A Generative Design Technique for Exploring Shape Variations

Shahroz Khan[a,*], Muhammad Junaid Awan[a]

[a]*Brainlabs Software, Taxila, Pakistan DOI.*

Abstract

Because innovative and creative design is essential to a successful product, this work brings the benefits of generative design in the conceptual phase of the product development process so that designers/engineers can effectively explore and create ingenious designs and make better design decisions. We proposed a state-of-the-art generative design technique (GDT), called *Space-filling-GDT* (Sf-GDT), for the creation of innovative designs. The proposed Sf-GDT has the ability to create variant optimal design alternatives for a given computer-aided design (CAD) model. An effective GDT should generate design alternatives that cover the entire design space. Toward that end, the criterion of space-filling is utilized, which uniformly distribute designs in the design space thereby giving a designer a better understanding of possible design options. To avoid creating similar designs, a weighted grid search approach is developed and integrated into the Sf-GDT. One of the core contributions of this work lies in the ability of Sf-GDT to explore hybrid design spaces consisting of both continuous and discrete parameters either with or without geometric constraints. A parameter-free optimization technique, called Jaya algorithm, is integrated into the Sf-GDT to generate optimal designs. Three different design parameterization and space formulation strategies; explicit, interactive, and autonomous, are proposed to set up a promising search region(s) for optimization. Two user interfaces; a

*Corresponding author

Email address: shahrozkhan2020@gmail.com, khansh@brainlabssoft.com (Shahroz Khan)

web-based and a Windows-based, are also developed to utilize Sf-GDT with the existing CAD software having parametric design abilities. Based on the experiments in this study, Sf-GDT can generate creative design alternatives for a given model and outperforms existing state-of-the-art techniques.

Keywords: Generative Design, Computer-Aided Design, Parametric Design, Space-filling Design, Jaya Algorithm

---

1. Introduction

Engineering or industrial product design is a complex process in which a design arrives at its final form after passing through a series of design phases. The conceptual phase is an initial and important component of these phases; it is recognized as a foundational step in any product development process. This phase can be complex and time-consuming if the appearance of the product under consideration is valuable to its target customers. To select an appealing design, designers often develop a number of design alternatives using two-dimensional (2D) sketches. However, the formulation of these alternatives is a critical and time-consuming task, especially for novice designers. To create these alternatives, designers have to develop and explore the entire design space effectively within a product's design requirements or the customer's preferences.

Exploration of design alternatives is recognized as a major characteristic of the conceptual design phase [1, 2]. Pahl et al. [3] categorized the conceptual phase into two sub-phases. In the first sub-phase, design alternatives are formalized based on the design requirements. In the second phase, these alternatives are ranked based on a preliminary analysis to select a potential design. Computer-aided design (CAD) is rarely used during this phase; it is primarily utilized later to analyze, validate, and fabricate the design [4]. For the most part, design engineers convert a design selected at the conceptual phase into a CAD model when they explore a narrow design space in order to analyze the performance of the design.

With recent advancements in artificial intelligence, optimization, design sim-

ulation, and parametric design techniques, the role of computers in the field of design is changing. In comparison to traditional CAD modeling techniques, these new techniques allow designers and engineers to iterate through a large number of design alternatives [5]. Generative design systems use these techniques to provide a promising way to explore design space to create alternative designs based on the specific performance objective defined by a user. A typical generative design system takes a problem definition as input and produce a single or set of optimal solutions for a given problem. Commercially available generative design systems, such as Altair's OptiStruct, solidThinking's Inspire, Siemen's Frustum, and efiForm [6], etc., are based on the topology optimization techniques [7]. These techniques are the mathematical methods that optimize a layout of material distribution within a predefined design space. Mostly, in these systems, the objective is to maximize/minimize compliance, the temperature at a certain point or globally, or minimize weight under volume, stress or displacement constraints. Typically, a generative system involves three steps to set up a problem. First, design engineer transforms 2D sketches into a three-dimensional (3D) CAD model. Then, various constraints and properties are defined based on the design specifications. Later, the design engineer executes generations to obtain a single or multiple optimization solutions. Different researchers have developed GDTs to create architectural structures [8], site layouts [9], and energy efficient [10] and eco-friendly building designs [11]. However, few studies[12, 4] have investigated how to create generative systems to explore designs based on their external form appearance.

Therefore, it is beneficial to develop a system that can automatically generate a variety of unique design alternatives for the outer form of a product based on its design requirements. The prime objective of this study is to develop a GDT that can effectively explore a design space and generate optimal aesthetically convincing design alternatives for a product at the conceptual phase of the design process. To develop the proposed technique, the following points were considered in order to make it effective. The proposed technique should:

1. Have an effective search and generation strategy to generate optimum design alternatives.
2. Be able to autonomously set up a viable design space for a given model.
3. Be able to work with both continuous and discrete design parameters.
4. Be able to create uniformly distributed and variant designs from the entire design space.
5. Have the ability to effectively explore both constrained and unconstrained design spaces.

By considering the points mentioned above, the present study proposes a new generative design technique, *Space-filling-GDT* (Sf-GDT). Sf-GDT has the ability to generate variant optimal designs. However, the decision on the selection of appropriate design parameters and setting a suitable design space for a given problem is critical. Therefore, Sf-GDT provides different space formulation strategies for the users to obtain optimal designs. Generative formulation of designs is a high-dimensional constrained optimization problem as there are generally a high number of design parameters and geometric constraints. Therefore, there is a need for a simple yet effective optimization approach that can search different optimum designs. Among many well-known optimization techniques, genetic algorithms have been widely used in generative systems. However, the performance of the genetic algorithms extensively depends on the selection of tuning parameters [13] and the proper tuning of these parameters requires an entirely different set of expertise, which most designers do not possess [12]. For this reason, we selected a newly proposed simple, effective, and parameter-free optimization approach called Jaya algorithm [14].

To generate N optimum design alternatives, the user first parameterizes the given CAD model and define a viable design space based on any of the three different proposed space formulation techniques. Within the defined design space, Sf-GDT randomly generates an initial population of solutions/designs, which consists of a further N subpopulations, one for each design alternative. Afterward, Sf-GDT applies the search strategy of the Jaya algorithm to each

subpopulation to converge the initial solutions to the optimum position in the design space while minimizing a cost/objective function, which ensures uniform and diverse design exploration. A weighted grid search approach is proposed, which enables Sf-GDT to maintain diversity between designs. Sf-GDT has also the ability to explore design space by synthesizing the design with different style forms, which can be implemented as discrete design parameters. To generate designs from the constraint spaces, Sf-GDT uses Deb's heuristic constrained handling rules [15]. Figure 1 illustrates the outcome of the proposed technique.

Following major contributions are made to Sf-GDT to enhance its ability for optimal creation of designs.

1. The search strategy of Jaya algorithm is extended to generate N optimal designs.
2. A weighted grid search technique is embedded in Sf-GDT to maintain diversity between designs.
3. The ability of Sf-GDT is enhanced to explore hybrid design spaces consisting of continuous and discrete design parameters.
4. Different design parameterization and space formulation strategies are proposed for an effective creation of design space.
5. Deb's [15] heuristic constraint handling rules are applied to generate designs in constrained spaces.
6. A web-based and Windows-based user interfaces are developed to utilize Sf-GDT with existing CAD software.

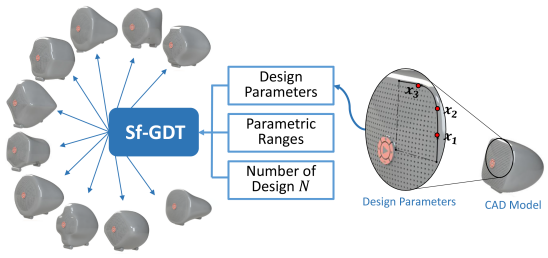

Figure 1: Illustration of the outcomes of Sf-GDT.

The remainder of this paper is organized as follows: Section 2 gives a comprehensive review of the relevant literature. Section 3 discusses the proposed approach to generating new designs. The numerical results of the proposed technique are given in Section 4. Section 5 describes the usage of the proposed technique with existing CAD software. Concluding remarks and opportunities for future work are presented in Section 6.

2. Related works

The proposed technique is inspired by the prior research in generative and space-filling design techniques and is based on Jaya algorithm. Below, we discuss some previous works done by different researchers in these fields.

2.1. Jaya Algorithm

Most of the well-known meta-heuristic optimization techniques require algorithm-specific parameters and proper tuning of these parameters is a critical factor, which affects their performance [13]. For example, the genetic algorithm uses selection operator, mutation and crossover probability; particle swarm optimization uses inertia weight, cognitive and social parameters; artificial bee colony uses the number of onlooker bees, scout bees, and employed bees. The Jaya algorithm does not require tuning of specific parameters except common controlling parameters like population size and a number of generations. This simplicity and tuning free nature of the Jaya algorithm make it suitable for generative design systems.

