reference points in a problem with dimensionality $n = 2$, domain $L = 8$, and optimal solution of the objective function $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_f^*\|^2$, where $\boldsymbol{x}_f^* = [0.25, 0.25]^{\mathrm{T}}$ for the different reference points. Figure 4a, where the reference point is the same as the optimal solution of $[0.25, 0.25]^{\mathrm{T}}$, confirms that feasible solutions are obtained for regions other than the region where the reference point is located. However, in Figure 4b, where the reference point is different from the optimal solution, most of the points in z are projected onto the region where the reference point is located. In other words, in the 10-dimensional experiment, the coverage was 0.01 because only feasible solutions were obtained in the region where the reference point was located, and the success rate was zero because the reference point did not coincide with the region where the optimal solution was located.

TABLE 3. Feasibility and coverage of the decoder G determined by each method and the success rate of optimizing the problem using the decoder G. BruteForce is the result of direct optimization without using the decoder G. For SeqTrain, the model shown in Table 2 was used among the models obtained from three decoder training trials. The success rates were 0.8 and 0.7 when the other two models were used.

	SeqTrain	HM	SVDD	BruteForce
feasibility	0.95	1.0	0.76	-
coverage	0.99	0.01	0.96	-
success rate	0.7	0.0	0.0	0.18

V. APPLICATION TO TOPOLOGY OPTIMIZATION PROBLEM

The experiment outlined in this section was conducted to confirm the effectiveness of the proposed method in transforming optimization problems by using an example TO that is closer to a real-world problem. As mentioned in Section I, TO with volume fraction and perimeter constraints is expected to have disconnected feasible subdomains.

A. MBB BEAM

In the experiment outlined in this section, we used the Python implementation of a classic problem in topology optimization: the symmetric MBB beam [1], [49]. Figure 5 shows the design domain, anchorage of the structure, and location of the external force. The objective of this problem is to determine a structure in a finite design domain that minimizes the distortion of the beam when an external force is applied.

Following the modified SIMP method [7], [55], which is commonly used in TO, the MBB beam was formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{\rho}} f(\rho) &= \boldsymbol{U}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{U} = \sum_e \boldsymbol{u}_e^{\mathrm{T}}\boldsymbol{k}_e\boldsymbol{u}_e \\ \text{s.t. } \boldsymbol{K}\boldsymbol{U} &= \boldsymbol{F} \\ g &= \frac{V(\boldsymbol{\rho})}{V*} - 1 = \sum_e \frac{v_e \rho_e}{V*} - 1 \leqslant 0 \\ 0 &\leqslant \rho_e \leqslant 1, \end{aligned}$$

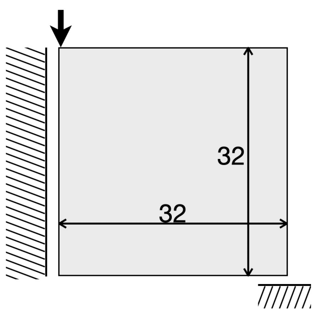


FIGURE 5. Symmetric MBB-beam design area, fixation points, and external force locations.

where $\boldsymbol{K}$, $\boldsymbol{U}$, and $\boldsymbol{F}$ denote the overall stiffness matrix, displacement vector, and external force vector, respectively. The subscript e refers to each element, where $\boldsymbol{k}_e = \boldsymbol{k}(\rho_e) = E(\rho_e)\boldsymbol{k}_e^0$ is the element's stiffness matrix and $\boldsymbol{k}_e^0$ is the element's stiffness matrix for the unit Young's modulus. $\boldsymbol{\rho}$ is the design variable, and each element $\rho_e \in [0, 1]$ represents the density. $V(\boldsymbol{\rho})$ is the volume of the structure, and V^* is the upper bound of the volume. The first constraint implies that the structure is not broken by external forces, the second is a volume constraint, and the third is an upper- and lower-bound constraint. The Young's modulus for each element of the design variables is defined by

$$E(\rho_e) = E_{\min} + \rho_e^p(E_0 - E_{\min}), \tag{6}$$

where p is the penalty factor, $E_{\min} = 10^{-9}$ is a value that prevents the calculation from failing if the element is empty, and $E_0 = 1$ is the Young's modulus of the solid.

In this experiment, the size of the design domain was set to 32×32, and the design variable was set to $\boldsymbol{x} \in \mathbb{S} = [-1, 1]^n$ during optimization using CMA-ES, which was converted to $[0, 1]^{32\times 32}$ during the evaluation of the objective function.

