mathematics

MDPI

Article

3D Vase Design Based on Interactive Genetic Algorithm and Enhanced XGBoost Model

Dongming Wang and Xing Xu *

School of Physics and Information Engineering, Minnan Normal University, Zhangzhou 363000, China; g2022091010@stu.mnnu.edu.cn
* Correspondence: xx1889@mnnu.edu.cn

Abstract: The human–computer interaction attribute of the interactive genetic algorithm (IGA) allows users to participate in the product design process for which the product needs to be evaluated, and requiring a large number of evaluations would lead to user fatigue. To address this issue, this paper utilizes an XGBoost proxy model modified by particle swarm optimization and the graphical interaction mechanism (GIM) to construct an improved interactive genetic algorithm (PXG-IGA), and then the PXG-IGA is applied to 3D vase design. Firstly, the 3D vase shape has been designed by using a bicubic Bézier surface, and the individual genetic code is binary and includes three parts: the vase control points, the vase height, and the texture picture. Secondly, the XGBoost evaluation of the proxy model has been constructed by collecting user online evaluation data, and the particle swarm optimization algorithm has been used to optimize the hyperparameters of XGBoost. Finally, the GIM has been introduced after several generations, allowing users to change product styles independently to better meet users' expectations. Based on the PXG-IGA, an online 3D vase design platform has been developed and compared to the traditional IGA, KD tree, random forest, and standard XGBoost proxy models. Compared with the traditional IGA, the number of evaluations has been reduced by 58.3% and the evaluation time has been reduced by 46.4%. Compared with other proxy models, the accuracy of predictions has been improved up from 1.3% to 20.2%. To a certain extent, the PXG-IGA reduces users' operation fatigue and provides new ideas for improving user experience and product design efficiency.

Keywords: XGBoost; interactive genetic algorithm; 3D vase shape; Bézier surface

MSC: 68W50

Citation: Wang, D.; Xu, X. 3D Vase Design Based on Interactive Genetic Algorithm and Enhanced XGBoost Model. *Mathematics* **2024**, *12*, 1932. https://doi.org/10.3390/math12131932

Academic Editor: Alma Y. Alanis

Received: 17 May 2024
Revised: 16 June 2024
Accepted: 18 June 2024
Published: 21 June 2024

1. Introduction

As a distinctive craft and utility item, ceramics have held significant importance throughout the history of human civilization. From the earliest earthenware to the later exquisite porcelain, the narrative of ceramics is full of stories and legends. In ancient civilizations, ceramics were essential in everyday life, serving as vessels for food and other necessities. With the advancement of technology and the refinement of craftsmanship, ceramics diversified into a plethora of forms and functions.

With the advancement of society, people's demand for ceramics has gradually surpassed traditional styles, and they are beginning to pursue more unique designs and shapes. However, traditional vase designs often rely on manual drawing by designers with accumulated experience. While they possess a high degree of artistry, they also have certain limitations. Designers' personal aesthetics and creativity may be constrained by their own knowledge and experience, thus being unable to fully explore all possibilities in the design space. Additionally, the manual design and modification process is time-consuming and laborious, making it difficult to quickly generate a large number of design proposals with different styles and forms.

One of the key steps in ceramic design is modeling, which involves transforming creative ideas into actual product forms. Modeling requires mathematically representing the design's appearance and structure, often using mathematical surfaces to describe the shape of the product, with Bézier surfaces being widely applied. Previous studies have used neural networks to control 3D models to address wound reconstruction in the medical field [1,2], providing insights into using algorithms to control modeling to solve various problems. Therefore, ceramic modeling designs combined with artificial intelligence have emerged. However, AI-generated designs may produce unexpected ceramic shapes, failing to meet the desired outcomes [3].

In this context, the interactive genetic algorithm (IGA) [4,5], as an intelligent optimization algorithm, demonstrates its unique advantages and potential. By simulating the processes of natural selection and genetic mutation, the interactive genetic algorithm is able to automatically generate and optimize design solutions. Users play the role of "selectors" in the design process, evaluating each generation of design individuals. The algorithm continuously adjusts and optimizes design solutions based on these evaluations. This interactive process not only combines human aesthetic judgment with the powerful computational capabilities of computers but also enables the exploration and discovery of innovative designs that traditional methods may find difficult to achieve in a relatively short period of time.

IGA is developed from the genetic algorithm (GA). The GA is an evolutionary optimization algorithm that can solve some problems that can be defined by mathematical formulas. The GA involves the optimization of target systems, models, and performance in multiple fields. In recent years, the GA has been applied to effective feature selection for IoT botnet attack detection [6], active disturbance rejection control of bearingless permanent magnet synchronous motors [7], gesture recognition CAPTCHA [8], state-of-charge estimation of lithium-ion batteries [9], and residential virtual power plants [10]. The application of the GA in these areas has become increasingly widespread, with more and more researchers applying the GA to solve various practical problems in their own fields.

The IGA is also an optimization algorithm. The most significant difference between the IGA and the traditional GA and other metaheuristic algorithms is that it can realize human–computer interaction. Through communication with people, the IGA can be guided in the process of evolution. Unlike the GA, the IGA is not limited to solving problems with clearly defined formulas but is also able to deal with some problems that cannot be clearly defined by formulas. Currently, the IGA has been widely applied in automatic terrain generation systems [11], 3D gaming model design [12], fashion design [13,14], and music melody composition [15], etc.

However, there are many factors that affect the IGA, such as individual knowledge reserve, personal preferences, thinking, and emotions, which can affect the fitness evaluation of the algorithm. These factors can cause fluctuations in fitness, resulting in different optimal solutions for everyone. In the IGA, the evolutionary direction of the population is uncertain. Since the user only needs to evaluate the individual's fitness, the IGA reorganizes the population characteristics according to the individual's fitness, thereby generating the next generation. Its evolutionary process does not align with the users' thinking, thus the next generation generated by such a process does not always suit the users' aesthetic tastes. Therefore, users usually need to evolve for multiple generations. In the process of small population evolution, in the IGA, due to the limited number of populations, the number of individual characteristics is also limited, which leads to the phenomenon that it is easy to produce a local optimum solution at the end of evolution. If no new population is added, the population may fall into a local optimal solution and be difficult to overcome. Each time, the population evolution needs to receive the users' fitness evaluation. However, the core issue with the IGA is that a large number of user evaluations and interactive operations may lead to the user's fatigue, Specifically, the interactive genetic algorithm requires users to evaluate and provide feedback on multiple individuals in each generation of the population. As the number of generations increases, the amount of information and

the number of interactions that the user needs to handle increase dramatically. This high frequency of interactions not only consumes the user's energy and time but also may lead to subjective bias and inconsistent evaluation standards during the evaluation process. For instance, users may carefully evaluate each design individual at the initial stage but, as time goes on, the increasing fatigue may cause users to become impatient and make hasty evaluations. This situation not only reduces the reliability and accuracy of user evaluations but also may affect the convergence speed of the algorithm and the quality of the final optimization results.

To address the above issues, a large number of researchers have used proxy model methods to predict the fitness value, thereby reducing the number of user evaluations and alleviating fatigue. Huang et al. [16] constructed KD tree proxy models and random forest proxy models to assist with evaluations based on historical user evaluation information. Lu et al. [17] constructed a user cognitive proxy model based on the BP neural network (BPNN). Gypa et al. [18] proposed an IGA integrated with a support vector machine for propeller optimization. Zhen and Nie [19] constructed the objective fitness values of the IGA based on the weight values. Sheikhi and Kaedi [20] tackled the user's fatigue problem in the interactive genetic algorithm by using the candidate elimination algorithm. As users may not be professional product designers, they may not be able to accurately evaluate the product and can only have a rough interval estimation of the product. Therefore, some researchers have thought of using individual interval fitness values to reduce the uncertainty of fitness values. Sun et al. [21] proposed an improved semi-supervised learning co-training algorithm to assist the IGA, which considers the uncertainty of interval-based fitness values when training and weighting two co-training models. Gong et al. [22] proposed an IGA based on the proxy model of individual interval fitness.

XGBoost is an efficient and flexible machine learning algorithm that is used in various fields in combination with the GA. Deng et al. [23] proposed a hybrid gene selection method based on XGBoost and a multi-objective genetic algorithm for cancer classification. Wu et al. [24] used an improved genetic algorithm and XGBoost classifier for transformer fault diagnosis. Ghatasheh et al. [25] employed a genetic algorithm to optimize XGBoost for spam prediction. Gu et al. [26] used the genetic algorithm in combination with an XGBoost model for prediction of maximum settlement in mines. Li et al. [27] utilized a genetic algorithm and the XGBoost algorithm for identification of mixed mine water inrush.

Currently, there are fewer applications of XGBoost combined with the IGA. In order to solve the core problem of the user's fatigue, this study proposes a method to improve the IGA using an XGBoost proxy model and the GIM and using particle swarm optimization to optimize the hyperparameters of the XGBoost proxy model. This method uses the collected user evaluation information to construct the XGBoost model. The proxy model predicts the fitness value for each individual, assisting users in their evaluations. If users feel that the predicted score differs significantly from the expected value, they can make modifications. The GIM helps users to adjust the shape and appearance of the product independently and enables them to find satisfactory individual solutions more quickly. The main contribution of this paper is the proposal to use the XGBoost proxy model to improve the IGA. By predicting the users' evaluation fitness value based on collected historical user data, the number of user evaluations is reduced, thereby addressing the issue of the user's fatigue. Additionally, the GIM is integrated into the interactive interface, allowing users to fine-tune the shape of the evaluated individuals, enhancing user satisfaction while avoiding the fatigue caused by repetitive operations. In this study, the algorithm is applied to a 3D vase design platform to verify its accuracy, optimization capability, and ability to mitigate the user's fatigue.

2. Algorithm and Principle

2.1. The Proposed Method

The algorithm of this study is based on the combination of the interactive genetic algorithm and the XGBoost proxy model improved by particle swarm optimization. The

proxy model is constructed by users' historical data to predict the individual's fitness value and help users to evaluate and reduce user operations. To allow users to participate more intuitively in the process of product design, a graphical interaction mechanism has been implemented in this study. The users can freely modify the characteristics of the individual product, enabling the interactive genetic algorithm to efficiently generate a customized design. Such an approach reduces the time required for design and enhances users' satisfaction.