The optimization process in Jaya starts by randomly generating a population P of initial solutions for a given size s within the $n-dimensional$ defined design space. In order to achieve an optimum solution during the search, the algorithm always tries to move towards the best solution and moves away from the worst solution. Suppose for a specific problem there are n number of design parameters (i.e. $j = 1, 2, \ldots, n$) and s is the number of solutions (i.e. $k = 1, 2, \ldots, s$). If

the value of the j^{th} parameter for the k^{th} solution during the i^{th} iteration is represented as $X_{j,k,i}$, then this value is updated according to Equation (1).

$$X'_{j,k,i} = X_{j,k,i} + r_{1,k,i}(X_{j,best,i} - |X_{j,k,i}|) - r_{2,k,i}(X_{j,worst,i} - |X_{j,k,i}|) \quad (1)$$

$X_{j,best,i}$ and $X_{j,worst,i}$ are the updated values of the parameter j for the best and worst solutions, respectively. $X'_{j,k,i}$ is the updated value of $X_{j,k,i}$, and $r_{1,k,i}$ and $r_{2,k,i}$ are the two random numbers in the range [0,1]. At the end of each iteration i if $X'_{j,k,i}$ is better than $X_{j,k,i}$ then it is accepted otherwise rejected.

Several improvements have also been made on the Jaya algorithm in order to improve its performance and to expend its application in different fields. For example, Huang and Wang [16] introduced an elite opposition-based Jaya algorithm called EO-Jaya. EO-Jaya is a swarm intelligence based algorithm with no specific parameters to tune its performance. The elite opposition learning strategy was incorporated into EO-Jaya's solution updating phase, which enhances the solution diversity. A hybrid parallel Jaya algorithm for a multi-core environment called HHCP was developed by Michailidis [17]. HHCP Jaya has a hierarchical cooperation search mechanism to solve large-scale global optimization problems. Another version of Jaya called SAMP-Jaya algorithm was introduced by Rao and Saroj [18] for solving the constrained and unconstrained numerical and engineering optimization problems.

Jaya algorithm and its variations have also been implemented to different fields of science and engineering such as manufacturing [19], classification [20], power [21], combinatorial optimization [22] and topology optimization of truss structures [23].

2.2. Generative Design

To date, the field of generative design has been passed through the various advancements for different applications. Several GDTs have been proposed by different researchers for architectural applications and for the creation of a specific class of products. Apart from the techniques developed for the architectural

applications, here, we discuss some recent studies that are close to the proposed technique.

An exhaustive searched based GDT was proposed by Krish [12] for creating design alternatives. In which, designs are randomly searched in the design space and to generate dissimilar designs, the designer defines a threshold value, which is set on the Euclidean distance, between the generated designs. A major drawback of this technique lies in its exhaustive search strategy, which hinders designers from exploring and creating optimum design options. A practical generative design system called *DreamSketch* was developed by Kazi et al. [4] to support generative design at the conceptual phase. In DreamSketch, a user creates an initial design by sketching and then its alternatives are generated in the sketched context. In order to benefit from DreamSketch, a user requires possessing digital sketching abilities. A shape sampling technique, similar to ours, have been proposed by Gunpinar and Gunpinar [24], and Khan and Gunpinar [25]. However, these techniques lack the ability to work with discrete parameters and present no practical approach to design parametrization and design space formulation. Furthermore, the sampling technique of [25] is computationally expensive compared to the proposed technique. A biologically motivated algorithm was developed by Runions et al. [26] for the generative creation of leaf venation patterns. Sousa and Xavier proposed a symmetric-based generative technique for digital fabrication of geometric shapes like a triangular prism, cuboctahedron, and rhombicuboctahedron, etc.

In literature, techniques like shape grammars [27], shape syntheses [28] and L-systems [29] have been utilized by researchers to develop generative systems. Shape grammars are a generative method for creation of design alternatives by incorporating geometric logics/rules and have been utilized in different applications such as product design [30], architectural design [27], and embroidery design [31], etc. Despite being its usage for different application, shape grammars' usage is limited to the industry because of its computational complexity and difficulty in developing user interfaces [32]. L-systems are a variation of shape-grammars and has been used for different design problems such as com-

plex city planning [33] and computer pattern design [34]. L-systems are also based on the design rules applied in the form of a string. Among these methods, shape syntheses are preferable for creating a higher design variation of a given design. However, these techniques can only be employed for creating variations of existing designs/shapes. In which system is first trained on a large dataset of existing designs/shapes that are then synthesized to create variations.

2.3. Space-filling Design

There is a considerable amount of research that has been done on the optimal selection of space-filling Design of Experiments (DoE). However, most works done by researchers are proposed for the unconstrained design spaces. The research problem becomes more complicated when a selection of designs has to be performed in a constrained and high-dimensional design space like in the research of this paper. Fuerle and Sienz [35] proposed a method to produce designs in constrained spaces. However, this method is not feasible for high-dimensional problems more than $3D$. Draguljić et al. [36] proposed a CoNcaD algorithm for constructing non-collapsing and space-filling designs for bounded nonrectangular design spaces. Trosset [37] and Stinstra et al. [38] used maximin criterion for the construction of space-filling designs in the constrained 10-dimensional design space. The technique proposed by Trosset [37] and Stinstra et al. [38] does not guarantee the sampled DoE to be non-collapsing.

3. Proposed Technique

This section presents details of the proposed Sf-GDT that explores a design space to generate N designs. We first outline the core idea behind Sf-GDT approach and then the ability of Sf-GDT to explore constrained spaces with continuous and discrete design parameters will be explained.

3.1. The Sf-GDT

Basic terminologies are described first in relation to problem setting. A CAD model m can be represented by n number of design parameters $x_{m,1}, x_{m,2}, x_{m,3}, \ldots, x_{m,n}$.

Each design parameter defines a dimension in the design space. To form the design space limits, the upper and lower bounds for each design parameter are set. $[x^l_{m,j}]$ and $[x^u_{m,j}]$ represents the lower and upper bounds of the j^{th} design parameter, respectively, where $j = 1, 2, 3, \ldots, n$. Therefore, a $n - dimensional$ design space is formed by a set of n design parameters along with their lower and upper bounds.

To generate an optimal set of N design alternatives with the appropriate degree of dissimilarity, designs must be uniformly distributed with the maximum separating distance within the $n - dimensional$ design space. Therefore, a cost function based on the Audze and Eglais [39] technique is utilized, which follows a physical analogy: *Molecules in a space exert repulsive forces on each other that lead to potential energy in a space. These molecules are in equilibrium in case of minimum potential energy.* The analogous potential energy $U_1(B)$ for the creation of the space-filling designs is defined as:

$$U_1(B) = \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \frac{1}{L_{pq}^2} \tag{2}$$

where

$$L_{pq} = \sqrt{\sum_{j=1}^{n} (\overline{x}_{p,j} - \overline{x}_{q,j})^2} \tag{3}$$

Here, L_{pq} is the distance between the designs p and q, and $\overline{x}_{p,j}$ and $\overline{x}_{q,j}$ are the *scaled* parameter values for the j^{th} dimension of these designs, which are computed by scaling parameter values between 0 (i.e., lower bound for the parameter) and 1 (i.e., upper bound for the parameter). The design space formed from these bounds is called *scaled design space*. Recall that N is the number of designs to be generated and n is the number of dimensions in the design space.

The optimization problem for Sf-GDT can be formulated as the minimization of $U_1(B)$ to generate N optimum solutions (or designs). However, standard Jaya algorithm provides a single optimal solution by guiding the ini-

tial population of individuals to an optimum position. Therefore, the search strategy of Jaya algorithm has to modify in order to provide N optimum solutions. The optimal design creation process of Sf-GDT starts by creating the random initial population P consisting of N subpopulations (i.e., $P = [(p_1)_{s\times n},\ (p_2)_{s\times n},\ (p_3)_{s\times n},\ \ldots (p_N)_{s\times n}]^T$). $p_L = [X_1,\ X_2,\ \ldots X_s]^T$ denotes the L^{th} subpopulation of P and $L = 1, 2, 3, \ldots, N$. Each subpopulation consists of s solutions and the s^{th} solution X_s is comprised of n design parameters (i.e., $X_s = [x_{s,1},\ x_{s,2},\ \ldots x_{s,n}]$).

For each solution, there is a subpopulation of size s, during convergence all the N subpopulations are guided to their optimum position with Equation (1) under the consideration of their best and worst solutions. N worst and best solutions are selected, one from each subpopulation. The best and the worst solutions are the individuals that minimize and maximizes the cost function, receptively. The cost function is calculated based on the best solutions of the subpopulations. The division of population P into subpopulations is similar to [18, 25]. Let $B = [B_1, B_2, \ldots, B_N]$ and $W = [W_1, W_2, \ldots, W_N]$ are sets of best and worst solutions, respectively, and B_L and W_L is the best and worst solution for the L^{th} subpopulation. For the selection of N best and worst initial solutions, there are $2 \times s^N$ combinations. This means that the cost function has to be evaluated $2 \times s^N$ times. For instance, at $N = 10$ and $s = 40$ setting, $2 \times 10485760000000000$ evaluations of cost function has to be performed for the selection of $B = 10$ and $W = 10$ solutions. This can result in a high computational cost if N or s are assigned to a larger value. Therefore, an initial-designs-selection strategy is utilized for the selection of N initial worst and best solutions.