In addition to the preceding formulation, we imposed a perimeter constraint defined by

$$g(\boldsymbol{x}) := Q(\mathrm{Sign}(\boldsymbol{x})) - Q_{\mathrm{th}},$$

where $Q : \{-1, 1\}^n \to \mathbb{R}^+$ is the perimeter length, Q_{th} is the upper limit, and $\mathrm{Sign} : [-1, 1]^n \to \{-1, 1\}^n$ is a function that converts each element of $\boldsymbol{x}$ to -1 if it is less than or equal to zero, and 1 otherwise. The perimeter length constraint imposes a limit on the perimeter length of a structure, which reduces its complexity to the extent that it can be manufactured [4], [24]. We set $Q_{\mathrm{th}} = 256$, and the upper limit of the volume constraint was set such that the ratio of the volume of the structure to that of the entire design domain ($= 1024$) was 0.2.

B. EXPERIMENTAL SETTING

In this experiment, we used convolutional neural networks (CNNs) as the decoder G. The specifications of the network architecture are presented in Table 4. The number of models was set to $K = 10$, the initial value of the coefficients of the constraint term was $\gamma_0 = 0.1$, and the increasing

TABLE 4. Network architecture specifications. The decoder was the composite of K sub-models, $G_\theta = G_{\theta_K} \circ \cdots \circ G_{\theta_1}$, and each sub-model was $G_{\theta_k}(x) = x + G^{\mathrm{res}}_{\theta_k}(x)$.

The kernel size of the Conv and Deconv layers was 5×5. BN denotes batch normalization. Data were input to each layer after both BN and activation were applied.

Kernel	BN	Activation	Resample	Output Shape
Residual block $G^{\mathrm{res}}_{\theta_k} : \mathbb{R}^n \to [-1,1]^{32\times 32}$				
Dense	✓	Mish	-	$4 \times 4 \times 64$
Conv	✓	Mish	Same	$4 \times 4 \times 512$
Deconv	✓	Mish	Up	$8 \times 8 \times 256$
Deconv	✓	Mish	Up	$16 \times 16 \times 128$
Deconv	✓	Mish	Up	$32 \times 32 \times 1$
-	-	Tanh	Same	$32 \times 32 \times 1$
Constraint Surrogate Model $\mathcal{V} : [-1,1]^{32\times 32} \to \mathbb{R}$				
Conv	✓	ReLU	Same	$32 \times 32 \times 32$
Conv	✓	ReLU	Down	$16 \times 16 \times 64$
Conv	✓	ReLU	Down	$8 \times 8 \times 128$
Conv	✓	ReLU	Down	$4 \times 4 \times 256$
Dense	-	-	-	1

coefficient was $\alpha = 1.3$. ADAM [33] was used for training, with a mini-batch size of 64, a learning rate of 0.0001, and $\beta_1 = 0.3$. A uniform distribution $U(\mathbb{S})$ was used as the generating distribution P_z of z. The number of updates for each sub-model G_{θ_k} in the training was set to 100000 iterations. The training data collection method for SVDD and HM was the same as that for Section IV.

Because the perimeter constraint is calculated by converting the design variables to binary values, the gradient becomes zero everywhere; this means that the training of the model will not proceed. To avoid this, we used a proxy model $\mathcal{V} : \mathbb{R}^n \to \mathbb{R}^+$, which approximates the perimeter function Q. The surrogate model $\mathcal{V}$ was trained to minimize the mean squared error

$$\text{minimize} : \mathbb{E}_{z\sim P_z}\left[(\mathcal{V}(G(z)) - Q(G(z)))^2\right].$$

The proxy model $\mathcal{V}$ was updated once before updating G_{θ_k} in each iteration.

CMA-ES optimization using the trained model was performed for 20 trials for SeqTrain and three trials for SVDD and HM. For each trial, the mean vector, step size, and covariance matrix were initialized as $\boldsymbol{m}^{(0)} \sim U(\mathbb{S})$, $\sigma^{(0)} = 2/5 = 0.4$, and $\boldsymbol{C}^{(0)} = \boldsymbol{I}_n$, respectively. The end condition of each trial was set to be when the maximum number of evaluations of the objective function reached 200000. For other CMA-ES settings, we used the values recommended in [26]. To handle infeasible solutions, CMA-ES and ε-ordering were combined for optimization, as in the experiment in Section IV. As a baseline, we show the results of 20 trials of optimization using CMA-ES with ε-ordering without a decoder (BruteForce).