2.2. Principle of XGBoost Algorithm

XGBoost [28], originally proposed by the team led by Tianqi Chen, is an optimized distributed gradient enhancement library designed to achieve higher efficiency, flexibility, and portability. It is an efficient and widely used gradient lifting algorithm for machine learning and data mining tasks. The basic components of XGBoost are the decision trees, which are referred to as weak learners. These weak learners collectively form the XGBoost. The core idea is to grow a tree by constantly adding trees and constantly splitting features. Each time a tree is added, it is actually learning a new function $f(x)$. There is a sequence between the decision trees that make up XGBoost; the generation of the latter decision tree will consider the prediction results of the previous decision tree; that is, the deviation of the previous decision tree will be taken into account. The data set required for each decision tree is the entire data set, so the process of generating each decision tree can be regarded as the process of generating a complete decision tree.

When predicting a new sample, it is necessary to input the new sample into each decision tree of XGBoost in turn. In the first decision tree, a predictive value will be generated. In the second decision tree, another predicted value will be generated. By analogy, the new sample is consistently put into all the decision trees. Finally, the predicted values calculated by all the decision trees are aggregated to obtain the final predictive value for the new sample. The prediction model is then constructed, and it is defined in the following way:

$$y_i^t = \sum_{k=1}^{t} f_k(x_i) = y_i^{t-1} + f_t(x_i), \tag{1}$$

In Equation (1), $f_k(x_i)$ is the predicted value of the decision tree and t is the number of decision trees. The XGBoost algorithm retains the prediction of the previous $t-1$ round during each model training and adds a new function $f_t(x_i)$ to the model, which is the prediction result of the ith sample at the tth model training.

The prediction accuracy of the model is determined by the deviation and variance of the model. The loss function represents the deviation of the model and, in order to keep the variance small, the regularization term needs to be added to the objective function to prevent over-fitting. Therefore, the objective function is composed of the loss function of the model and the regularization term that suppresses the complexity of the model. The definition of the objective function is as follows:

$$Obj(\theta) = L(\theta) + \Omega(\theta) = \sum_{i=1}^{n} L(y_i, y_i^t) + \sum_{k=1}^{t} \Omega(f_k), \tag{2}$$

In Equation (2), it can be seen that the objective function $Obj(\theta)$ is composed of two parts: $\sum_{i=1}^{n} L(y_i, y_i^t)$ is the sum of the errors generated between the true value and the predicted value of n test samples, and L is the loss function of XGBoost.

$\Omega(f_k)$ is the regularization penalty function of model complexity, which is defined as

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{k=1}^{T} w_k^2, \tag{3}$$

In Equation (3), γ represents the penalty coefficient, and λ represents a fixed coefficient. γ is used to control the number of leaf nodes in the decision tree. When the number of leaf nodes is too large it is easy to produce over-fitting. λ is used to control the weight of each decision tree, to ensure that its value is not too large, and to avoid limited space for the subsequent decision tree. T represents the number of leaf nodes of the decision tree. f_k, w_k is the vector formed by all the leaf node values of the decision tree, and this formula is the complexity of a single classifier.

The objective function is simplified by Taylor's formula and defined as

$$Obj = \sum_{i=1}^{T} \left[G_i w_i + \frac{1}{2}(H_i + \lambda) w_i^2 \right] + \gamma T, \tag{4}$$

In Equation (4), G_i is the sum of the first gradient of the ith leaf node of all decision trees and H_i is the synthesis of the second gradient of the ith leaf node of all decision trees. When each decision tree is generated, the structure is determined, and then G_i, H_i and T will also be determined.

To achieve the best performance of XGBoost, it is essential to use appropriate methods to construct the optimal structure of the decision tree. There are two ways to split the nodes of the XGBoost algorithm: the greedy algorithm and the approximation algorithm. The greedy algorithm is the main node-splitting method in XGBoost. Starting from the root node, the greedy strategy is used to select the best splitting feature as the splitting node to segment the training data. Then the greedy strategy is continuously used to split until the decision tree can no longer continue to split. The information gain of each splitting feature is calculated. The feature with the largest information gain is the best splitting feature, which is defined as

$$L_{split} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)}{H_L + H_R + \lambda} \right] - \gamma, \tag{5}$$

In Equation (5), L_{split} is the information gain, and $\frac{G_L^2}{H_L+\lambda}$, $\frac{G_R^2}{H_R+\lambda}$, $\frac{(G_L+G_R)}{H_L+H_R+\lambda}$ are the values of the left subtree, the right subtree and the undivided decision tree, respectively. When $L_{split} < 0$, the decision tree gives up segmentation.

2.3. Principles of Particle Swarm Optimization

Particle swarm optimization (PSO) [29] is an optimization algorithm based on the foraging behavior of birds in nature, and it is used to solve optimization problems. The PSO simulates the social behavior and collaborative learning processes between individuals in bird flocks to find the optimal solution.

1. Individual representation: In the particle swarm optimization algorithm, each candidate solution in the solution space is called a particle, and each particle represents the hyperparameters of XGBoost, which includes the objective function, the learning rate, the maximum depth of each tree, the sub-sampling rate of the training samples of each tree, the sub-sampling rate of the features of each tree, and the number of trees. The loss function is used to minimize the mean square error, as shown in Equation (6):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{6}$$

2. Fitness function: The problem solved by the particle swarm optimization algorithm usually needs to maximize or minimize an objective function, which is called a fitness function. The fitness function is the RMSE of the XGBoost proxy model. The smaller the value is, the better the model's effectiveness will be, as shown in Equation (7):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \quad (7)$$

3. Initialization: At the onset of the algorithm, a certain number of particles are randomly generated, and their position and velocity are initialized. In general, the positions of particles are randomly distributed in the solution space, and the velocity is initialized as a zero vector.
4. Fitness evaluation: For each particle, the fitness value is calculated corresponding to its position.
5. Individual optimal position update: For each particle, the individual optimal position should be updated, the current fitness value should be compared with the fitness value of the individual optimal position, and it should be updated if it is better.
6. Global optimal position update: The position of the particle with the best fitness value among all individuals should be selected as the global optimal position.
7. Velocity and position update: According to certain rules, update the velocity and position of the particles in order to move towards the individual optimal position and the global optimal position. The velocity update depends on the historical velocity of the particle, the individual optimal position, and the global optimal position. The velocity update equation is shown in Equation (8):

$$V_{iL}^{k+1} = wV_{iL}^{k} + c_1 r_1\left(P_{bL}^{k} - X_{iL}^{k}\right) + c_2 r_2\left(P_{gL}^{k} - X_{iL}^{k}\right), \quad (8)$$

In Equation (8), where $V_{iL}^{k} = [v_{i1}^{k}, v_{i2}^{k}, \ldots, v_{iL}^{k}]$ is the velocity of the ith particle at time k, $P_{bL}^{k} = [P_{i1}^{k}, P_{i2}^{k}, \ldots, P_{iL}^{k}]$ is the historical individual optimal position of the ith particle, and $P_{gL}^{k} = [P_{g1}^{k}, P_{g2}^{k}, \ldots, P_{gL}^{k}]$ is the global optimal position. The inertia weight w represents the influence of the velocity of the previous generation of particles on the velocity of the current generation of particles. A larger inertia weight contributes to global optimization, while a smaller inertia weight contributes to local optimization. k is the k th generation of the population, c_1 and c_2 are the individual velocity factor and the global velocity factor, respectively, and r_1 and r_2 are random numbers between 0 and 1.

$$X_{iL}^{k+1} = X_{iL}^{k} + V_{iL}^{k+1}, \quad (9)$$

In Equation (9), $X_{iL}^{k} = \left[x_{i1}^{k}, x_{i2}^{k}, \ldots, x_{iL}^{k}\right]$ is the position of the ith particle at time k.

8. The termination condition: According to the set termination condition (such as the number of iterations to reach the preset value or the fitness to reach the threshold), determine whether to end the algorithm. If the termination condition is not met, go back to step 4.
9. Output results: Output the solution corresponding to the global optimal position as the optimal hyperparameters of XGBoost.

2.4. Proxy Model Flowchart and Pseudocode

When using PSO to optimize the hyperparameters of XGBoost, the hyperparameters of XGBoost are typically treated as the optimization variables in PSO. The specific steps are as follows: First, the dataset for training and testing is collected, including features and labels. Next, a fitness function is defined, which takes the hyperparameters of XGBoost as the input and returns the RMSE of the model on the training data. Then, the parameters for the PSO algorithm are set, as detailed in Table 1. Subsequently, the PSO algorithm is used to search for the optimal hyperparameter combination of XGBoost to minimize the fitness function. Once the optimal hyperparameters are found, these parameters are used to train the XGBoost model. Finally, the performance of the trained XGBoost model is evaluated

on the test set, and the trained model is saved to a file for future use. The PSO-XGBoost algorithm flow is shown in Figure 1, and the pseudo-code is shown in Algorithm 1.