The initial-designs-selection strategy for the selection of N best initial solutions is based on the fact that the best individuals have the ability to select other exceptional individuals from a group. Following the similar analogy, N is first set to 2 in the cost function, and two individuals of the first two subpopulations that minimize the cost are selected as best solutions B_1 and B_2. Afterward, a solution that minimizes the cost function is selected as the best

solution B_3 from the third subpopulation. This solution is selected under the consideration of the preselected solutions B_1 and B_2 by setting $N = 3$. The selection process is repeated in a similar manner until N best solutions from the N number of subpopulations are determined. Similarly, this selection strategy is utilized to select N worst solutions, which maximizes the cost function. Note that this selection strategy checks $2 \times s^2 + \sum_2^N s$ individuals' combinations to select each set of N best and worst initial solutions.

In Sf-GDT, the optimization process in any iteration is completed by performing the N number of sub-iterations, one for each subpopulation. Each subpopulation moves towards the better position in design space individually while keeping the best and worst solutions of other subpopulations the same. During optimization, a new position for a solution is found using Equation (1). Let X_k and X'_k be the current and new positions of a solution in the first subpopulation, respectively. The new position of the solution is accepted if the cost value of $B' = [X'_k, B_2, \ldots, B_N]$ is less than $B = [X_k, B_2, \ldots, B_N]$. The best B and worst W solutions are updated after each sub-iteration for the subpopulation. The best solution is an individual having a minimum cost value (computed with the best solutions of other subpopulations) among the other solutions in the same subpopulation. Sub-iterations in other subpopulations are performed in the same way. An iteration is completed when a sub-iteration for each of the subpopulation is performed. After Sf-GDT stops the convergence process the best solutions of the subpopulations are regarded as final optimal designs.

Furthermore, the alternatives obtained from Sf-GDT can work as Design of Experiments (DoE) for physics simulations, which can be run for validation of designs' functionality, structural integrity, and usability. DoE are crucial in physical analyses, which has the major goal of determining which design parameters have more effect on the simulation results. Most analyses are computationally expensive, and running the analysis for collapsing designs and non-space-filling would ultimately result in an unnecessary computational effort [36].

3.2. Weighted Grid Search Technique

Minimization of $U_1(B)$ favors placement of the designs at the maximum separating distance from each other. In the case of high-dimensional design space, this function itself locates some designs at the boundaries of the design space [40]. This will result in the violation of a non-collapsing criterion [36] (i.e., designs not sharing any parameter values within a specific interval), thereby generating similar designs.

For generative designs, it is desired to spread designs evenly also in the inner portions of the design space. Therefore, the non-collapsing criterion for the generated designs should be satisfied as much as possible. A weighted grid search technique is introduced in order to generate non-collapsing designs in the design space. A new term, $U_2(B)$, is included in the cost function, which is as follows:

$$U_2(B) = \alpha \times \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \sum_{j=1}^{n} f(y_{p,j}, y_{q,j}) \tag{4}$$

$$f(y_{p,j}, y_{q,j}) = \begin{cases} 1 & \text{if } y_{p,j} = y_{q,j} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where

$$\begin{cases} \text{if } \; x_{p,j}^{e} \leq x_{p,j} < x_{p,j}^{e+1} \; \text{ then } \; y_{p,j} = e \\ \text{if } \; x_{q,j}^{e} \leq x_{q,j} < x_{q,j}^{e+1} \; \text{ then } \; y_{q,j} = e \end{cases} \tag{6}$$

The term $U_2(B)$ is based on the degree of violation for the non-collapsing criterion. Here, α is a user-defined parameter adjusting weight of the $U_2(B)$ term. $y_{p,j}$ and $y_{q,j}$ in Equation (4) are the corresponding integer coordinate values for $x_{p,j}$ and $x_{q,j}$ in the j^{th} dimension, respectively. To calculate $y_{p,j}$ and $y_{q,j}$ the range of each design parameter is partitioned into N equal intervals (levels) as follows: $[x_{m,j}^{l} = x_{m,j}^{1}, x_{m,j}^{2}, \ldots, x_{m,j}^{N} = x_{m,j}^{u}]$ and an integer coordinate e is assign to them using Equation (6), where e ranges from 1 to N. Based

on these integer values, the piecewise function f in Equation (5) decides if the designs p and q are collapsing or non-collapsing.

Maximum value for this term can be $n \times \binom{N}{2}$. $\binom{N}{2}$ represents the combinations between designs, which is as follows: $\binom{N}{2} = \frac{N!}{2!(N-2)!}$. Setting the parameter α to small values will lead to semi non-collapsing designs and larger values will produce more non-collapsing designs. The cost function $U(B)$, which is given in Equation (7), have to minimize to create space-filling and non-collapsing designs. This function is overall composed of a parameter α, and $U_1(B)$ and $U_2(B)$ for space-filling and non-collapsing criteria, receptively. Algorithm 1 summarizes the step-wise procedure of Sf-GDT.

$$Minimize \quad U(B) = \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \frac{1}{L_{pq}^2} \quad + \quad \alpha \times \sum_{p=1}^{N-1} \sum_{q=p+1}^{N} \sum_{j=1}^{n} f(y_{p,j}, y_{q,j}) \quad (7)$$

Figure 2 (a) shows a 3D CAD model parameterized with two design parameters x_1 and x_2. The design parameter and their parametric ranges ($[x_1^l] \leq x_1 \leq [x_1^u]$ and $[x_2^l] \leq x_2 \leq [x_2^u]$) forms a 2D design space. 20 design alternatives for this CAD model are created using the proposed Sf-GDT under the space-filling criterion (Equation 2), Non-collapsing criterion (Equation 4) and combined space-filling and non-collapsing criteria (Equation 7), which are shown in Figure 2 (b), (c) and (d), respectively. It should be noted that each point/dot in the Figure 2 represents a position of a design in the design space. In Figure 2 (a), it can be seen that the design alternatives are spread evenly in the design space, therefore, space-filling designs can be obtained using Sf-GDT. However, in this case, there are more designs are the boundary of the design. Therefore, exploration performed only based on this criterion may not produce satisfactory designs because during space exploration designer desires to obtain designs that also evenly covers the inners regions of the design spaces. Furthermore, the design alternatives in Figure 2 (c) are created with the only non-collapsing criterion and has resulted in designs with the poor space-filling property. It can be observed in the Figure 2 (c) that the better space exploration is achieved when the when both space-filling and non-collapsing properties are considered (i.e.

when space exploration is performed using the Equation 7, which involves both space-filling and non-collapsing criterion). Later, the 3D CAD model in Figure 2 (e) is parameterize with design parameters x_1, x_2 and x_3. Here, these three design parameters form a 3D design space. 20 design alternatives are created in the 3D space using the Sf-GDT (see Figure 2 (f)). Again, the points in this space represent the positions of the 20 design alternatives.

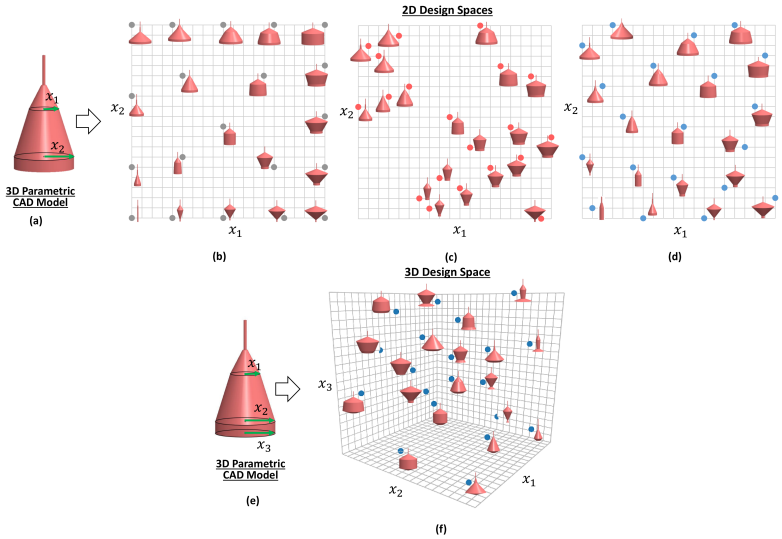

Figure 2: Design alternatives for a 3D CAD model with two design parameters (a) are obtained in 2D spaces considering; (b) only space-filling criterion, (c) only non-collapsing criterion, and both space-filling and non-collapsing criteria using Sf-GDT (d). Design alternatives for the same CAD model with three parameters (e) are generated in 3D design space using Sf-GDT while considering both space-filling and non-collapsing criteria (f).

3.3. Sf-GDT for Discrete Parameters

Sf-GDT also gives the ability to the designer to explore design space by synthesizing the design with different "style" profiles (e.g., round, triangular, and rectangular, etc). The designer can add an option for variable base styles that can be implemented as discrete design parameters. Sf-GDT is customized in the following way in order to be employed for the discrete design parameters.