One of the weaknesses of decoder-based optimization is that it is impossible to obtain a solution that is not included in the image of the trained decoder. In this experiment, after optimizing the decoder obtained by SeqTrain, we also

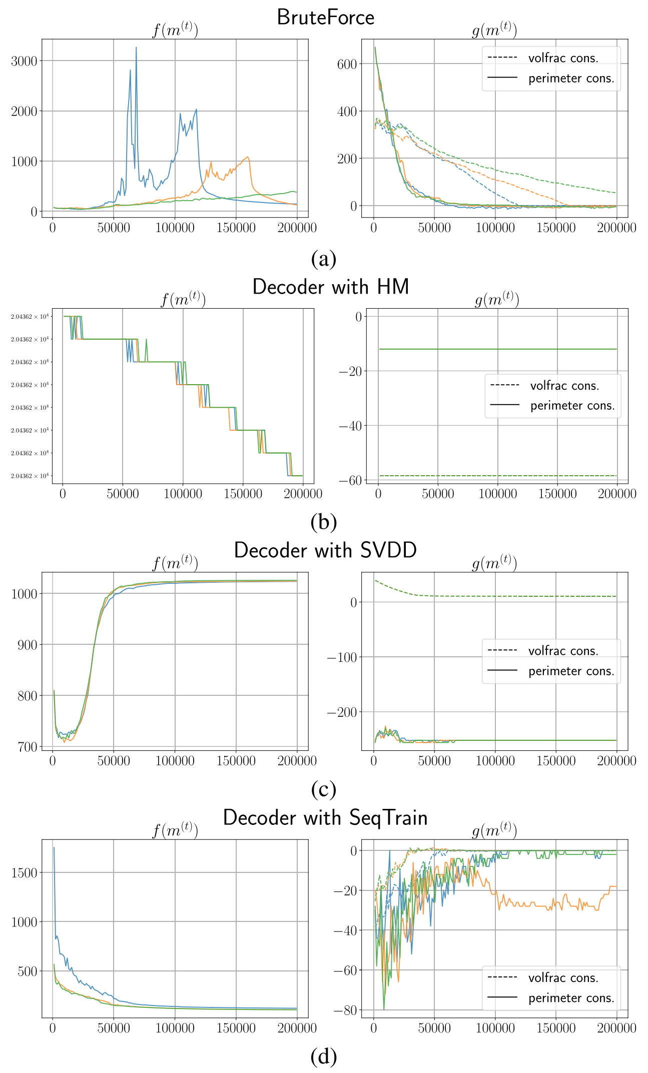


FIGURE 6. Three-trial behavior of the (left) objective function $f(m^{(t)})$ and (right) constraint function $g(m^{(t)})$ on the mean vector $m^{(t)}$ during optimization using (a) BruteForce, (b) Decoder with HM, (c) Decoder with SVDD, and (d) Decoder with SeqTrain (proposed). The same color indicates the same trial. The horizontal axis represents the number of evaluations performed.