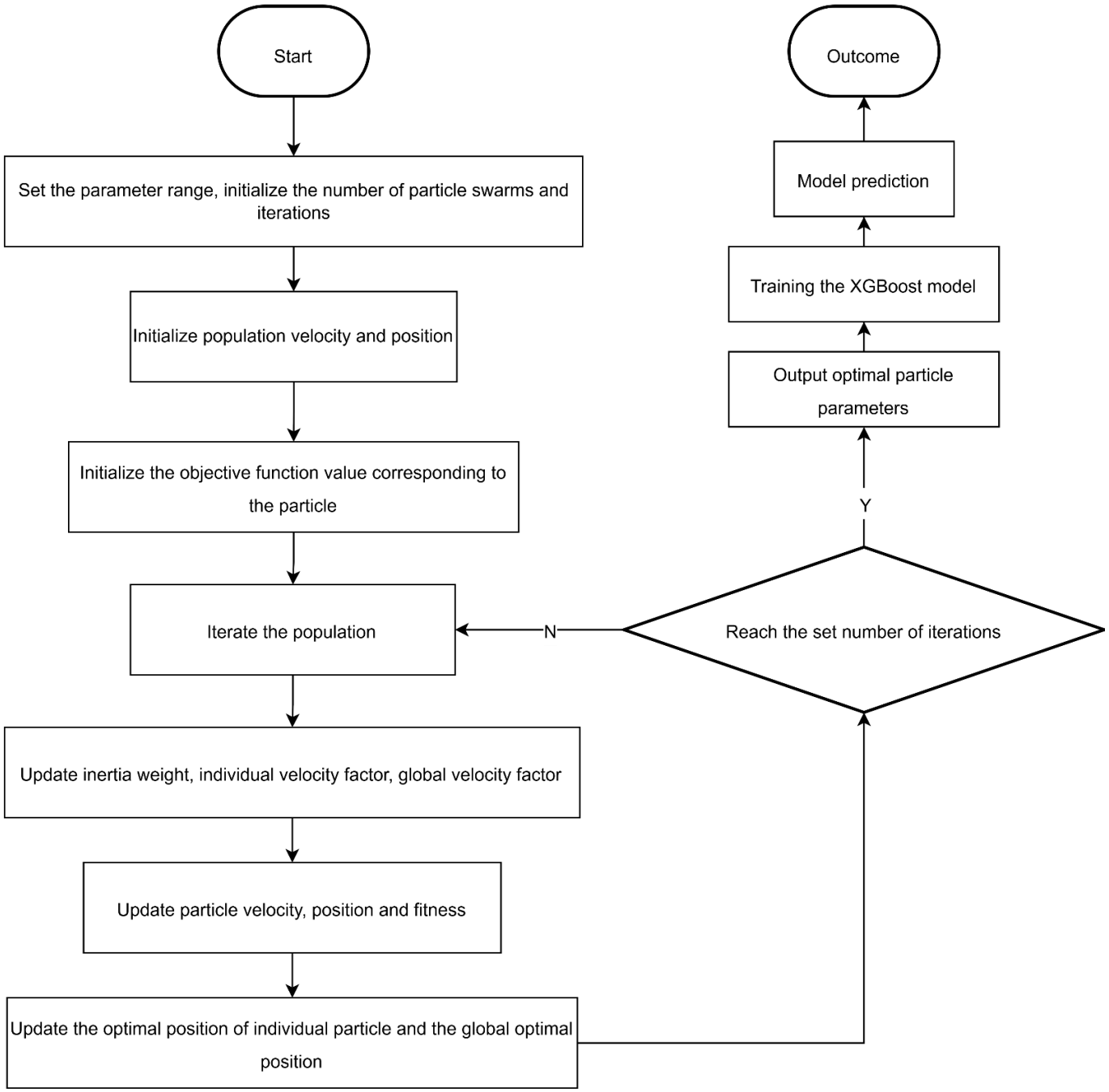

Figure 1. The implementation process of evaluation of the proxy model.

Table 1. PSO parameters setting.

Parameter	Numerical Value
number of particles	10
number of dimensions	5
number of iterations	50
inertia weight	0.5
c1, c2	1.5
r1, r2	random generation (0–1)

Algorithm 1. The pseudo-code of the improved XGBoost proxy model

```
Input: populationsize, max_generations
Output: best_hyperparameters, predictions
Initialize a population of random hyperparameter sets
best_hyperparameters = None
best_rmse = infinity
For generation in range(max_generations) do
    For particle in population do
        Evaluate the fitness of the particle's hyperparameters using RMSE
            If fitness is better than the particle's personal best Then
                Update the particle's personal best hyperparameters
            If fitness is better than the global best Then
                                                Update the global best hyperparameters
                                    end
                                end
                            end
                        For particle in population do
                    Update particle velocity and position using the PSO formula
                end
end
learning_rate,max_depth,subsample,colsample_bytree,n_estimators=best_hyperparameters
max_depth = int(max(1, max_depth))
best_xgb_model = xgb.XGBRegressor(objective='reg:SSE',
                                  learning_rate=learning_rate,
                                  max_depth=max_depth,
                                    subsample=subsample,
                                  colsample_bytree=colsample_bytree,
                                  n_estimators=int(n_estimators),
                                  random_state=42)
best_xgb_model.fit(x_train, y_train)
predictions = best_xgb_model.predict(x_test)
Return: best_hyperparameters, predictions
```

2.5. Data Collection and Update

The evaluation of the proxy model helps users to assess products. The proxy model, also known as an approximation model, is a model constructed to substitute for users in evaluation. Generally, a large amount of evaluation sample information makes the evaluation of the proxy model more accurate. The number of evaluations per user is limited, and too many ratings will cause users' fatigue. In order to solve this problem, this paper collects all the evaluation information data of users who have used the 3D vase design system [30], including the users' personal information, the individual characteristics data, and the fitness value information of the vase. These data are used to construct a PSO-XGBoost model, which can allow the current user to find the other similar users quickly and obtain their historical evaluation data to predict the fitness of the new individual.

However, it should be noted that the evaluation of the proxy model based on similar individual information and the adaptive value predicted by the proxy model may be different from the actual evaluation of the user. Therefore, users are allowed to modify and submit the individual fitness predicted by the proxy model, and the user's evaluation data are saved. Considering that updating the proxy model is a time-consuming process, in order to avoid affecting the user's design experience, a system has been designed to automatically update the proxy model when the population evolution is terminated by the user. As shown in Figure 2, this ensures that the model is updated in a timely manner.

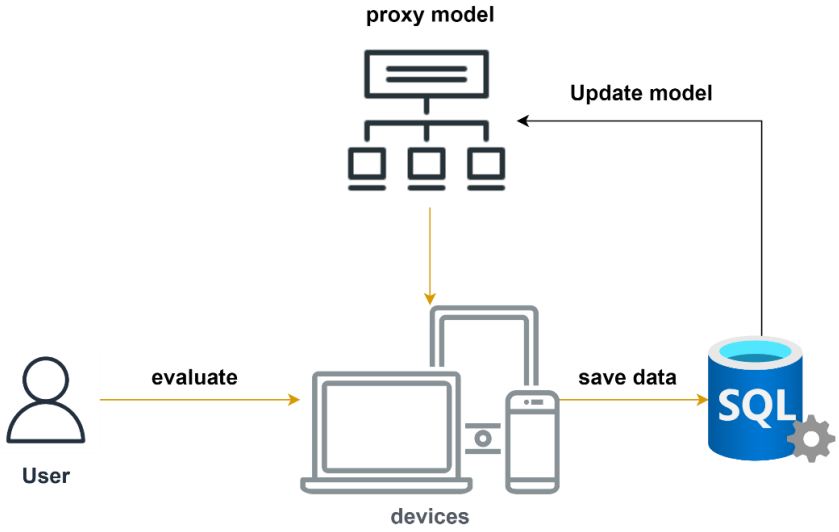

Figure 2. Update the proxy model.

3. Vase Design

3.1. Vase Construction and Coding

In order to meet the various needs of different users and generate different types of vases, the vases are designed based on bicubic Bézier surfaces [31]. First, this method requires constructing a mesh model of the vase using the control points of the Bézier surface and then changing the coordinates of the control points to change the shape of the vase, thereby generating different vases. The definition of a bicubic Bézier surface is as follows:

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} P_{i,j} B_{i,3}(u) B_{j,3}(v), (u,v) \in [0,1] \times [0,1]. \tag{10}$$

In Equation (10), $B_{i,3}(u)$ and $B_{j,3}(v)$ are cubic Bernstein basis functions, and $P_{i,j}$ is the control point of the surface. The equation can be changed into

$$p(u,v) = UMPM^T V^T. \tag{11}$$

In Equation (11), $U = \begin{bmatrix} u^3 u^2 u 1 \end{bmatrix}$, $V = \begin{bmatrix} v^3 v^2 v 1 \end{bmatrix}$, $M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$,

$P = \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} & P_{0,3} \\ P_{1,0} & P_{1,1} & P_{1,2} & P_{1,3} \\ P_{2,0} & P_{2,1} & P_{2,2} & P_{2,3} \\ P_{3,0} & P_{3,1} & P_{3,2} & P_{3,3} \end{bmatrix}$.

The construction of the vase model is relatively simple and mainly includes three parts: the bottle mouth, the bottle body, and the bottle bottom. Due to the central rotational symmetry of the vases, the vase model can be considered as formed by the rotation of the Bézier curve. To enhance the intricacy of the vase curve, this study utilized two cubic Bézier curves for the vase contour [32], as depicted in Figure 3. The control points P_0–P_6 form a cubic spline curve, with P_3 serving as the connection point for the two curves. To ensure a smooth connection between the two curves, it is essential to maintain the collinearity of points P_2, P_3, and P_4. On the right side of the image is the control point grid model for the double cubic Bézier surfaces. To maintain the central rotational symmetry of the vase, the x and y coordinates of the initial control points in the same row are multiplied by the same scaling factor. This ensures a uniform reduction or enlargement of the control points

in the same row, preventing deformation of the vase. The control points aligned with P_0, such as P_{00}–P_{03}, as shown in Figure 3, undergo scaling by the scaling factor applied to the P_0–P_6 control points. Therefore, this study adjusts the scaling factor to P_0–P_6, allowing the control points of the vase to be modified, thereby simplifying the users' operation in shaping the vase.

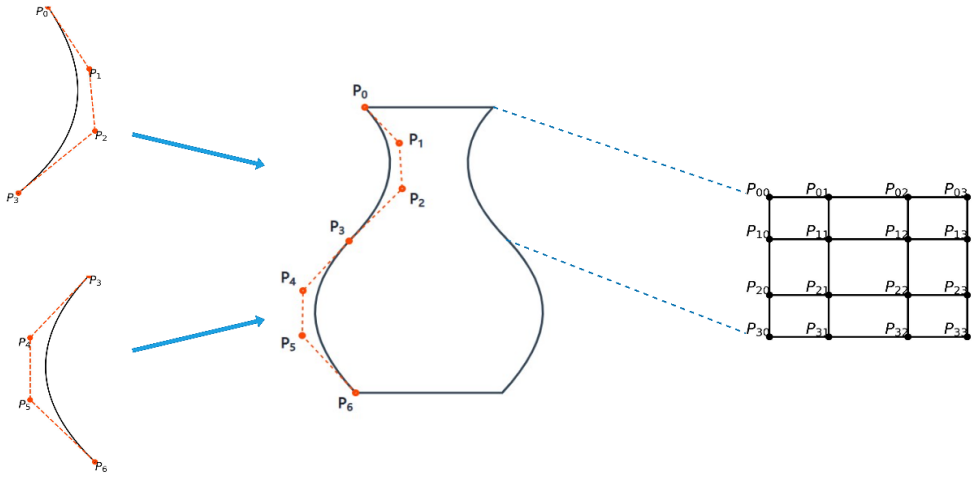

Figure 3. Vase contour.