Suppose, an integer value (round→1, rectangular→2, and triangle→3) is

Algorithm 1 The pseudo-code of Sf-GDT

Create an input CAD model and parameterize it with n design parameters $(x_1, x_2, \ldots, x_n)$.
Initialize the number of parameters (n), parameter ranges, number of design to be created (N), subpopulation size (p) and parameter α.
Randomly create an initial population P of feasible solutions/designs within the parametric ranges consisting of N subpopulations $(p_L)_{s\times n}$ of size s, where $1 \leq L \leq N$.
Obtain set of N initial best (B) and worst (W) designs, one from each subpopulation based on the initial-designs-selection strategy.
$B = [B_1, B_2, \ldots, B_N]$, $W = [W_1, W_2, \ldots, W_N]$
while termination criterion is not satisfied **do**
  for $L = 1$ to N **do**
    for $k = 1$ to s **do**
      Update the design X_k of $(p_L)_{s\times n}$ using Equation 1 based on the B_L and W_L and obtain an updated/new design X'_k.
      Calculate the cost value $U(B')$ and $U(B)$ using Equation 7 for $B = [X'_1, B_2, \ldots, B_N]$ and $B = [X_1, B_2, \ldots, B_N]$.
      if $U(B') < U(B)$ **then**
        Accept the design X'_k.
      else
        Accept the design X_k.
      end if
    end for
    Obtain the updated $(p_L)_{s\times n}$, which is $(p'_L)_{s\times n}$.
    Find the new best B'_L and worst W'_L solutions from $(p'_L)_{s\times j}$.
    Replace B_L and W_L with B'_L and W'_L in the initial set ($B = [B'_1, B_2, \ldots, B_N]$, $B = [W'_1, W_2, \ldots, W_N]$).
  end for
end while
Final N optimal designs are obtained.

assigned to each of three styles. Let $x_{p,d}$ be the d^{th} discrete parameter of design p containing t styles and $[x^l_{p,d}]$ and $[x^u_{p,d}]$ are the lower and upper bounds for $x_{p,d}$, respectively. In the above case, $t = 3$, $[x^l_{p,d}] = 1$ and $[x^u_{p,d}] = 3$. Instead of dividing this parameter into N number of intervals, it should be divided into

t number of styles. Now, the range of $x_{p,d}$ is divided into t equal number of intervals as follows: $[x^l_{p,d} = x^1_{p,d}, x^2_{p,d}, \ldots, x^t_{p,d} = x^u_{p,d}]$. After Sf-GDT converge, all the design parameters, including $x_{p,d}$, of each design consists of continuous values. The parameter $x_{p,d}$ contains the style profiles in the form of discrete values and required to be converted to discrete values. Otherwise, no decision can be made on the selection of the style shape. Therefore, after generating N designs, continuous values of $x_{p,d}$ for each design will be converted into discrete values by using Equation (8).

$$\text{if} \quad x^r_{p,d} \leq x_{p,d} < x^{r+1}_{p,d} \quad \text{then} \quad x_{p,d} = r \tag{8}$$

Here, r in an integer number ranging from 1 to t.

3.4. Generation of Design from Constrained Spaces

The design space consists of feasible and infeasible regions in the presence of geometric constraints. Feasible regions consist of feasible designs that satisfy the predefined constraints. Infeasible designs are located in the infeasible regions. There are different types of constraint handling techniques are available in the literature, such as the incorporation of static penalties, dynamic penalties, adaptive penalties etc. In this study, Deb's heuristic constrained handling method [15] is adopted in order to avoid Sf-GDT from selecting designs from constrained spaces. Deb's method uses a tournament selection operator in which two solutions are selected and compared with each other. A design p is said to be constrained-dominate other design q if any of the following heuristic rules are true:

1. Design p is feasible and design q is not.
2. Designs p and q both are infeasible but design p violate less number of constraints.
3. Designs p and q both are feasible but design p has better cost function value.

If design p constrained-dominate design q then design p is selected. This domination is checked at the end of each sub-iteration. There can be the case when both designs, p and q, are infeasible and have the same number of constraint violations then the design with better cost value is selected. In case of constraint space, Sf-GDT generates an initial population P consisting of only feasible solutions. So, during the selection of the initial best and worst solutions, the initial-designs-selection strategy does not have to check these constrained handling rules.

3.5. Design Parameterization

An effective design parameterization of a CAD model is required to create variant designs. All the important features of the design should be parameterized with the appropriate number of parameters. However, a decision on the suitable set of parameters is a critical step in the parametrization, which requires the strong understanding of the design requirement and key attributes. There are different techniques available in the literature on how to form a well-structured parametric model [41]. A well-structured model can enable the designer to create a variety of design alternatives within its design requirement than a poorly structured model. The high number of design parameters may not keep the original form of the design. As mostly designers desire to keep the common underlying structure of the model while generating its alternatives. On the other hand, less number of parameters can narrow down the design space and larger variation of designs may not be achieved. Therefore, the decision on the selection of appropriate design parameters should be carefully made.

One strategy, which the designer can follow, is to first detect the important features of a given model and then these features can be parametrized with a relatively higher number of parameters and designs can be generated with these parameters. Later, after some trials, the designer can detect quixotic parameters and eliminate them by directly modifying the CAD model. Such capability of the generative design system is recognized as 'designerly' method, which allows designers to modify the model under consideration and use its

generative capabilities at any phase of the design process [12]. After exploring the designs based on the important features, later, if required, design space can be explored based on its nominal features.

3.6. Formulation of design space

As stated before, the design space for any CAD model is formed by the number of the design parameters and their bounds. The dimensionality of the design space depends on the number of design parameter used to define the CAD model and the limits of the design space are set by defining the upper and lower bounds for each design parameter. However, formulation of a suitable design space is a decisive task as the performance of a technique in term of creating better design alternatives mainly depends on it. Setting up the design space should be carefully done in order to achieve the maximum performance of the Sf-GDT and should have sufficient high potential region. If design space is too narrow then Sf-GDT will result in the creation of similar/same designs. On the other hand, a vast design space can result in the waste of computational effort in exploring undesirable regions of the design space. Typically, a design space is set up by defining the upper and lower bounds of the design parameters. Where each parameter represents a dimension in the design space. Defining the upper and lowers bounds usually done based on the initial design specifications and designers' understanding of the design.

In Sf-GDT, design space formulation can happen in three different way; explicit formulation, autonomous formulation, and interactive formulation.

Explicit Formulation: The explicit formulation of the design space happens when the design specifications are known at the conceptual stage and based on these specifications the designer limits the space.

Autonomous Formulation: The autonomous formulation helps to coarsely form the design space as a percentage of the initial parameter values of the design. This formulation happens when no primary understanding of the design specifications are available in the conceptual phase. The autonomous formulation gives a good initial guess of suitable space limits. With this formulation,

the designer can first inadequately build up an initial map of promising regions of the design space and then explore designs in that space. Afterward, the designer can further reform the design space based on the previous exploration results. There can be some infeasible designs in the autonomously formalized space, but this can be overridden by implementing geometric constraints.

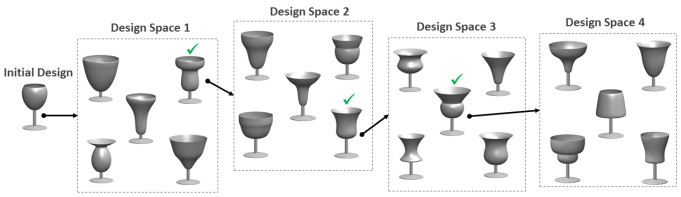

Figure 3: Interactive formulation of design space.

Interactive Formulation: In the interactive formulation of the design space, the designer creates multiple spaces and gradually proceeds to a final design. First, the designer can autonomously form an initial design space around the given CAD model and creates designs in this space. Afterward, the designer can select a design and then formalize an autonomous space around that design. In this way, the designer can interactively proceed by selecting designs and forming the design spaces until he/she achieves a final desired design. For example, Figure 3 gives the illustration of the interactive formulation of the design space. In which initial space (design space 1) is formed around the initial design. A design (marked in green) is selected from this space and then a new space (design space 2) is formed around the previously selected design. This process continues until the final design is achieved. During selection, if the designer selects more than one design, then a new design space is created around the centroid of the selected designs. The designer can also refine or shrinks the space after each interaction as he/she approaches the final design. Once the final design is selected then, if desired, it can be further modified easily due to its parametric nature.

4. Results and Discussion

In this section, we first demonstrated a step-wise procedure for implementing Sf-GDT on a simple 3D CAD model and then discuss the performance of Sf-GDT for different test models and settings. The proposed technique has also been compared with the existing state-of-the-art techniques.