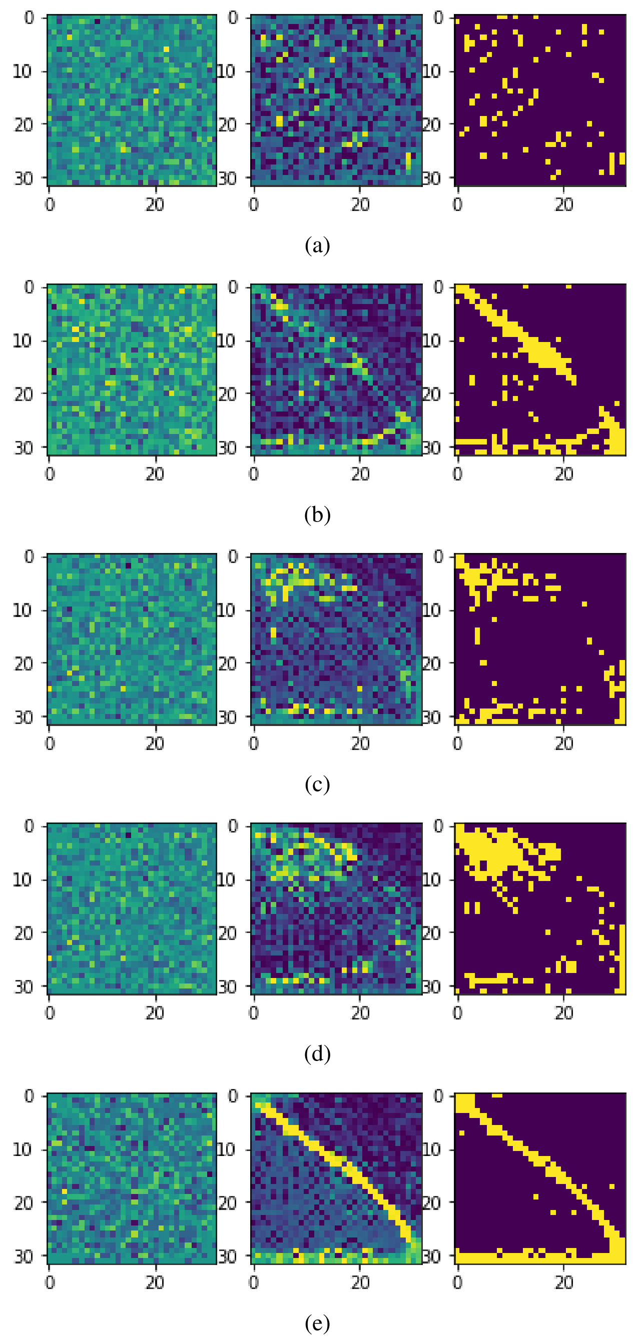


FIGURE 7. Visualization of the optimization process using the proposed method's decoder, G. First column: mean vector $m^{(t)}$; second column: $x_{\text{map}} = G(m^{(t)})$; third column: Sign($x_{\text{map}}$), at iteration (a) $t = 0$, (b) 100, (c) 200, (d) 300, and (e) at the end of optimization. The displayed image is a 1D vector transformed to 32 × 32.

performed hybrid optimization using BruteForce on the obtained results. First, a solution search was performed on the decoder for half the maximum number of evaluations. The resulting solution was then locally searched by BruteForce, with this solution taken to be the initial mean vector $\boldsymbol{m}^{(0)}$ with a small initial step size $\sigma^{(0)} = 0.01$. This hybrid approach searches for a feasible solution with a superior objective function value on a global scale using a decoder, and it refines the solution locally in the latter half of the process without being bound by the expressive power of the decoder.

C. RESULTS AND DISCUSSION

Table 5 summarizes the objective function values $f(\boldsymbol{m}_{\text{last}})$ in the mean vector obtained after MBB-beam optimization. The SIMP method is a gradient-based method widely used in topology optimization. We implemented the SIMP solver published at the same URL [1] as the MBB-beam publication.

The results in Table 5 confirm that the minimization of the objective function does not proceed adequately in the optimization using SVDD or HM. The behavior of the volfrac and perimeter constraints for each optimization process illustrated in Figure 6 exhibits almost no change. Because the

TABLE 5. Average value of the objective function $f(m_{\text{last}})$ at the mean vector m_{last} of CMA-ES obtained after the optimization of the MBB beam. The mean and standard deviation of 20 trials (all of which obtained feasible solutions) are shown for SeqTrain and Hybrid, the mean and standard deviation of 16 out of 20 trials in which feasible solutions were obtained are shown for BruteForce, and the means of three trials are shown for SVDD and HM. The mean and standard deviation at the end of the first stage in Hybrid were 123.6 and 8.98, respectively.

	BruteForce	SVDD	HM	SIMP	SeqTrain	Hybrid
$f(m_{\text{last}})$	109.3 (56.0)	1024.6	20436.2	115.5	115.2 (16.1)	62.3 (3.25)

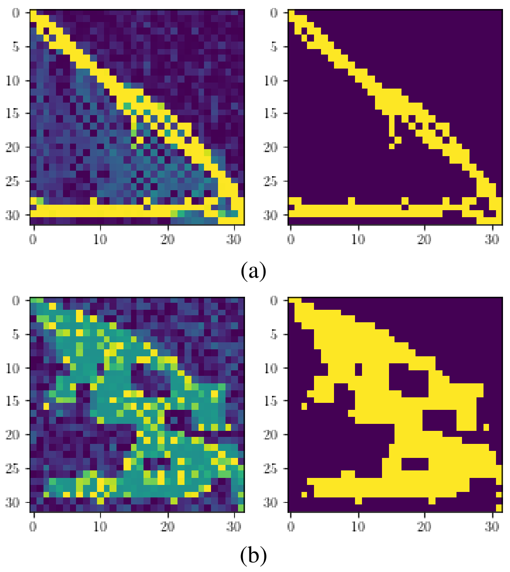


FIGURE 8. Visualization of the mean vectors obtained by BruteForce: (a) trial converged to a feasible solution; and (b) trial converged to an infeasible solution. Left column: m_{last}; right column: Sign(m_{last}).