Before constructing the vase, it is necessary to generate the vase mesh model using control points. When creating the mesh model, a quadrilateral mesh is used; however, in the rendering and computation processes, triangular meshes are generally more stable, especially during deformation or transformations. Therefore, when using Three.js to add materials and render the vase, the mesh model is converted into a form composed of triangles. The vase mesh model is illustrated in Figure 4. As two cubic Bézier curves were employed to form the vase contour, it becomes essential to divide the vase body into upper and lower sections. The upper section comprises four bicubic Bézier surfaces, and the lower section is formed by four surfaces. The vase mouth is constructed by a circle formed by a curve, and the bottom is composed of a circle formed by the concatenation of four Bézier surfaces. When joining the upper and lower surfaces, the last row of the upper surface serves as the first row of the lower surface. Similarly, when joining the left and right surfaces, the last column of the left surface serves as the first column of the right surface. To ensure the smoothness of surface concatenation, the symmetric control points at the junction are first uniformly adjusted and then the normal vectors of the surfaces are calculated. This ensures the correct computation of normal vectors at each surface point. Correct normal vectors are crucial for rendering smooth and realistic surfaces, especially for handling lighting and shading. When light strikes the surface, normal vectors are used to calculate the angle between the light and the surface, influencing the scattering and reflection of light. This is essential to achieve visual smoothness and a realistic texture, particularly in the context of lighting and shading processing. In order to elevate the authenticity of the designed vase, a richer pattern has been incorporated onto its surface. Employing advanced image texture mapping technology [33], this intricate pattern is seamlessly fused with the vase, creating a harmonious and lifelike aesthetic.

The IGA's gene coding is made up of three parts: texture picture, vase height, and control point parameters. The random combination of the three parts constitutes different chromosomes representing different vase models. The composition of the vase gene coding is shown in Figure 5. The curve of the bottle body is composed of anchor points (P_0, P_3, P_6) and curvature control points (P_1, P_2, P_4, P_5). The mouth and bottom of the vase are controlled by P_0 and P_6, respectively.

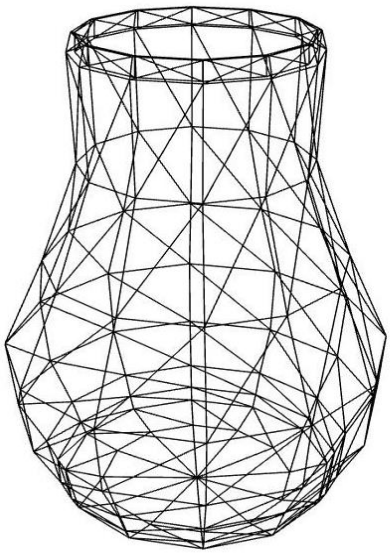

Figure 4. Vase mesh model.

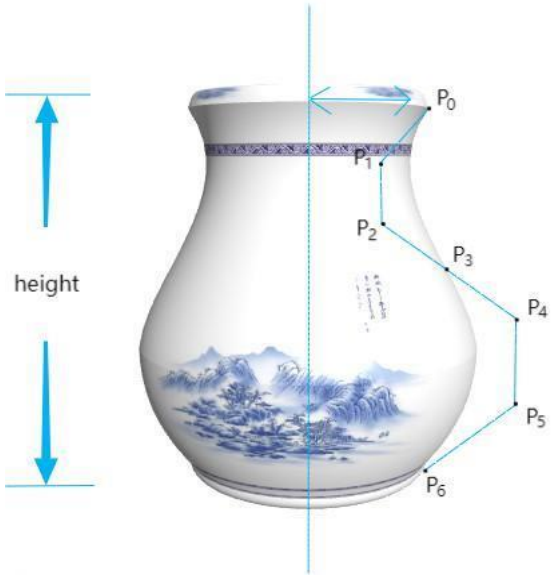

Figure 5. The coding composition of the vase.

The individual utilizes binary encoding, consisting of control point parameters, vase height, and texture images, as illustrated in Figure 6. The x and y coordinates of control points P_0 to P_6 are represented by seven sets of 8-bit binary codes, ranging from 0 to 2.55 times their original values. The z-axis coordinates of control points P_0 to P_6 are represented by 8-bit binary codes, also ranging from 0 to 2.55 times their original values. There are a total of 64 texture images, each represented by a 6-bit binary code. Therefore, the genetic encoding for a vase comprises 70 bits. Once the population evolution has been completed, the decoding of the corresponding binary codes allows the retrieval of individual vase characteristics, which can then be displayed on the interactive platform.

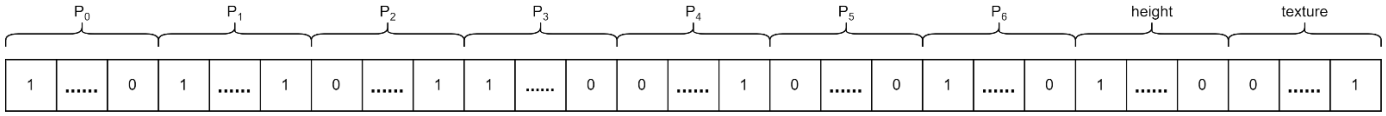

Figure 6. Gene codes.

3.2. Evolutionary Operators

The evolutionary operators use roulette selection and elite strategy, multi-point crossover, and uniform mutation. Figure 7 shows the operation diagram of the crossover operator and the mutation operator. When the chromosomes intersect, multiple crossover points are randomly set in the individual chromosomes, and then gene exchange will be performed. When the chromosome mutates, the mutation probability of each gene point is the same, and the gene point mutates when the probability is satisfied.

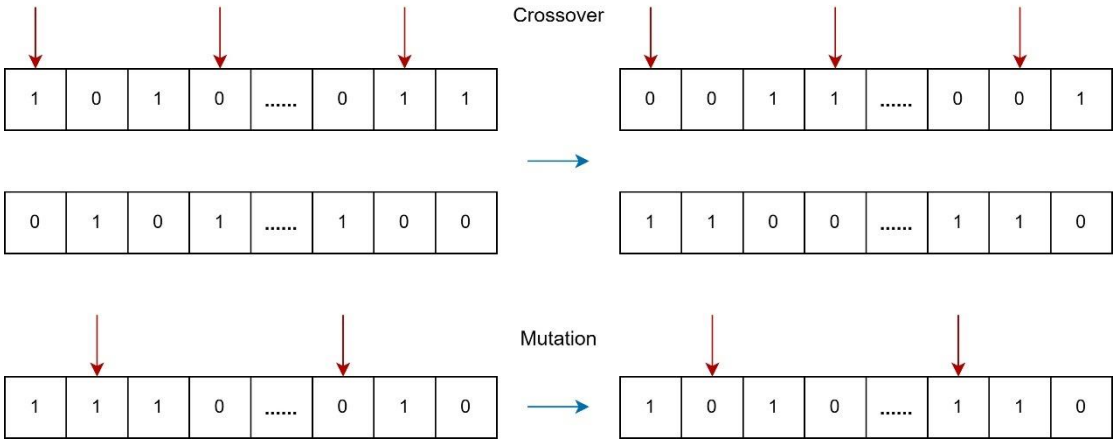

Figure 7. Crossover and mutation.

3.3. Graphic Interaction Mechanism

In this paper, the GIM is introduced in the middle and late stages of individual evolution. With the GIM, users can modify the characteristics of individual vases according to their personal preferences so that they can quickly find the products they are satisfied with. Considering that not all users have a good understanding of vase design, computer graphics are integrated [34] and a parametric method is used to construct a 3D model. As shown in Figure 8, the picture of the vase on the left is a 3D vase generated by the bicubic Bézier surfaces, and several buttons on the right are used to control the parameters of the vase. With the GIM, the user can change the vase parameters by dragging the button so that the shape of the vase will be changed. When the users are not satisfied with the shape of the vase, they can use the GIM to change the shape and add their own preferred individuals to the population. This enriches the population diversity, so that the evolution does not easily fall into the local optimum.

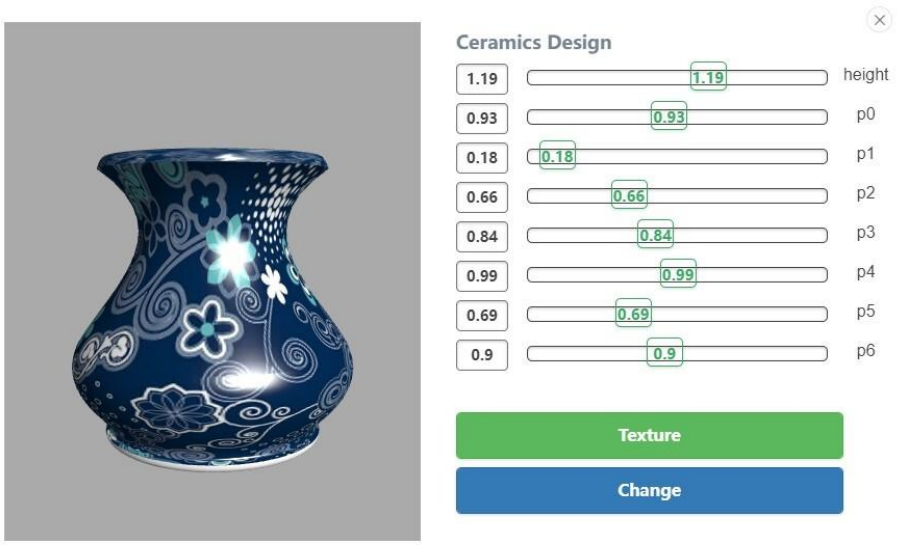

Figure 8. Graphic interaction mechanism.