4.1. Implementation of the Sf-GDT

To generate design alternatives with Sf-GDT, first, develop an input CAD model, shown in Figure 2 (e). This CAD represents a celling lamp and parameterized with three design parameters, x_1, x_2 and x_3. Each design parameter denotes the radius of the circular region of the lamp model. Then, form an explicit design space for this model my defining the parametric ranges as $1 \leq x_1 \leq 20$, $1 \leq x_2 \leq 20$ and $1 \leq x_3 \leq 15$, and finally, perform following steps to generate design alternatives for this CAD model:

Step 1: Initialize the following parameters:

- Subpopulation size (s) = 2
- Number of designs (N) = 4
- Weight parameter (α) = 5
- Number of design parameters (n) = 3
- Ranges of design parameters

Step 2: Randomly generate a population P consists of N subpopulations. Each subpopulation contains $s = 2$ initial designs/solutions X_1^g and X_2^g. The super script g represents the subpopulation to which these solutions belong. The initial population is shown below:

$$\mathrm{P} = \begin{bmatrix} (p_1)_{2\times 3} & (p_2)_{2\times 3} & (p_3)_{2\times 3} & (p_4)_{2\times 3} \end{bmatrix}^T$$

$$(p_1)_{2\times 3}=\begin{bmatrix}X_1^1\\X_2^1\end{bmatrix}=\begin{bmatrix}6.4 & 9.8 & 5.0\\3.9 & 16.1 & 13.6\end{bmatrix}\quad (p_2)_{2\times 3}=\begin{bmatrix}X_1^2\\X_2^2\end{bmatrix}=\begin{bmatrix}13.6 & 2.5 & 5.3\\19.1 & 5.4 & 10.3\end{bmatrix}$$

$$(p_3)_{2\times 3}=\begin{bmatrix}X_1^3\\X_2^3\end{bmatrix}=\begin{bmatrix}7.4 & 11.9 & 3.7\\10.9 & 17.0 & 12.7\end{bmatrix}\quad (p_4)_{2\times 3}=\begin{bmatrix}X_1^4\\X_2^4\end{bmatrix}=\begin{bmatrix}16.6 & 18.4 & 5.6\\19.8 & 17.0 & 10.7\end{bmatrix}$$

Step 3: Select an initial set of best and worst solutions, one from each subpopulation, using the initial-designs-selection strategy described in Section 3. This strategy works as follows:

1. Calculate the cost value $U(B)$ using Equation 7 for $s^2 = 4$ combinations of solutions in population p_1 and p_2. Then select a combination which gives lowest (highest) value of the cost as best (worst) solution. First, calculate the potential energy $U_1(B)$ using Equation 2 and number of collapsing designs $U_2(B)$ using Equation 4 and then input these values in Equation 7 to calculate $U(B)$.

   Calculate cost value for $B = [X_1^1, X_1^2]$:

   Scale X_1^1 and X_1^2 between 0 and 1 $X_1^1=\begin{bmatrix}6.4\\9.8\\5.0\end{bmatrix}^T\rightarrow\begin{bmatrix}0.28\\0.46\\0.29\end{bmatrix}^T$ $X_1^2=\begin{bmatrix}13.6\\2.5\\5.3\end{bmatrix}^T\rightarrow\begin{bmatrix}0.66\\0.08\\0.31\end{bmatrix}^T$

   $$U_1(B)=\sum_{p=1}^{1}\sum_{q=p+1}^{2}\frac{1}{L_{pq}^2}=\frac{1}{L_{12}^2}$$

   Distance between first and second design of $B=L_{12}=\sqrt{(0.28-0.66)^2+(0.46-0.08)^2+(0.29-0.31)^2}=0.54$

   $$U_1(B)=\frac{1}{L_{12}^2}=3.4\quad U_2(B)=1.0$$

   Cost function $U(B)=U_1(B)+\alpha\times U_2(B)=8.4$

   Similarly, calculate cost for $B = [X_1^1, X_2^2]$, $B = [X_2^1, X_1^2]$ and $B = [X_2^1, X_2^2]$:

   $$B=[X_1^1,X_2^2]\rightarrow U(B)=9.5$$
   $$B=[X_2^1,X_1^2]\rightarrow U(B)=0.9$$
   $$B=[X_2^1,X_2^2]\rightarrow U(B)=1.0$$

   Solution set $[X_2^1, X_1^2]$ ($[X_1^1, X_2^2]$) give lowest (highest) cost, therefore, X_2^1 (X_1^1) and X_1^2 (X_2^2) are regarded as the best (worst) solutions of p_1 and p_2, respectively.

2. Under the consideration of X_2^1 (X_1^1) and X_1^2 (X_2^2) find a best (worst) solution of p_3.

   Calculate cost for $B = [X_2^1, X_1^2, X_1^3]$:

   $X_2^1=\begin{bmatrix}3.9\\16.1\\13.6\end{bmatrix}^T\rightarrow\begin{bmatrix}0.15\\0.79\\0.90\end{bmatrix}^T$ $X_1^2=\begin{bmatrix}13.6\\2.5\\5.3\end{bmatrix}^T\rightarrow\begin{bmatrix}0.66\\0.08\\0.31\end{bmatrix}^T$ $X_1^3=\begin{bmatrix}7.4\\11.9\\3.7\end{bmatrix}^T\rightarrow\begin{bmatrix}0.34\\0.57\\0.19\end{bmatrix}^T$

$$U_1(B)=\sum_{p=1}^{2}\sum_{q=p+1}^{3}\frac{1}{L_{pq}^2}=\frac{1}{L_{12}^2}+\frac{1}{L_{23}^2}$$
$$L_{12}=\sqrt{(0.15-0.66)^2+(0.79-0.08)^2+(0.90-0.31^2}=1.06$$
$$L_{23}=\sqrt{(0.66-0.34)^2+(0.08-0.57)^2+(0.31-0.19)^2}=0.85$$
$$U_1(B)=\frac{1}{L_{12}^2}+\frac{1}{L_{23}^2}=2.3 \quad U_2(B)=1.0$$
$$U(B)=U_1(B)+\alpha\times U_2(B)=7.3$$

Similarly, calculate cost for $B=[X_2^1,X_1^2,X_2^3]$:

$$B=[X_2^1,X_1^2,X_2^3]\rightarrow U(B)=24.1$$

The solution X_1^3 (X_2^3) give lowest (highest) cost value and thus regarded as best solution of p_3

3. Select the best (worst) solution of the subpopulation p_4

$$B=[X_2^1,X_1^2,X_1^3,X_1^4]\rightarrow U(B)=20.7$$
$$B=[X_2^1,X_1^2,X_1^3,X_2^4]\rightarrow U(B)=14.2$$

The best (worst) solution of p_4 is X_2^4 (X_1^4).

4. The initial best (worst) solution set is $B=[B_1,B_2,B_3,B_4]=[X_2^1,X_1^2,X_1^3,X_2^4]$ ($W=[W_1,W_2,W_3,W_4]=[X_1^1,X_2^2,X_2^3,X_1^4]$).

Step 4: Update solution X_1^1 of p_1 based on its best and worst solutions using Equation 1.

$$X_1'^1=X_1^1+r_1(B_1-|X_1^1|)-r_2(W_1-|X_1^1|)=\begin{bmatrix}4.9\\12.3\\9.6\end{bmatrix}^T$$

Where

$$r_1=\begin{bmatrix}0.6\\0.4\\0.5\end{bmatrix}^T \quad r_2=\begin{bmatrix}0.02\\0.5\\0.6\end{bmatrix}^T$$

Step 5: Calculate the cost $U(B')$ and $U(B)$ for $B=[X_1'^1,B_2,\ldots,B_N]$ and $B=[X_1^1,B_2,\ldots,B_N]$, respectively.

$$B'=[X_1'^1,B_2,B_3,B_4]\rightarrow U(B')=23.7$$
$$B=[X_1^1,B_2,B_3,B_4]\rightarrow U(B)=64.1$$

As $U(B')<U(B)$ so accept the new solution $X_1'^1$ and reject the old solution X_1^1.

Step 6: Similarly, update the solution X_2^1 of p_1.

$$X_2'^1 = X_2^1 + r_1(B_1 - |X_2^1|) - r_2(W_1 - |X_2^1|) = \begin{bmatrix} 3.8 \\ 19.6 \\ 12.7 \end{bmatrix}^T$$

Step 7: Calculate the cost $U(B')$ and $U(B)$ using $B = [X_2'^1, B_2, \ldots, B_N]$ and $B = [X_2^1, B_2, \ldots, B_N]$, respectively.

$$B' = [X_2'^1, B_2, B_3, B_4] \rightarrow U(B') = 14.0$$
$$B = [X_2^1, B_2, B_3, B_4] \rightarrow U(B) = 14.2$$

$U(B') < U(B)$, so accept the new solution $X_2'^1$.

Step 8: Obtain the updated subpopulation p_1'

$$(p_1')_{2\times 3} = \begin{bmatrix} X_1'^1 \\ X_2'^1 \end{bmatrix} = \begin{bmatrix} 4.9 & 12.3 & 9.6 \\ 3.8 & 19.6 & 12.7 \end{bmatrix}$$

Step 9: Find the new best (B_1) and worst (W_1) solutions of p_1'.

$$B_1 = X_1'^1 = \begin{bmatrix} 4.9 \\ 12.3 \\ 9.6 \end{bmatrix} \quad W_1 = X_2'^1 = \begin{bmatrix} 3.8 \\ 19.6 \\ 12.7 \end{bmatrix}$$

Step 10: Replace B_1 and W_1 with B_1' and W_1' in the initial set ($B = [B_1', B_2, B_3]$, $B = [W_1', W_2, W_3]$).

Step 11: Repeat the steps 4 to 10 to obtain p_2', p_3' and p_4'.

Step 12: Repeat the steps 4 to 11 until the change in the cost function becomes negligibly small between a few consecutive iterations. After 13^{th} iteration algorithm converges and the best solution of each subpopulation is regarded as final optimum design.