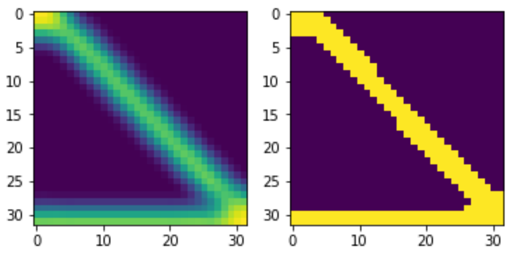


FIGURE 9. Visualization of the final optimization solution obtained using SIMP. Left: solution x; right: Sign(x).

same behavior is observed in all three trials, it is highly likely that the output of the decoder G is limited to a very small area, which is why the optimization does not proceed. As discussed in Section IV-C, when the number of disconnected feasible domains in SVDD or HM is large, it is difficult to learn an appropriate decoder, which makes them unsuitable for this type of problem.

The results of BruteForce and SeqTrain in Table 5 confirm that both methods obtain the same or better objective function value compared with that of the SIMP topology optimization method. Note, however, that BruteForce converged outside of the constraint in 4 out of 20 trials; hence, only the average value of 16 trials is shown. The behavior of the optimization process for the three trials of BruteForce in Figure 6 indicates that one trial continued to violate the volumetric (volfrac) constraint. This indicates that the method is converging to outside the constraint. It can be seen that in the failed trials, the clumps are relatively large. To satisfy the volume constraint in this state, it is necessary to decrease each element. However, when a hole is created in the clump, the perimeter length increases rapidly, and such a solution is difficult to select in the optimization process. In other words, it is difficult to escape this situation when the search distribution is small. On the other hand, in a search using a decoder G as in the proposed method, the search domain is generally a feasible domain, so the possibility of convergence to an unconstrained local solution is extremely low. We can also confirm the advantage of the decoder-based approach over BruteForce in terms of the number of f-calls mentioned in Section II. BruteForce did not locate a feasible solution before 10^5 f-calls, whereas the proposed approach already located a feasible solution with a good f-value before 10^5 f-calls.

The advantage of the proposed decoder-based optimization method is best observed in the hybrid method. Although SeqTrain can stably obtain feasible solutions, it cannot search for solutions that cannot be represented by the decoder; therefore, it is difficult to reduce the objective function value even if the search is continued after obtaining a good global structure. On the other hand, in BruteForce, if a feasible domain with a good global structure can be found, a small objective function value can be obtained; however, the rate of finding such a feasible domain is small, and the optimization results show a large variation from trial to trial. When a hybrid method is used, a stable and excellent global structure is found, and the objective function value can be reduced by subsequent refinement.

VI. CONCLUSION

In this study, we proposed a decoder-based constraint-handling method that transforms an explicitly constrained BBO problem (Equation (1)), where the feasible domain $\mathbb{X}$ is a union of several disconnected regions, into an optimization problem (Equation (2)), where the search domain is feasible almost everywhere. We designed a method to represent the mapping (decoder) from the latent space $\mathbb{Z}$, which is a convex set, to the disconnected feasible domain $\mathbb{X}$ using a deep neural network, and to learn it without using training data. In the proposed method, the decoder is represented as a composite of multiple sub-models, each of which has skip connections. The loss function of each sub-model is defined independently, and the sub-models are trained sequentially. Numerical experiments using a test problem and a topology optimization problem show that the proposed method can find feasible domains with better objective function values and higher probability than both conventional decoder-based constraint-handling methods and non-decoder-based constraint-handling methods.

We recommend the hybrid method, as shown in Section V, as a way to use the decoder proposed in this study. First, the decoder was trained using the proposed method. Because the training of the decoder does not require the evaluation of the objective function value, which is computationally expensive, we can obtain a decoder G^* with sufficient performance by adjusting the training parameters using the ratio of feasible solutions as an indicator. Next, the trained decoder G^* was used to solve the minimization problem defined by Equation (2) to obtain the solution z^*. This allowed us to globally search for a feasible domain with a good objective function value. However, because the decoder does not necessarily represent the entire feasible domain, there is room for improvement. Finally, we locally solved the minimization problem defined in Equation (1), with $G^*(z^*)$ as the initial solution, to refine the solution within the feasible domain where $G^*(z^*)$ exists. The topology optimization results (Section V) show that the hybrid method can stably obtain a feasible solution with a lower objective function value than optimizing Equation (1) or Equation (2) alone.