3.4. Algorithm Procedure

In response to the IGA's core problem of the user's fatigue, this study has made improvements to the IGA by adding a proxy model to assist users to evaluate individuals. If users feel unsatisfied with the fitness values predicted by the proxy model, they have the option to adjust them to better align with their expectations. However, some users may have limited understanding of the designed products, resulting to uncertainty in the initial evaluation. In this study, the GIM has been introduced in the middle and later stages of evolution, allowing users to freely adjust the vase model according to their preferences. Once the evolution generation reaches the set generation, the evolution concludes and the user's evaluation data are stored to the database to update the proxy model, thereby enhancing its performance. The algorithm flow chart is shown in Figure 9. Algorithm 2 shows the pseudo-code of the algorithm.

Algorithm 2. Algorithm details steps

Input: *population_size,crossover_rate,mutation_rate,interactive_generation,termination_generation*
Output: *none*

```
(1) generation = 0
(2) population[generation] = initialize_population(population_size)
(3) model = PSO-XGBoost.build(DB.dataset())
(4) While generation<=termination_generation do
(5)         model.prediction(population[generation])
(6)         If generation > interactive_generation && is_adjust() Then
(7)             adjust(population[generation])
(8)         end
(9)         evaluate(population[generation])
(10)        chromosomes = encode(population[generation])
(11)        While (population[generation+1].size() < population_size) do
(12)            pairs = select(chromosomes)
(13)            If rand() < crossover_rate Then
(14)                pairs = crossover(pairs)
(15)            end
(16)            If rand() < mutation_rate Then
(17)                pairs = mutate(pairs)
(18)            end
(19)            population[generation+1].add(decode(pairs.first))
(20)            If (population[generation+1].size() < population_size) Then
(21)                population[generation+1].add(decode(pairs.second))
(22)            end
(23)        end
(24)        generation++
(25) end
(26) For g = 0 to termination_generation do
(27)        DB.save(population[g])
(28) end
(29) model.update(DB.dataset())
```

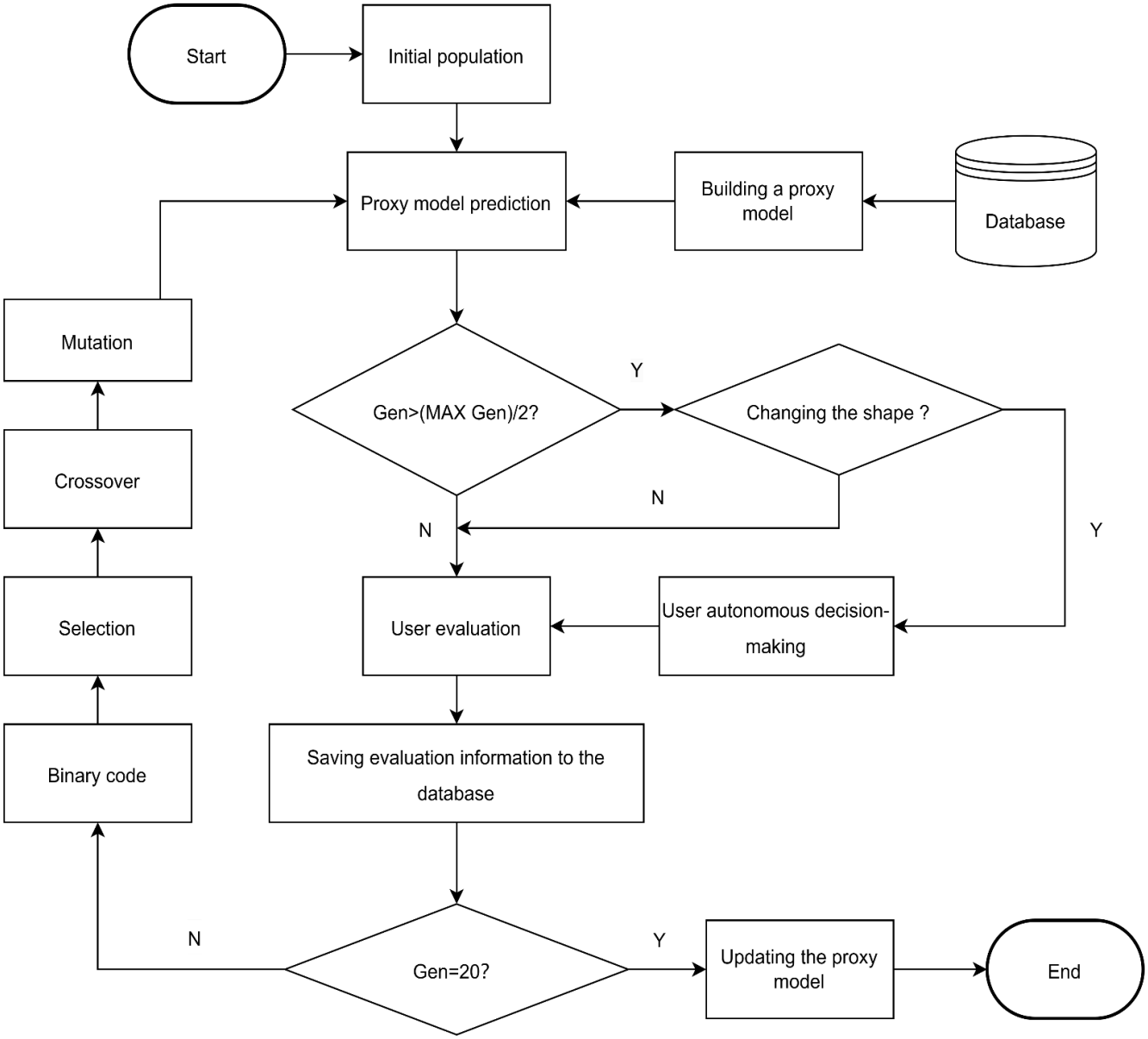

Figure 9. Algorithm flow chart.

4. Experimental Results and Analysis

4.1. Parameters Setting of Interactive Genetic Algorithm

The main problem of human–computer interaction is the user's fatigue. Appropriate population size and evolutionary termination generation can not only help users reduce fatigue, but also improve efficiency. Therefore, the population size of the system is set to six, and the generation of evolutions is set to 20. The system will judge when the number of iterations reaches the set number and the evolution will be automatically terminated. The crossover probability is set to 0.9 and the mutation probability is set to 0.1. The genetic parameters of the IGA are shown in Table 2.

Table 2. Genetic parameters setting.

Parameter	Numerical Value
maximum generation	20
population size	6
crossover probability	0.9
mutation probability	0.1

4.2. Comparison of Proxy Models

In order to verify the prediction performance of the PSO-XGBoost proxy model, this study compares the accuracy of the XGBoost model and the PSO-XGBoost model. Due to the current single database in the vase design platform, where all user evaluation data and individual evaluation details are stored, the platform's database is utilized as the dataset for training the surrogate model. The dataset is divided into training and testing sets in a 9:1 ratio. The training set is used to train the surrogate model, while the testing set is employed to validate the effectiveness of the predicted results. Because there is too much data in the test set, 100 samples in the test set are randomly selected for comparison, and the predicted value of the model is compared with the real value, as shown in Figure 10.

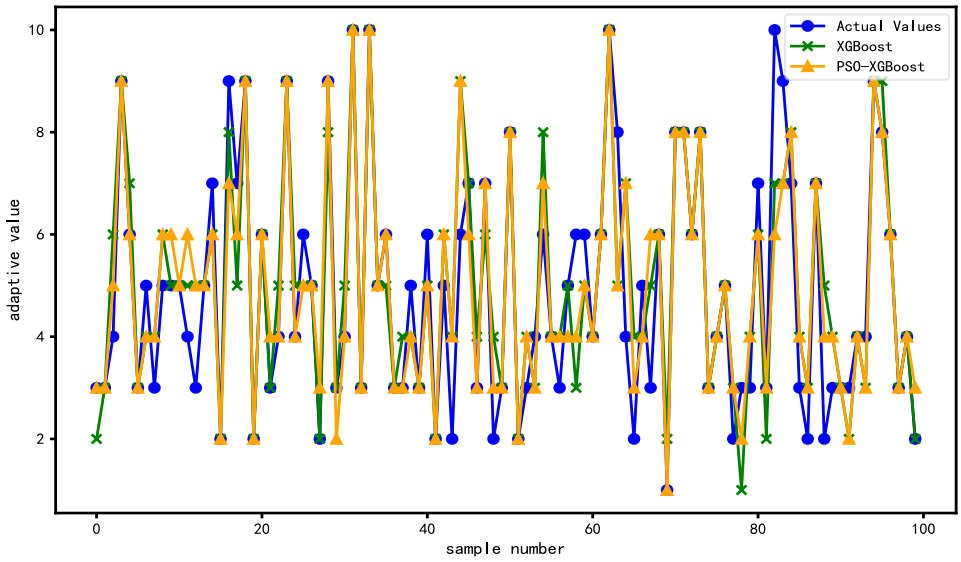

Figure 10. The predicted values of the XGBoost and PSO-XGBoost models are compared with the real values.

It can be seen from Figure 10 that the PSO-XGBoost model curve better matches the real value curve than the XGBoost model curve. The performance indicators of the two models are shown in Table 3.

Table 3. Performance comparison of proxy models.

Proxy Model	RMSE	MAE	Accuracy	R^2
XGBoost (1)	1.0858	0.695	87.5%	0.793
XGBoost (2)	1.0668	0.656	87.6%	0.798
XGBoost (3)	1.0677	0.642	87.7%	0.797
PSO-XGBoost	1.0286	0.618	88.8%	0.814

It can be seen from Table 3 that this study compares the RMSE, the MAE, the accuracy, and R^2 of the two proxy models under the same data set. XGBoost (1), XGBoost (2), and XGBoost (3) are models trained under different hyperparameters of XGBoost. The specific parameters are shown in Table 4. Compared with the XGBoost model, the RMSE of the

PSO-XGBoost model has decreased by 0.0391–0.0572, the MAE has decreased by 0.024–0.077, the accuracy has increased by 1.1–1.3%, and the R^2 has increased by 0.016–0.021. It is concluded that the PSO-XGBoost model has better performance and better prediction effect.