Step 13: Obtain final design alternatives, which are shown below:

$$B' = [B_1, B_2, B_3, B_4]$$

$$B_1 = \begin{bmatrix} 1.1 \\ 17.3 \\ 15 \end{bmatrix}^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^T \rightarrow B_2 = \begin{bmatrix} 13.6 \\ 2.5 \\ 4.5 \end{bmatrix}^T \rightarrow B_3 = \begin{bmatrix} 5.8 \\ 11.2 \\ 3.6 \end{bmatrix}^T \rightarrow B_4 = \begin{bmatrix} 19.9 \\ 17.2 \\ 10.7 \end{bmatrix}^T \rightarrow$$

4.2. Test Models

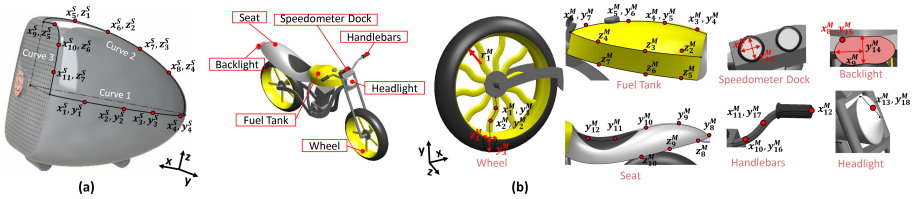

Figure 4: CAD models of (a) speaker and (b) Motorbike with their design parameters.

To validate the performance of Sf-GDT we also utilized more test models, such as a speaker, a motorbike, a ceiling lamp, and a wine glass, which are shown in Figure 4 (a), (b), Figure 6 (a) and Figure 7 (a), respectively. These models were selected based on their aesthetic importance. A wine glass defines elegance of the wine drinker, an aesthetic ceiling lamp and an elegant speaker box and motorbike design can attract more customers. Except for the motorbike, these models are single component 3D designs. Where the bike model is composed of several design components. For the complex test model like a motorbike, a user can first work on the low-level details of the design and then can move to the high-level details. For example, the user can first explore the form outline of the design using Sf-GDT and once a collection of different initial base forms is selected, the designer can then explore further design details by keeping the base form constant. The user may also first explore the design space to create design alternatives for each component and then assembles these alternative parts to create the final design. For the motorbike model, only components for that outer appearance is considered to be significant are created such as fuel

tank, seat, wheels, headlight, backlight, handlebars and speedometer dock. 3D surfaces of the wine glass, ceiling lamp, speaker, fuel tank and seat of motorbike models are created by interpolating Coons patches between spline curves and design parameters are defined with these curves. The motorbike's front and rare wheels are the 3D solid models.

The speaker model shown in Figure 4 (a) is represented using 22 design parameters ($n = 22$). The speaker model is created using three spline curves. First, a quarter section of the speaker model is created using these spline curves. Then, this section is mirrored first along the x-y plane and then mirrored along x-z plane. Curve 1 lies in x-y plane and position of its control points is represented by the parameter x_1^S, y_1^S, x_2^S, y_2^S, x_3^S, y_3^S, x_4^S and y_4^S and Curve 2 lies in x-z plane and parameters x_5^S, z_1^S, x_6^S, z_2^S, x_7^S, z_3^S, x_8^S and z_4^S denote the position of its control points. Similarly, x_{11}^S, z_7^S, x_{10}^S, z_6^S, x_9^S and z_5^S represents the control point position of curve 3. The parameter ranges of the speaker model are given in Table 1.

Each component of the motorbike model is parameterized separately and consist of total 42 design parameters ($n = 42$), which are shown in Figure 4 (b). Back wheel is parameterized with 6 continuous design parameter (x_1^M, y_1^M, x_2^M, y_2^M, y_3^M and r_1^M) and one discrete parameter (r_2^M), and front wheel is created as copy of the back wheel. x_1^M,x_2^M and y_1^M,y_2^M represent the position of control points in x-axis and y-axis, respectively, and y_3^M and r_1^M are the width of the tire and radius of the wheel. The discrete parameter, r_2^M, defines the number of spokes. The fuel tank is created using three spline curves, one in the y-x plane and two in x-z plane and represented with 14 design parameters. Similarly, the seat of the motorbike is parameterized with 8 parameters and created using two spline curves, one in 3D space and other in the x-y plane. The design parameters x_7^M and y_{13}^M denotes the width of the speedometer dock in x-axis and y-axis. The backlight and headlight are represented with x_8^M, x_9^M, y_{14}^M, y_{15}^M and x_{13}^M, y_{18}^M, receptively, where x_9^M and y_{14}^M adjusts the length and width of the backlight. The handlebars are also created with spline curves with design parameters x_{10}^M, x_{16}^M, x_{11}^M, x_{17}^M and x_{12}^M representing the position of control

points. The parametric ranges of each design parameter are provided in Table 1.

Table 1: Parameter ranges for the test models

Speaker Model				
$5 \leq x_1^S \leq 150$	$5 \leq y_1^S \leq 90$	$5 \leq x_2^S \leq 150$	$5 \leq y_2^S \leq 90$	$5 \leq x_3^S \leq 150$
$5 \leq y_3^S \leq 90$	$5 \leq x_4^S \leq 150$	$5 \leq y_4^S \leq 90$	$5 \leq x_5^S \leq 150$	$5 \leq z_1^S \leq 90$
$5 \leq x_6^S \leq 150$	$5 \leq z_2^S \leq 90$	$5 \leq x_7^S \leq 150$	$5 \leq z_3^S \leq 90$	$5 \leq x_8^S \leq 150$
$5 \leq z_4^S \leq 90$	$5.0 \leq x_9^S \leq 90$	$5 \leq z_5^S \leq 90$	$5 \leq x_10^S \leq 90$	$5 \leq z_6^S \leq 90$
$5 \leq x_11^S \leq 90$	$5 \leq z_7^S \leq 90$			
Motorbike Model				
$6 \leq r_1^M \leq 11$	$20 \leq x_1^M \leq 25$	$3.5 \leq y_1^M \leq 6$	$20 \leq x_2^M \leq 25$	$3.5 \leq y_2^M \leq 5$
$1.8 \leq y_3^M \leq 2.3$	$1 \leq x_3^M \leq 2$	$5 \leq y_4^M \leq 7$	$3 \leq x_4^M \leq 8$	$6 \leq y_5^M \leq 12$
$2 \leq x_5^M \leq 6$	$6 \leq y_6^M \leq 12$	$10 \leq x_6^M \leq 12$	$6 \leq y_7^M \leq 12$	$4 \leq z_2^M \leq 6$
$4 \leq z_3^M \leq 6$	$3 \leq z_4^M \leq 5$	$3.5 \leq z_5^M \leq 5.5$	$3 \leq z_6^M \leq 6$	$3.5 \leq z_7^M \leq 5$
$15 \leq y_8^M \leq 20$	$17 \leq y_9^M \leq 23$	$15 \leq y_10^M \leq 20$	$8 \leq y_{11}^M \leq 13$	$10 \leq y_{12}^M \leq 15$
$3 \leq z_8^M \leq 7$	$3 \leq z_9^M \leq 6.5$	$2 \leq z_{10}^M \leq 6$	$2 \leq x_7^M \leq 3.5$	$2 \leq y_{13}^M \leq 3.5$
$1.5 \leq y_{14}^M \leq 3$	$2.5 \leq x_8^M \leq 4$	$1 \leq y_{15}^M \leq 2$	$4 \leq x_9^M \leq 6$	$2 \leq x_{10}^M \leq 5$
$0.8 \leq y_{16}^M \leq 3$	$1 \leq x_{11}^M \leq 4$	$2 \leq y_{17}^M \leq 4$	$7.5 \leq x_{12}^M \leq 12$	$4 \leq x_{13}^M \leq 7$
$2 \leq y_{18}^M \leq 5$	$1 \leq r_2^M \leq 7$			
Celling Lamp Model				
$1 \leq y_1^L \leq 10$	$1 \leq y_2^L \leq 10$	$1 \leq y_3^L \leq 10$	$1 \leq r_1^L \leq 20$	$1 \leq r_2^L \leq 20$

Sf-GDT was tested for both constrained and unconstrained design spaces with different algorithm setting and design space formulation. Design alternatives for the speaker and motorbike models were created with the application of Sf-GDT in the explicitly formalized unconstrained space and can be seen in Figure 5. A careful inspection of the designs in Figure 5 can reveal that the generated alternatives by Sf-GDT for each model are distinct from each other to a great extent. This validates the ability of Sf-GDT to create distinct designs for any given CAD model. Table 2 provides the algorithm settings and the values of various parameters/criteria such as design alternative (N) and design parameters (n) for the test models, $U_1(B)$ and $U_2(B)$ values, computational time and number of iterations (i) performed while creating the designs alternatives.