There are three elements in the proposed method that must be determined by the user. The first is the initial value, γ_0, of the loss function's penalty term's coefficient γ_k and its rate of increase, α. As shown in Section IV-C, if γ_k is larger than necessary, mode collapse may occur, and G may learn only a part of the region. To prevent this, it is recommended that the initial value γ_0 and the rate of increase α be minimized. However, as such values depend on the scale of the constraint function, investigating the guidelines for determining these hyperparameters will contribute to performance improvement. The second is the architecture of G. To obtain a better G, it is desirable to select an architecture that considers the characteristics of the task. In the experiments in this study, we chose a network with fully connected layers for a toy problem and a network with convolutional layers for a topology optimization problem. For the latter, we used prior knowledge that the design variable in our topology optimization problem is naturally treated as a 2D grayscale image and that convolutional layers are known to excel at handling image formats in the machine learning field. Unless such prior knowledge is available, fully connected layers with number of nodes a few times greater than the input dimension n may be the default choice. Choosing a reasonable network architecture may sometimes require expert knowledge in neural networks. The third is the termination condition of the training. In the experiments conducted in this study, the end condition of each connected model was unified at 200k iterations. However, for different tasks and models, this may result in inadequate training. Currently, only the convergence curve of the loss function and the feasibility ratio of the model output guide the termination condition, and these need to be determined by the user through trial-and-error. However, this adjustment requires practical experience in deep learning, which is a weakness of the proposed method. If there is an index to evaluate the goodness of the model, such as the spread and continuity of the model output, the adjustment of the proposed method will be easier and the quality of the acquired model will be improved. These are important issues to be addressed in the future.

REFERENCES

[1] *Topology Optimization Codes Written in Python*. Accessed: Nov. 6, 2022. [Online]. Available: https://www.topopt.mek.dtu.dk/apps-and-software/topology-optimization-codes-written-in-python

[2] Y. Akimoto and N. Hansen, "Diagonal acceleration for covariance matrix adaptation evolution strategies," *Evol. Comput.*, vol. 28, no. 3, pp. 405–435, 2019.

[3] Y. Akimoto, N. Sakamoto, and M. Ohtani, "Multi-fidelity optimization approach under prior and posterior constraints and its application to compliance minimization," in *Proc. Parallel Problem Solving From Nature (PPSN)*, 2020, pp. 81–94.

[4] L. Ambrosio and G. Buttazzo, "An optimal design problem with perimeter penalization," *Calculus Variat. Partial Differ. Equ.*, vol. 1, no. 1, pp. 55–69, Mar. 1993.

[5] A. Arias-Montano, C. A. C. Coello, and E. Mezura-Montes, "Multi-objective evolutionary algorithms in aeronautical and aerospace engineering," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 662–694, Oct. 2012.

[6] A. Atamna, A. Auger, and N. Hansen, "Augmented Lagrangian constraint handling for CMA-ES—Case of a single linear constraint," in *Proc. Parallel Problem Solving From Nature (PPSN)*, 2016, pp. 181–191.

[7] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Struct. Optim.*, vol. 1, no. 4, pp. 193–202, 1989.

[8] M. P. Bendsøe, *Optimization of Structural Topology, Shape, and Material*. Berlin, Germany: Springer, 1995.

[9] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method," *Comput. Method Appl. Mech. Eng.*, vol. 71, no. 2, pp. 197–224, 1988.

[10] P. Bontrager, A. Roy, J. Togelius, N. Memon, and A. Ross, "DeepMasterPrints: Generating MasterPrints for dictionary attacks via latent variable evolution," in *Proc. IEEE 9th Int. Conf. Biometrics Theory, Appl. Syst. (BTAS)*, Oct. 2018, pp. 1–9.

[11] Z. Bouzarkouna, D. Y. Ding, and A. Auger, "Well placement optimization with the covariance matrix adaptation evolution strategy and meta-models," *Comput. Geosci.*, vol. 16, no. 1, pp. 75–92, Jan. 2012.

[12] J. Bremer and M. Sonnenschein, "Constraint-handling for optimization with support vector surrogate models—A novel decoder approach," in *Proc. 5th Int. Conf. Agents Artif. Intell.*, vol. 2, 2013, pp. 91–100.

[13] P. Chrabszcz, I. Loshchilov, and F. Hutter, "Back to basics: Benchmarking canonical evolution strategies for playing atari," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 1419–1426.

[14] A. Díaz and O. Sigmund, "Checkerboard patterns in layout optimization," *Struct. Optim.*, vol. 10, no. 1, pp. 40–45, Aug. 1995.

[15] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–13.

[16] Y. Dong, H. Su, B. Wu, Z. Li, W. Liu, T. Zhang, and J. Zhu, "Efficient decision-based black-box adversarial attacks on face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7706–7714.

[17] G. Fujii and Y. Akimoto, "Topology-optimized thermal carpet cloak expressed by an immersed-boundary level-set method via a covariance matrix adaptation evolution strategy," *Int. J. Heat Mass Transf.*, vol. 137, pp. 1312–1322, Jul. 2019.