Table 4. XGBoost with different parameters.

Parameter	XGBoost (1)	XGBoost (2)	XGBoost (3)
objective	reg:squarederror	reg:squarederror	reg:squarederror
learning_rate	0.1	0.12	0.13
max_depth	5	6	7
subsample	0.7	0.7	0.7
colsample_bytree	0.7	0.7	0.7
n_estimators	1000	1100	1300

In order to further prove the effectiveness of the PSO-XGBoost model, this study compares the K-D tree (KDT) proxy model, the random forest (RF) proxy model, and the PSO-XGBoost proxy model. Randomly select 100 samples in the test set for comparison and use different proxy models to predict user evaluation, as shown in Figure 11.

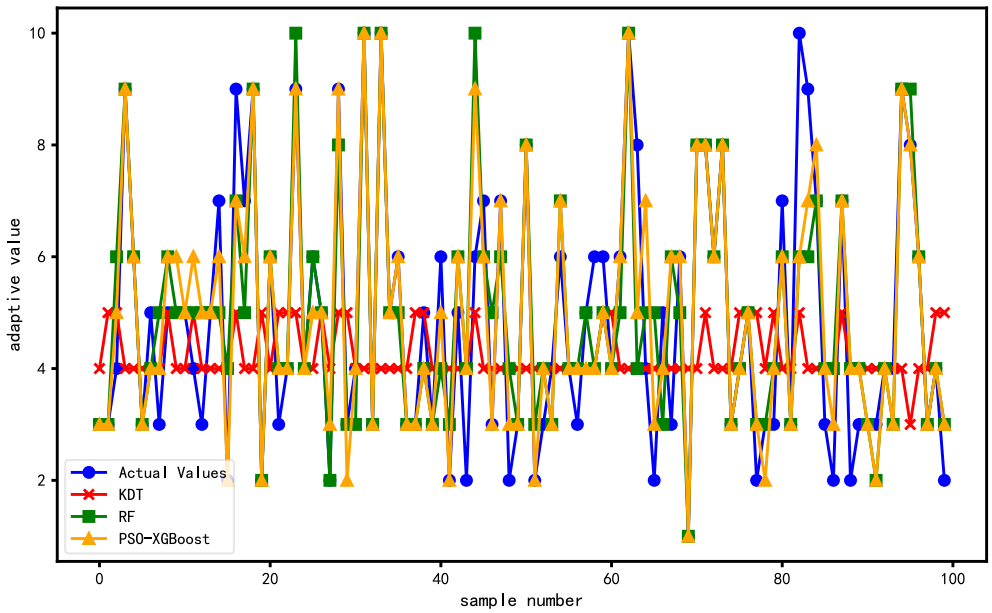

Figure 11. The predicted values of the KDT, RF, and PSO-XGBoost models were compared with the true values.

It can be seen from Figure 11 that the predicted value curve of the PSO-XGBoost model is consistent with the real value curve, and the predicted value curves of the KDT model and the RF model have certain gaps from the real value curve. The PSO-XGBoost proxy model is compared with the KDT model and the RF model performance indicators, as shown in Table 5.

Table 5. Performance comparison of proxy models.

Proxy Model	RMSE	MAE	Accuracy	R^2
KDT	2.5677	1.953	68.6%	0.537
BPNN	1.2400	0.812	84.1%	0.738
RF	1.1533	0.750	84.9%	0.766
BCMO-XGBoost	1.0247	0.626	89.1%	0.815
PSO-XGBoost	1.0286	0.618	88.8%	0.814
PSO-XGBoost(average)	1.1015	0.723	86.8%	0.7854

It can be seen from Table 5 that this study compares the RMSE, the MAE, the accuracy, and R^2 of different proxy models under the same data set. Compared with the KDT and the RF proxy models, the RMSE of the PSO-XGBoost model decreased by 1.5391 and 0.1247, respectively, the MAE decreased by 1.335 and 0.132, respectively, the accuracy increased by 20.2 % and 3.9 %, respectively, and the R^2 increased by 0.277 and 0.048, respectively. It is concluded that the prediction effect of the PSO-XGBoost model is better than that of KDT model and RF model. In addition to algorithms in the field of machine learning, this paper also compared the BPNN algorithm [18] in the field of deep learning. From the performance indicators, it can be seen that the prediction performance of PSO-XGBoost is slightly better. To further demonstrate the performance of PSO-XGBoost, this study also took into account the randomness in the PSO optimization parameters. Therefore, PSO-XGBoost was run 10 times to obtain the average performance. From Table 5, it can be observed that compared to the optimal PSO-XGBoost model, the performance of PSO-XGBoost (average) is slightly inferior, but it is better than that of the previous three algorithms. In addition, this paper also utilized Balancing Composite Motion Optimization (BCMO) [35] to optimize the hyperparameters of XGBoost, referred to as BCMO-XGBoost. As shown in Table 5, BCMO-XGBoost exhibits slightly better performance in terms of RMSE and accuracy compared to PSO-XGBoost. However, BCMO-XGBoost shows slightly worse performance in terms of MAE and R^2 compared to PSO-XGBoost. The optimization stability of BCMO-XGBoost is expected to be better, albeit with longer runtime. In general, PSO-XGBoost performs well.

4.3. Comparison of Optimization Ability

In order to prove the effectiveness of the methods proposed in this study, the traditional IGA, IGA using KDT proxy model (KDTGIM-IGA) [17], IGA using RF proxy model (RFGIM-IGA), IGA using XGBoost proxy model (XGBGIM-IGA), and IGA using PSO-XGBoost proxy model (PXG-IGA) were compared. Using the method of controlling variables, the evolutionary parameters of the five methods are set to the same value. The evolutionary operator uses roulette selection and elite strategy, multi-point crossover, and uniform mutation. In this study, five users were selected to operate each algorithm once. The users operated through the interactive interface, and the scoring interval was 0–10 points. When the population evolves to the end of the 20th generation, the interactive interface is shown in Figure 12. This study mainly compares the average fitness, average maximum fitness, number of user evaluations, and user evaluation time of five different algorithms to verify the ability of the method proposed in this study to optimize performance, reduce the number of user evaluations, and alleviate the user's fatigue.

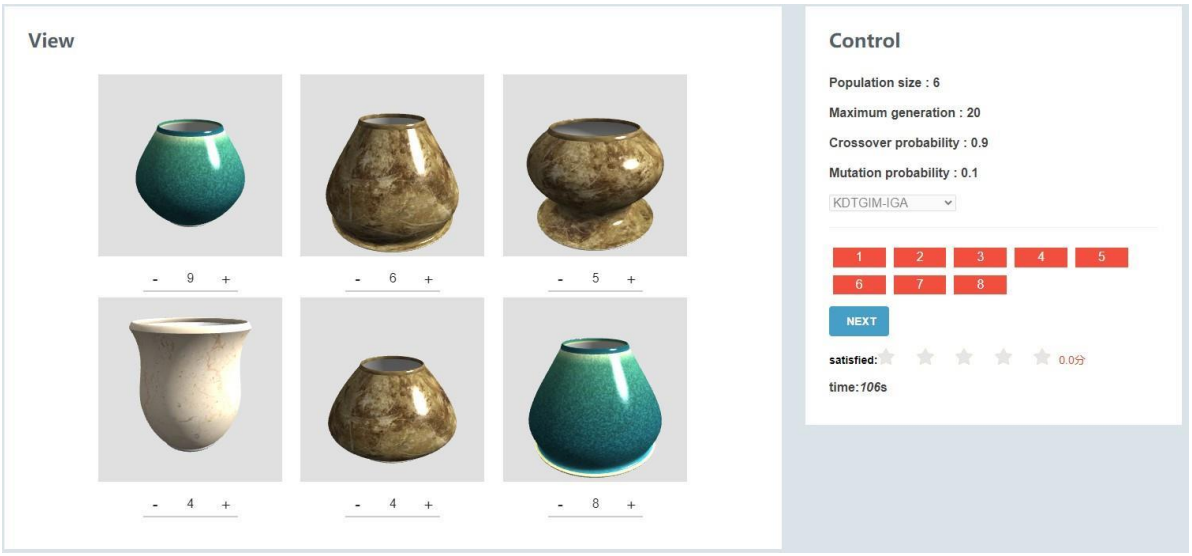

Figure 12. Interactive interface.

According to the experimental results, Figures 13 and 14 compare the fitness distribution of the five users using the five methods. Figure 13 shows the trend graph of the average fitness value of each generation of evolutionary individuals. These five curves represent the change trend of the average fitness value of each generation of the five methods. The curve of the PXG-IGA proposed in this study shows an upward trend with the evolution of generations, and the curve of the PXG-IGA is higher than that of the other algorithms after the tenth generation, indicating that the PXG-IGA is better than the other algorithms. In the first 10 generations, there was no significant difference between the traditional IGA and other IGA curves with the proxy models. However, after 10 generations, due to the use of the graphical interaction mechanism, the curves of the other four algorithms are significantly improved compared with the IGA curves. From Figure 14, the change trend of the average maximum fitness value of the user evaluation individual with the increase in the generations can be observed. Obviously, the average maximum fitness curve of the PXG-IGA is higher than that of the other methods.