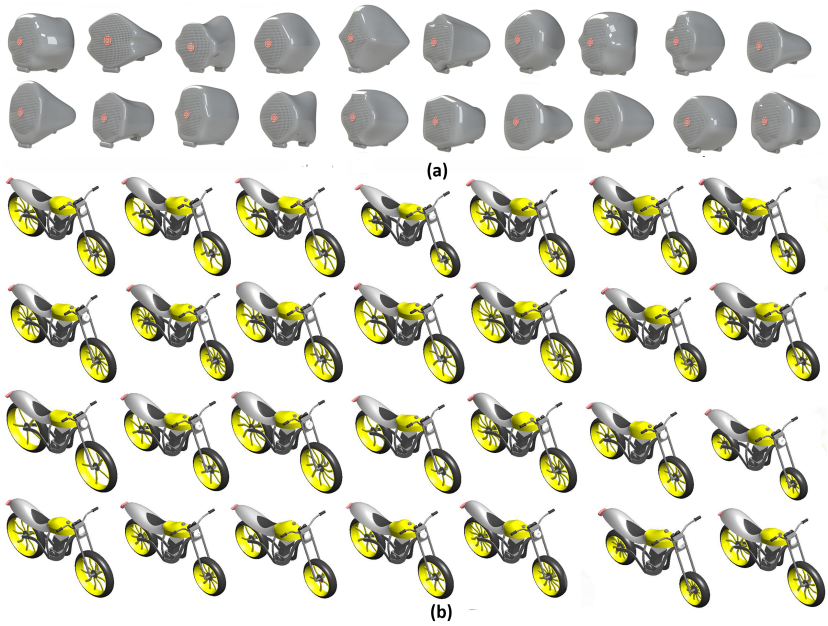

Figure 5: Design alternatives generated by Sf-GDT for (a) speaker and (b) motorbike models.

4.2.1. Sf-GDT With Discrete Parameters

A ceiling lamp model (see Figure 6 (a)) is used to demonstrate the performance of Sf-GDT for continuous and discrete parameters. The continuous parameter, y_1^L, y_2^L, and y_3^L, represents the vertical length of the lamp along the y-axis, and r_1^L and r_2^L are the radii of upper and lower circular region of the lamp. Where, discrete parameters, r_1^d and r_2^d, each containing five style forms ($t = 5$). These style forms, circle→1, square→2, ellipse→3, hexagon→4 and

Table 2: Algorithm setting and the results obtained from Sf-GDT for CAD models utilized for experimentation.

Designs	N	n	$U_1(B)$	$U_2(B)$	Maximum value of $U_2(B)$	CT (minutes)	i
Figure 5 (a)	20	22	46.43	10	4180	3.72	500
Figure 5 (b)	20	7	151.97	60	1330	1.47	300
Figure 7 (b)	20	10	109.79	0	1900	1.92	300
Figure 7 (c)	20	10	107.94	2	1900	2.10	300
Figure 7 (d)	40	10	487.02	2	7800	21.74	1500

octagon→5, are defined on the profile-1 (P1) and the profile-2 (P2) of the lamp model. Figure 6 (b) shows the design alternatives generated by Sf-GDT based on both discrete and continuous parameters. The ranges of the continuous design parameters of ceiling lamp model are given in Table 1. It was observed that the designs with both continuous and discrete parameters have more variation compared to the designs with only continuous parameters.

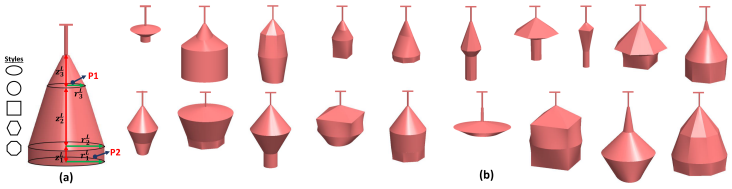

Figure 6: (a) Parametric CAD model of ceiling lamp. (b) Design alternatives of ceiling lamp model generated using Sf-GDT with continuous and discrete parameters.

4.2.2. Sf-GDT in Constrained Design Spaces

Sf-GDT can generate a variety of designs for a given model in the constrained and unconstrained design spaces. Both the design specifications and user preferences can be represented by constraints. To validate the performance of Sf-GDT, design specification such as the capacity of a wine glass to store a certain amount of wine, was given as a geometric constraint. The parametric representation of the wine glass model is shown in Figure 7 (a). 10 design parameters ($n = 10$) are used to represent this model. The design parameter y_0^G is the vertical length of glass stem and the design parameter x_1^G, x_2^G, y_1^G, x_3^G, y_2^G, x_4^G, y_3^G, x_5^G and y_4^G represent the 2D position of the control points of spline curve used to create the profile of the glass.

The glass design alternatives in Figure 7 (b) and (c) can store less than or equal to 200 (≤ 200) and greater than or equal to 700 (≥ 700) milliliter (ml) of wine, respectively. No design in Figure 7 (b) and (c) have violated these geometric constraints.

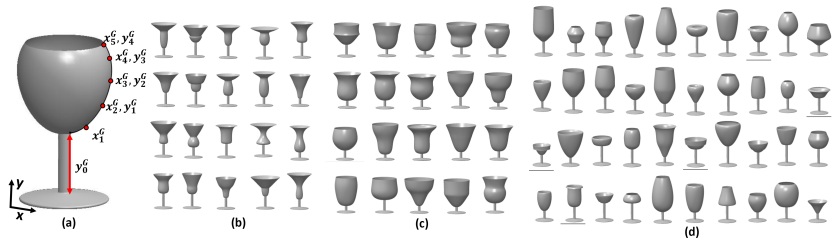

Figure 7: (a) Parametric representation of a wine glass model. Design alternatives of glass model generated by Sf-GDT in constrained space with (b) constraint-1 (c) constraint-2. (d) Design alternatives of glass model generated by utilizing Sf-GDT in an autonomously formed design space.

4.2.3. Performance of Sf-GDT in Different Design Space Formulations

The performance of Sf-GDT is also validated under different design space formulation (i.e. explicit, autonomous, and interactive) for the wine glass. The wine glass designs in Figure 7 (d) are generated by Sf-GDT in an autonomously formed design space with 50% extension of initial design. It can be observed from Figure 7 (d) that the underlined designs are implausible. These designs may not be feasible as a final market product. As mentioned before, one way to overcome this issue is to define geometric constraints.

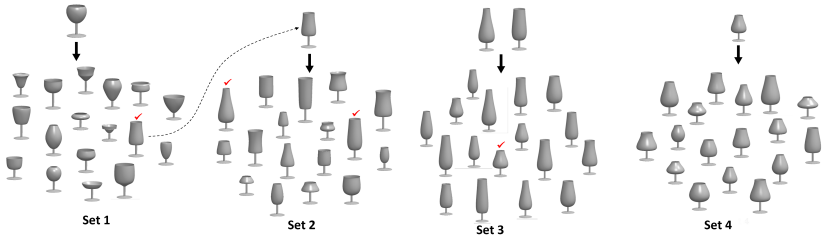

Figure 8: Design alternatives generated for the wine glass model in an interactively formalized design space.

In interactive space, an initial envelope can be set up either explicitly or autonomously. For example, Figure 8 demonstrate the interactive formulation of designs. In Figure 8, set-1 contains 17 design alternatives for the wine glass model. These designs were generated in a space that was created with a 50%

autonomous extension of the initial design. This set contains both plausible and implausible designs. From this set, one design was selected (checked in red). In the next step, this design was considered as new input model and a new set (set 2) of designs were created again with 50% autonomous extension of space around the new design. Note that set 2 contains all the plausible designs. From this set, two designs were select and new space was formed around the centroid of these two designs with 30% extension. Afterward, again 30% extension was done for the creation of designs in set-4 and from this final design was selected and the interactive process was stopped.

4.3. Computational Time (CT)

A PC having an Intel Core i7-5500 CPU, 2.4 GHz processor and 16 GB memory was used for the experiments in this study, and C++ programming language along with Siemens' Parasolid APIs were utilized for implementation and testing of Sf-GDT. We measured the CT taken to obtain results in Figure 5, 6 and 7, which is shown in Table 2; it varied between 1.47 and 21.74 minutes. The study on the effects of these parameters on CT is important in order to effectively utilize Sf-GDT. In the proposed approach, CT mainly depends on the number of designs to be generated (N), the dimensionality of the design space (n) and the size of the subpopulations (s). Increase in the values of these parameters will increase Sf-GDT's processing time. As the values of either N or s increases CT for Sf-GDT to create designs increases.

4.4. Parameter Tuning

For the experiments in this study, α was set equal to 10 except for the motorbike model for which $\alpha = 20$ was utilized because of the high number of design parameter ($n = 42$). We recommend the users to set an initial value of α equal to $n/2$, which can be altered later depending on the users' intention to create complete or semi-non-collapsing designs.

It is noteworthy that the value of $U_2(B)$ can be high for the problems with discrete parameters compared to the same problem with discrete parameters.

Instead of dividing the discrete parameter(s) into N intervals, we divide these parameters divided into t intervals, where t is the number of style profiles. If t is less than N, then, the number of collapsing designs will increase, which will result in a high value of $U_2(B)$. In order to have a low number of collapsing designs, t should be greater than or equal to N ($t \geq N$). However, $t < N$ does not affect the space-filling quality of the designs.

The size of the subpopulations (s) also plays an important role in the generation of space-filling designs. High values of s create diverse initial solutions for Sf-GDT, which facilitates its search for the global optimum solutions. In contrast, the application of Sf-GDT with the high values of s can result in a higher CT. We recommend setting s to a value higher than n. For the experiments in the current study, s was set equal to 15 except for the designs in Figure 5 (b). For that $s = 23$ was selected.