[18] G. Fujii and Y. Akimoto, "Electromagnetic-acoustic biphysical cloak designed through topology optimization," *Opt. Exp.*, vol. 30, no. 4, pp. 6090–6106, 2022.

[19] G. Fujii, M. Takahashi, and Y. Akimoto, "CMA-ES-based structural topology optimization using a level set boundary expression—Application to optical and carpet cloaks," *Comput. Methods Appl. Mech. Eng.*, vol. 332, pp. 624–643, Apr. 2018.

[20] G. Fujii, M. Takahashi, and Y. Akimoto, "Acoustic cloak designed by topology optimization for acoustic–elastic coupled systems," *Appl. Phys. Lett.*, vol. 118, no. 10, Mar. 2021, Art. no. 101102.

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol 27, 2014, pp. 2672–2680.

[22] J. Gurrola-Ramos, A. Hernandez-Aguirre, and O. Dalmau-Cedeno, "COLSHADE for real-world single-objective constrained optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2020, pp. 1–8.

[23] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 2455–2467.

[24] R. B. Haber, C. S. Jog, and M. P. Bendsøe, "Variable-topology shape optimization with a control on perimeter," in *Proc. 20th Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, Sep. 1994, pp. 261–272.

[25] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[26] N. Hansen, "The CMA evolution strategy: A tutorial," 2016, *arXiv:1604.00772*.

[27] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proc. 12th Annu. Conf. Companion Genetic Evol. Comput.*, 2010, pp. 1689–1696.

[28] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions," INRIA, Rennes, France, Tech. Rep., RR-6829, 2009.

[29] N. Hansen and S. Kern, "Evaluating the CMA evolution strategy on multimodal test functions," in *Proc. Parallel Problem Solving From Nature (PPSN)*, 2004, pp. 282–291.

[30] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, Mar. 2003.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[32] G. A. Jastrebski and D. V. Arnold, "Improving evolution strategies through active covariance matrix adaptation," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 9719–9726.

[33] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

[35] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evol. Comput.*, vol. 7, no. 1, pp. 19–44, 1999.

[36] I. Kriest, V. Sauerland, S. Khatiwala, A. Srivastav, and A. Oschlies, "Calibrating a global three-dimensional biogeochemical ocean model (MOPS-1.0)," *Geosci. Model Dev.*, vol. 10, no. 1, p. 127, 2017.

[37] A. Kumar, S. Das, and I. Zelinka, "A modified covariance matrix adaptation evolution strategy for real-world constrained optimization problems," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2020, pp. 11–12.

[38] A. Kumar, S. Das, and I. Zelinka, "A self-adaptive spherical search algorithm for real-world constrained optimization problems," in *Proc. Genetic Evol. Comput. Conf. Companion*, Jul. 2020, pp. 13–14.

[39] M. Lunacek, D. Whitley, and A. Sutton, "The impact of global structure on search," in *Proc. Parallel Problem Solving From Nature (PPSN)*, 2008, pp. 498–507.

[40] P. MacAlpine, S. Barrett, D. Urieli, V. Vu, and P. Stone, "Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1053–1147.

[41] A. Maki, N. Sakamoto, Y. Akimoto, H. Nishikawa, and N. Umeda, "Application of optimal control theory based on the evolution strategy (CMA-ES) to automatic berthing," *J. Mar. Sci. Technol.*, vol. 25, no. 1, pp. 221–233, Mar. 2020.

[42] A. L. Marsden, M. Wang, J. E. Dennis, Jr., and P. Moin, "Optimal aeroacoustic shape design using the surrogate management framework," *Optim. Eng.*, vol. 5, no. 2, pp. 235–262, Jun. 2004.

[43] E. Mezura-Montes and C. A. Coello-Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, Dec. 2011.

[44] D. Misra, "Mish: A self regularized non-monotonic activation function," in *Proc. 31st Brit. Mach. Vis. Conf.*, 2020, pp. 1–14.

[45] A. Miyagi, Y. Akimoto, and H. Yamamoto, "Well placement optimization under geological statistical uncertainty," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 1284–1292.

[46] Y. Miyauchi, R. Sawada, Y. Akimoto, N. Umeda, and A. Maki, "Optimization on planning of trajectory and control of autonomous berthing and unberthing for the realistic port geometry," *Ocean Eng.*, vol. 245, Feb. 2022, Art. no. 110390.

[47] N. Sakamoto, E. Semmatsu, K. Fukuchi, J. Sakuma, and Y. Akimoto, "Deep generative model for non-convex constraint handling," in *Proc. Genetic Evol. Comput. Conf.*, 2020, pp. 636–644.