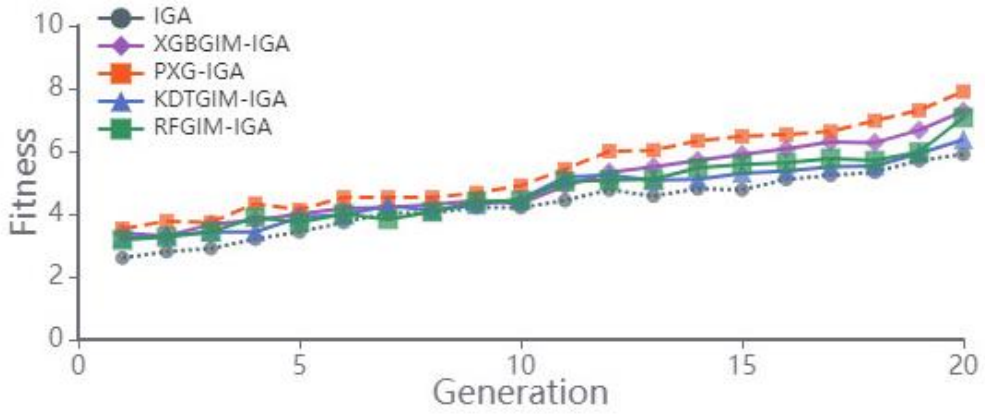

Figure 13. The average fitness comparison of users' evaluation individuals.

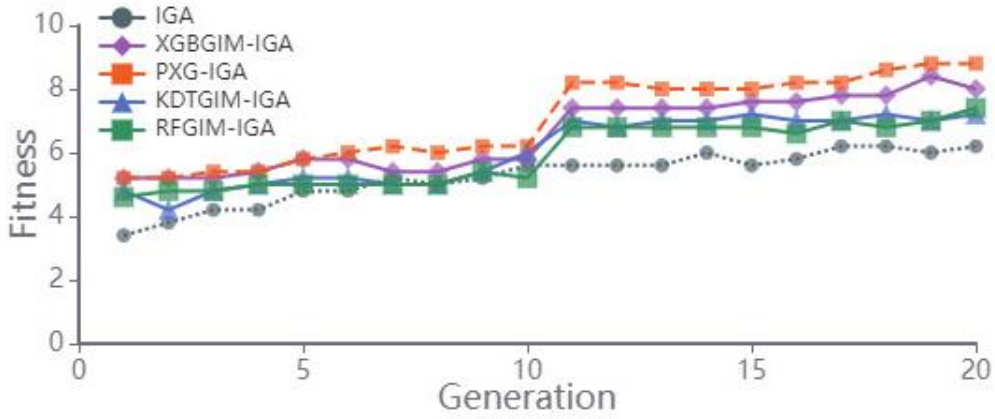

Figure 14. Comparison of average maximum fitness values of users' evaluation individuals.

4.4. Comparison of User Fatigue Alleviation

The greatest feature of the IGA is using human–computer interaction to solve those implicit performance index problems. However, humans are prone to fatigue and if users are required to operate too frequently it will increase their sense of fatigue. The user's fatigue will lead to noise in user evaluation and deviation in fitness. This study introduces the PSO-XGBoost proxy model to help users to evaluate, helping users to reduce the number of evaluations, thereby reducing fatigue. In order to prove the effect of the algorithm proposed in this study, the experiment compares the number of evaluations and time of evaluations of users using different algorithms. The number of evaluations refers to the number of individuals evaluated by the user. The more users evaluate the

number of individuals, the more likely they are to be fatigued. The time of evaluations can reflect the difficulty of the user's evaluation of the population and the performance of the system. The longer the evaluation time, the longer the user participates in the design, and the more likely the user is to be fatigued. Therefore, the number of evaluations and time of evaluations can be used to measure the indicators of the user's fatigue. The number of evaluations and time of evaluations of the five algorithms are shown in Table 6.

Table 6. Comparison of number of evaluations and time of evaluations.

User Number		**1**	**2**	**3**	**4**	**5**
IGA	number of evaluations	120	120	120	120	120
	time of evaluations	511 s	394 s	478 s	354 s	468 s
KDTGIM-IGA	number of evaluations	73	66	57	54	42
	time of evaluations	369 s	407 s	370 s	322 s	341 s
RFGIM-IGA	number of evaluations	68	65	83	31	36
	time of evaluations	340 s	341 s	314 s	286 s	292 s
XGBGIM-IGA	number of evaluations	58	60	30	42	75
	time of evaluations	358 s	236 s	270 s	242 s	248 s
PXG-IGA	number of evaluations	45	47	65	36	57
	time of evaluations	247 s	227 s	272 s	191 s	234 s

Figures 15 and 16 illustrate the number of evaluations and times of evaluations, as well as their means and standard deviations, for five users on the vase design platform using IGA, KDTGIM-IGA, RFGIM-IGA, XGBGIM-IGA, and PXG-IGA, respectively. From the comparison of mean values, the number of evaluations of the PXG-IGA is 3–70 evaluations fewer than that of the other algorithms, and the times of evaluations are 36.6–206.8 s shorter than those of the other algorithms. From the perspective of standard deviation comparison, because the number of evaluations of the IGA is 120 times each time, its standard deviation is 0, and the standard deviation in the number of evaluations in the PXG-IGA is the smallest compared with the other three algorithms. The standard deviation in the time of evaluations of the PXG-IGA is lower than that of IGA, KDTGIM-IGA, XGBGIM-IGA, and higher than that of RFGIM-IGA. This is because the evaluation behavior of different users may be very different, resulting in large fluctuations in the times of the evaluations of different users, so the standard deviation in the number of evaluations of the PXG-IGA is the smallest. On the contrary, the standard deviation in the time of evaluations will be larger than that of the RFGIM-IGA. By comparing the number of evaluations and time of evaluations of these five algorithms, it can be seen that the performance of the IGA with the proxy model is better than the traditional IGA. The average number of evaluations and time of evaluations of the PXG-IGA are lower than the other proxy model algorithms, and the stability of the algorithm is more stable than the other algorithms. This shows that the effectiveness of the PXG-IGA is better and can effectively mitigate the user's fatigue.

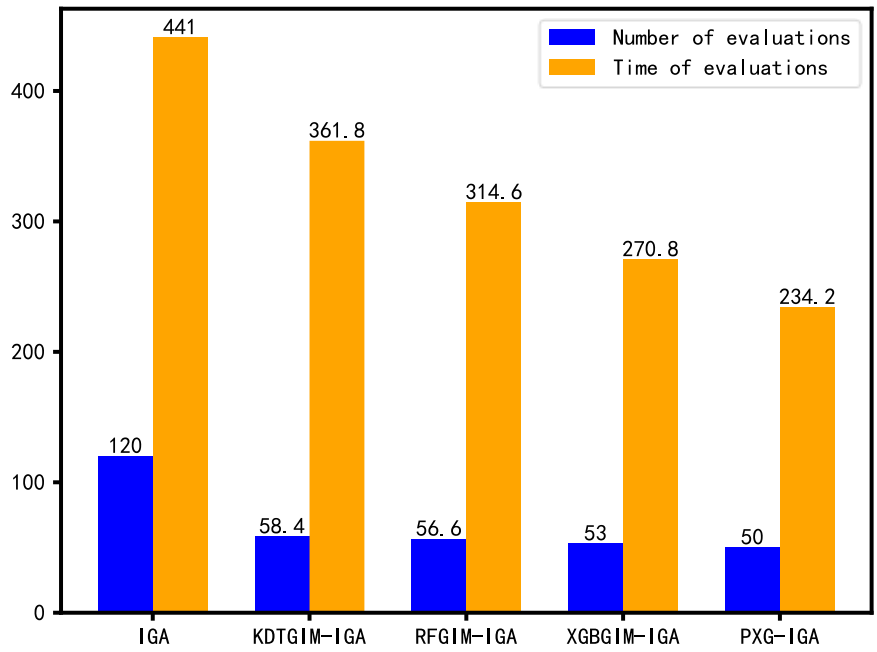

Figure 15. Comparison of the mean value of number of evaluations and time of evaluations.

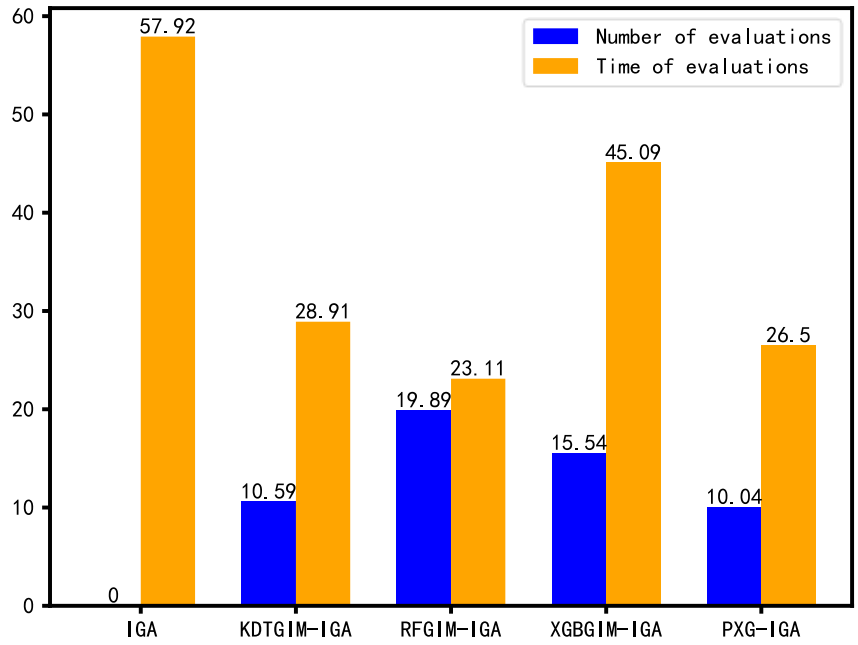

Figure 16. Comparison of standard deviation of number of evaluations and time of evaluations.

5. Conclusions

This system aims to promote the development of product design in a more intelligent and more convenient direction. A method for improving the IGA has been proposed which utilizes PSO-XGBoost as the proxy model and introduces the GIM. Based on this method, a 3D vase design system has been constructed. Based on the user's historical data, this method assists users to evaluate individual products by training the PSO-XGBoost proxy model, and constantly adds data of new users to update the model for improving accuracy. In order to design the preferred products faster, this study uses the GIM, which allows users to dynamically change individual features, introduce new features to the population, and increase its diversity.