[48] T. Sato, K. Watanabe, and H. Igarashi, "Multimaterial topology optimization of electric machines based on normalized Gaussian network," *IEEE Trans. Magn.*, vol. 51, no. 3, pp. 1–4, Mar. 2015.

[49] O. Sigmund, "A 99 line topology optimization code written in MATLAB," *Struct. Multidisciplinary Optim.*, vol. 21, no. 2, pp. 120–127, Apr. 2001.

[50] T. Takahama and S. Sakai, "Constrained optimization by the ϵ constrained differential evolution with gradient-based mutation and feasible elites," in *Proc. IEEE Int. Conf. Evol. Comput.*, Jul. 2006, pp. 1–8.

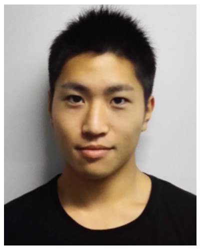
[51] K. Terayama, M. Sumita, R. Tamura, and K. Tsuda, "Black-box optimization for automated discovery," *Accounts Chem. Res.*, vol. 54, no. 6, pp. 1334–1346, Mar. 2021.

[52] J. Uhlendorf, A. Miermont, T. Delaveau, G. Charvin, F. Fages, S. Bottani, G. Batt, and P. Hersen, "Long-term model predictive control of gene expression at the population and single-cell levels," *Proc. Nat. Acad. Sci. USA*, vol. 109, no. 35, pp. 14271–14276, Aug. 2012.

[53] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi, "Evolving Mario levels in the latent space of a deep convolutional generative adversarial network," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2018, pp. 221–228.

[54] M. Y. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 192, nos. 1–2, pp. 227–246, Jan. 2003.

[55] M. Zhou and G. I. N. Rozvany, "The COC algorithm. Part II: Topological, geometrical and generalized shape optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 89, nos. 1–3, pp. 309–336, Aug. 1991.



NAOKI SAKAMOTO received the M.S. degree in electrical and computer engineering from Shinshu University, in 2018, and the Ph.D. degree in computer science from University of Tsukuba, Japan, in 2022. From 2019 to 2022, he was a Research Fellow of JSPS in Japan. His research interest includes applications of stochastic search heuristics.

REI SATO received the B.S. degree in information engineering and the M.S. degree in engineering from University of Tsukuba, Japan, in 2020 and 2022, respectively. From 2018 to 2022, he was a Trainee at RIKEN AIP, Japan. His research interests include neural architecture search and reinforcement learning.

KAZUTO FUKUCHI received the Ph.D. degree from the University of Tsukuba, Japan, in 2018. Since 2019, he has been an Assistant Professor with the Faculty of Engineering, Information and Systems, University of Tsukuba, and a Visiting Researcher with the Center for Advanced Intelligence Project, RIKEN, Japan. His research interests include mathematical statistics, machine learning, and their application.

JUN SAKUMA received the Ph.D. degree in engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 2003. From 2003 to 2004, he was a Researcher at the Tokyo Research Laboratory, IBM, Tokyo. From 2004 to 2009, he was an Assistant Professor at the Department of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology. From 2009 to 2016, he was an Associate Professor with the Department of Computer Science, University of Tsukuba. Since 2016, he has been a Professor with the Department of Computer Science, School of Systems and Information Engineering, University of Tsukuba, Japan. He has been the Team Leader of the Artificial Intelligence Security and Privacy Team, Center for Advanced Intelligence Project, RIKEN, since 2016. His research interests include data mining, machine learning, data privacy, and security. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan (IEICE).

YOUHEI AKIMOTO received the B.S. degree in computer science in 2007, and the M.S. and Ph.D. degrees in computational intelligence and systems science from the Tokyo Institute of Technology, Japan, in 2008 and 2011, respectively. From 2010 to 2011, he was a Research Fellow of JSPS in Japan, and from 2011 to 2013, he was a Postdoctoral Research Fellow at INRIA in France. From 2013 to 2018, he was an Assistant Professor at Shinshu University, Japan. Since 2018, he has been an Associate Professor with University of Tsukuba, Japan, and a Visiting Researcher with the Center for Advanced Intelligence Projects, RIKEN. His research interests include design principles, theoretical analyses, and applications of stochastic search heuristics. He served as a Track Chair for the continuous optimization track of GECCO in 2015 and 2016. He is an Associate Editor of *ACM TELO* and is on the Editorial Board of *ECJ*. He received the Best Paper Award from GECCO 2018 and FOGA 2019.