The experimental results indicate that the PSO-XGBoost model has significant advantages in prediction performance regarding proxy model comparison. As for optimization ability comparison, the PXG-IGA is obviously superior to the IGA, KDTGIM-IGA, XGBGIM-IGA, and RFGIM-IGA in terms of average fitness and average maximum fitness, especially after the tenth generation, when its effect stands out prominently. As for comparison of the alleviation of user fatigue, the PXG-IGA shows a significant reduction in the number of evaluations and evaluation time compared to the IGA, KDTGIM-IGA, XGBGIM-IGA, and RFGIM-IGA, providing users with a better evaluation experience. Finally, it can be concluded that the method proposed in this study can effectively mitigate the user's fatigue and enable the faster design of products that satisfy users. However, in terms of alleviating the user's fatigue, different users may have varying habits and preferences. This study does not examine the impact of individual differences on the effectiveness of fatigue relief, which is an aspect that requires further, in-depth research.

Author Contributions: Conceptualization, D.W. and X.X.; methodology, D.W. and X.X.; software, D.W.; validation, D.W. and X.X.; formal analysis, D.W. and X.X.; investigation, D.W.; resources, D.W.; data curation, D.W.; writing—original draft preparation, D.W.; writing—review and editing, D.W. and X.X.; visualization, D.W.; supervision, X.X.; project administration, D.W.; funding acquisition, X.X. All authors have read and agreed to the published version of the manuscript.

Funding: This study was partially supported by the Natural Science Foundation of Fujian Province of China (no. 2021J011007), the Fujian Provincial Department of Education Undergraduate Education and Teaching Research Project (No. FBJY20230083), the Principal's Foundation of Minnan Normal University (KJ19015), the Program for the Introduction of High-Level Talent of Zhangzhou, and the National Natural Science Foundation of China (no. 61702239).

Data Availability Statement: The data will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Nguyen, D.; Le, T.; Nguyen, P.; Le, N.T.K.; Nguyen-Xuan, H. Advancing Wound Filling Extraction on 3D Faces: An Auto-Segmentation and Wound Face Regeneration Approach. *Comput. Model. Eng. Sci.* **2024**, *139*, 2197–2214. [CrossRef]
2. Le-Duc, T.; Lee, S.; Nguyen-Xuan, H.; Lee, J. A Hierarchically Normalized Physics-Informed Neural Network for Solving Differential Equations: Application for Solid Mechanics Problems. *Eng. Appl. Artif. Intell.* **2024**, *133*, 108400. [CrossRef]
3. Lee, Y.-H.; Chen, Y.-T. The Application of Artificial Intelligence Combined with Parametric Digital Design Tools in the Ceramic Modeling Design Process for Beginners—A Geometric Vase as an Example. In Proceedings of the Artificial Intelligence in HCI, Washington, DC, USA, June 29–4 July 2024; Degen, H., Ntoa, S., Eds.; Springer Nature: Cham, Switzerland, 2024; pp. 381–394.
4. Zeng, D.; Zhou, Z.; He, M.; Tang, C. Solution to Resolve Cognitive Ambiguity in Interactive Customization of Product Shape. *Int. J. Comput. Intell. Syst.* **2020**, *13*, 565–575. [CrossRef]
5. Kim, H.-S.; Cho, S.-B. Application of Interactive Genetic Algorithm to Fashion Design. *Eng. Appl. Artif. Intell.* **2000**, *13*, 635–644. [CrossRef]
6. Liu, X.; Du, Y. Towards Effective Feature Selection for IoT Botnet Attack Detection Using a Genetic Algorithm. *Electronics* **2023**, *12*, 1260. [CrossRef]
7. Wang, X.; Zhu, H. Active Disturbance Rejection Control of Bearingless Permanent Magnet Synchronous Motor Based on Genetic Algorithm and Neural Network Parameters Dynamic Adjustment Method. *Electronics* **2023**, *12*, 1455. [CrossRef]
8. Shojae Chaeikar, S.; Mirzaei Asl, F.; Yazdanpanah, S.; Zamani, M.; Manaf, A.A.; Khodadadi, T. Secure CAPTCHA by Genetic Algorithm (GA) and Multi-Layer Perceptron (MLP). *Electronics* **2023**, *12*, 4084. [CrossRef]
9. Chen, C.; Li, Z.; Wei, J. Estimation of Lithium-Ion Battery State of Charge Based on Genetic Algorithm Support Vector Regression under Multiple Temperatures. *Electronics* **2023**, *12*, 4433. [CrossRef]
10. González-Romera, E.; Romero-Cadaval, E.; Roncero-Clemente, C.; Milanés-Montero, M.-I.; Barrero-González, F.; Alvi, A.-A. A Genetic Algorithm for Residential Virtual Power Plants with Electric Vehicle Management Providing Ancillary Services. *Electronics* **2023**, *12*, 3717. [CrossRef]
11. Walsh, P.; Gade, P. Terrain Generation Using an Interactive Genetic Algorithm. In Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, 18–23 July 2010; pp. 1–7.
12. Yoon, D.; Kim, K.-J. 3D Game Model and Texture Generation Using Interactive Genetic Algorithm. *Comput. Entertain.* **2016**, *14*, 1–16. [CrossRef]
13. Zhu, X.; Li, X.; Chen, Y.; Liu, J.; Zhao, X.; Wu, X. Interactive Genetic Algorithm Based on Typical Style for Clothing Customization. *J. Eng. Fibers Fabr.* **2020**, *15*, 1558925020920035. [CrossRef]

14. Zhuo, Z.; Honglian, C. 3D Modeling Design and Rapid Style Recommendation of Polo Shirt Based on Interactive Genetic Algorithm. *J. Eng. Fibers Fabr.* **2020**, *15*, 1558925020966664. [CrossRef]
15. Fukumoto, M.; Hatanaka, T. A Proposal for Distributed Interactive Genetic Algorithm for Composition of Musical Melody. *Inf. Eng. Express* **2017**, *3*, 59–68. [CrossRef]
16. Huang, D.; Xu, X.; Zhang, Y.; Xia, X. Improved Interactive Genetic Algorithm for Three-Dimensional Vase Modeling Design. *Comput. Intell. Neurosci.* **2022**, *2022*, e6315674. [CrossRef] [PubMed]
17. Lv, J.; Zhu, M.; Pan, W.; Liu, X. Interactive Genetic Algorithm Oriented toward the Novel Design of Traditional Patterns. *Information* **2019**, *10*, 36. [CrossRef]
18. Gypa, I.; Jansson, M.; Wolff, K.; Bensow, R. Propeller Optimization by Interactive Genetic Algorithms and Machine Learning. *Ship Technol. Res.* **2023**, *70*, 56–71. [CrossRef]
19. Wei, Z.; Nie, J. Research on Intelligent Design Mechanism of Landscape Lamp with Regional Cultural Value Based on Interactive Genetic Algorithm. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e6273. [CrossRef]
20. Sheikhi Darani, Z.; Kaedi, M. Improving the Interactive Genetic Algorithm for Customer-Centric Product Design by Automatically Scoring the Unfavorable Designs. *Hum. Cent. Comput. Inf. Sci.* **2017**, *7*, 38. [CrossRef]
21. Sun, X.; Gong, D.; Jin, Y.; Chen, S. A New Surrogate-Assisted Interactive Genetic Algorithm With Weighted Semisupervised Learning. *IEEE Trans. Cybern.* **2013**, *43*, 685–698. [CrossRef]
22. Gong, D.; Sun, X.; Ren, J. Surrogate Models Based on Individual's Interval Fitness in Interactive Genetic Algorithms. *Chin. J. Electron.* **2009**, *18*, 689–694.
23. Deng, X.; Li, M.; Deng, S.; Wang, L. Hybrid Gene Selection Approach Using XGBoost and Multi-Objective Genetic Algorithm for Cancer Classification. *Med. Biol. Eng. Comput.* **2022**, *60*, 663–681. [CrossRef] [PubMed]
24. Wu, Z.; Zhou, M.; Lin, Z.; Chen, X.; Huang, Y. Improved Genetic Algorithm and XGBoost Classifier for Power Transformer Fault Diagnosis. *Front. Energy Res.* **2021**, *9*, 745744. [CrossRef]
25. Ghatasheh, N.; Altaharwa, I.; Aldebei, K. Modified Genetic Algorithm for Feature Selection and Hyper Parameter Optimization: Case of XGBoost in Spam Prediction. *IEEE Access* **2022**, *10*, 84365–84383. [CrossRef]
26. Gu, Z.; Cao, M.; Wang, C.; Yu, N.; Qing, H. Research on Mining Maximum Subsidence Prediction Based on Genetic Algorithm Combined with XGBoost Model. *Sustainability* **2022**, *14*, 10421. [CrossRef]
27. Li, X.; Dong, D.; Liu, K.; Zhao, Y.; Li, M. Identification of Mine Mixed Water Inrush Source Based on Genetic Algorithm and XGBoost Algorithm: A Case Study of Huangyuchuan Mine. *Water* **2022**, *14*, 2150. [CrossRef]
28. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794.
29. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
30. Huang, D.; Xu, X. IGAOD: An Online Design Framework for Interactive Genetic Algorithms. *SoftwareX* **2022**, *19*, 101205. [CrossRef]
31. Huang, S.; Cheng, J.; Yang, C.; Zhou, C.; Zhao, S.; Lu, X. Optimization Design of a 2.5 Stage Highly Loaded Axial Compressor with a Bezier Surface Modeling Method. *Appl. Sci.* **2020**, *10*, 3860. [CrossRef]
32. Gogoi, M.P.; Mukherjee, S.; Goswami, T.K. Analyses of Fold Profiles Using Cubic Bézier Curve. *Int. J. Earth Sci. (Geol. Rundsch.)* **2021**, *110*, 183–191. [CrossRef]
33. Gao, X.R. Bezier Surfaces and Texture Mapping Using Java 3D. *Adv. Eng. Forum* **2012**, *6–7*, 1000–1003. [CrossRef]
34. Foley, J.D. *Computer Graphics: Principles and Practice*; Addison-Wesley Professional: Glenview, IL, USA, 1996; ISBN 978-0-201-84840-3.
35. Le-Duc, T.; Nguyen, Q.-H.; Nguyen-Xuan, H. Balancing Composite Motion Optimization. *Inf. Sci.* **2020**, *520*, 250–270. [CrossRef]