4.5. Convergence of Sf-GDT

The quality of any optimization technique mainly depends on its ability to provide an optimum solution or a solution close to the global optimum. The global optimum is a point in search space where the best solution(s) exists. As the Sf-GDT is based on the optimization technique, therefore, in order to verify the convergence ability to a global optimum its performance is observed against the number of iterations i it performs. The convergence ability of Sf-GDT is analyzed on different test models shown in Figure 5. Sf-GDT stops the optimization process when there is no improvement in the cost function $U(B)$ for some consecutive iterations (i); at this point, the designs being created reach the optimal position, and the algorithm is considered to converge to its optimality. Figure 9 shows the plot for $U(B)$ versus i for the designs in Figure 5, 6 and 7. A large number of iterations were performed for these models to analyze the convergence of Sf-GDT. No improvements were observed in $U(B)$ after some consecutive iterations. For the designs in Figure 6 (b), 7 (b) and 7 (c) there was no improvement occurred after approximately 300^{th} iteration and for the designs in Figure 5 (a), (b) and 7 (d) Sf-GDT converged at 500, 1200 and 1500

number of iterations, respectively. The convergence rate of Sf-GDT depends on the number of designs (N), the dimensionality of the design space (n), and the total number of geometric constraints.

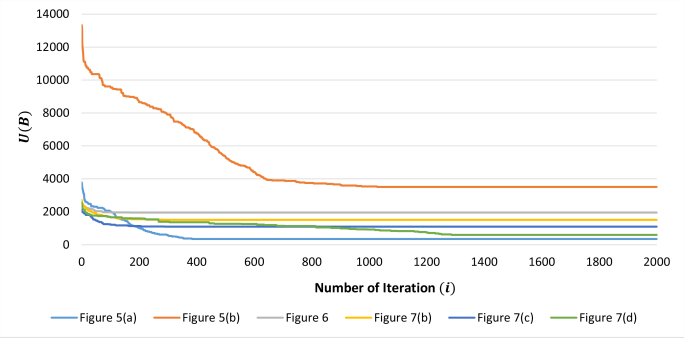

Figure 9: Plot showing the cost values versus number of A-GDT iterations for the models in Figure 5 (a), (b), Figure 6, 7 (b), (c) and (d)

4.6. Comparison with Existing Works

We compared the performance of Sf-GDT with the existing state-of-the-art techniques in the literature that have been proposed for generative and space-filling designs. First, we compared the performance of Sf-GDT with Krish's GDT [12]. Figure 10 (a) and (b) shows the design points representing designs generated by Krish's technique for the speaker model in Figure 4 (a) in a 2D design space. The designs in the 2D space give a better perspective to readers on how designs generated by [12] are spread in the design space. As mentioned in section 2, Krish utilized a threshold value, ranging from 0.0 to 1.0, to create dissimilar designs. The designs in Figure 10 (a) and (b) are created with threshold values of 0.5 and 1.0, respectively. It can be seen from the Figure 10 (a) and (b) that the design points ($N = 30$) are not uniformly distributed in the design space especially when the threshold value is 1.0. The designs in Figure 10 (a) and (b) have space-filling of 7149.9 and 44015, respectively. In case of threshold equal to 1.0 designs are clustered at the two corners of the design space and approximately more than 90% of space is left empty. In this case, designs are

also generated by Sf-GDT, which are shown in Figure 10 (c). This gives a comparative view to the readers on how Sf-GDT produces design alternatives for the same CAD model in 2D space. Note that the designs generated by Sf-GDT had space-filling of 2492.29, which is less than the designs generated by [12].

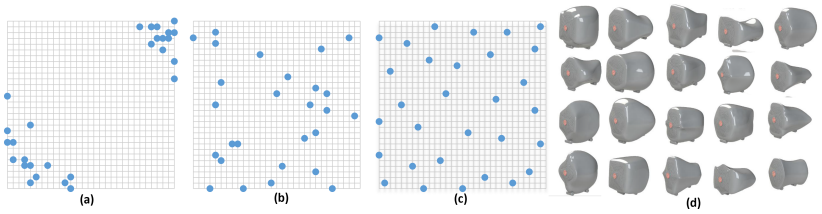

Figure 10: Design points created in 2D space by utilizing the technique of [12] with threshold values of (a) 0.5 and (b) 1.0. (c) Design points created using Sf-GDT. (d) Design alternative for speaker model created by utilizing the technique of [12].

4.6.1. User Study

Figure 10 (d) shows the designs created by Krish's technique [12] for the speaker model in Figure 4 (a) within a 22-dimensional design space, which were created to visually compare the results of Krish's technique and Sf-GDT. For this visual comparison, a user study was conducted to obtain the human perception about the quality of designs generated from the two techniques. This user study included 12 participants to compare the designs in Figure 10 (d) and 5 (a), which are obtained using Krish's technique and Sf-GDT, respectively. Six participants had more than two years of design experience in product development, and others were selected from the Amazon Mechanical Turk platform. The participants were asked to rate each design in Figure 10 (d) and 5 (a) based on a Likert scale, with anchors ranging from "very poor" to "very good" (1: very poor, 2: very good, 3: fair, 4: good, 5: very good). The participants involved in the study had not any information about the techniques used to generate these designs. This was done to minimize the possibility of a bias decision during design rating. In this study different set of rules was applied to the participants to ensure the reliability of the obtained results. The designs in Figure 10 (d) and 5 (a) were shuffled randomly and presented in two surveys, each with 25

designs. There was a repetition of five designs in each survey. For any participant, if there was no consistency in the ratings given to the designs and survey was completed in less than five minutes, then that participant's results were excluded from the study. Note that the design space utilized for the generation of alternatives in 10 (d) and 5 (a) was same.

Table 3 summarizes the user study's results. From the table, it can be observed that the average rating given by the participants to the designs generated with Krish's technique is lower than those obtained using Sf-GDT. Ten out of 12 participants preferred the designs generated using Sf-GDT, including the experience designers.

Table 3: Results of the user study

	Average Grade											
User	1	2	3	4	5	6	7	8	9	10	11	12
Sf-GDT	4.55	2.70	3.60	3.25	3.50	3.75	4.00	4.15	4.15	3.10	4.10	3.90
Krish [12]	2.10	3.20	3.65	2.25	3.10	2.45	2.95	4.10	3.75	2.55	2.80	2.75

	Space-filling	σ	μ	Skewness	p-value
Sf-GDT	46.430	0.630	3.020	0.12	0.00718
Krish [12]	55.039	0.520	3.730	-0.40	

A t-test was utilized to statistically examine the results of the user study. The data obtained from the user study were normally distributed, as the skewness value was close to zero and their mean values were approximately equal. The null hypothesis states that there is no significant difference between the ratings given to the designs generated using Krish's technique and Sf-GDT. The p-value obtained from the t-test is less than the significance level of 0.05, this indicates a stronger evidence against the null hypothesis.

From the results of the user study and statistical test, it can also be concluded that Sf-GDT outperforms the Krish's technique in term of creating appealing design alternatives for the users.

5. Usage of Sf-GDT with existing CAD software

Sf-GDT can be easily utilized with existing CAD software having parametric modeling functionality and can create a design table in the form of a spreadsheet such as Microsoft XL. CAD software such as SolidWorks has the ability to create and read external XL based design tables. A user interface called *DesignN* is developed to integrate Sf-GDT with such CAD software, which is shown in Figure 11. The design parameters of a model can be stored in the design table using build-in CAD functions. These parameter values can be given as input to DesignN to create design alternatives and their parameter values can be stored in a CSV file. These parameter values can be transferred to the design table that can then read by the CAD software to create designs. Data in the design table can also be structured in other formats required by the analytical software.

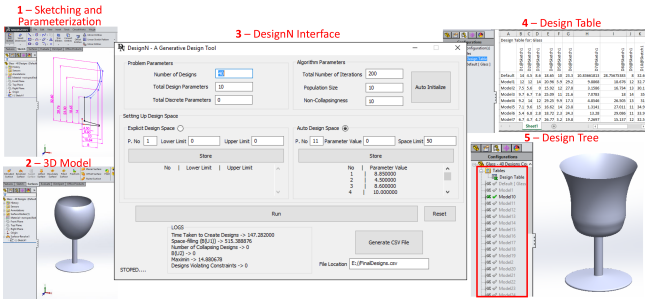

Figure 11: Window-based interface to integrate Sf-GDT with SoildWorks.

Figure 11 demonstrates the steps involved in the creation of designs for the wine glass model in SolidWorks via the window-based interface of DesignN. The wine glass model is first sketched and the design parameters are defined on this sketch. 3D surface model of the wine glass is created using Swept Surface feature of SolidWorks. Initial design parameter values are inputted to DesignN and designs are created in 50% autonomously formed design space. The parameter values of the generated designs are stored in the CSV file and are copied to the design table. SolidWorks read these parameter values and generate designs that are presented to the user within a Design Tree. Through this tree, each design can be visually inspected by the user for the final selection. A

web-based interface of DesignN is also developed, which can be accessed from https://geometric.brainlabsgp.com and a tutorial to use DesignN with SolidWorks can be found at https://youtu.be/QDcW2FPvq-Q.

6. Conclusions and Future Works

This paper proposes a state-of-the-art generative design technique for the automatic search and generation of design variations for a given CAD model based on its design specification. From these design alternatives, users can select a design(s) based on their aesthetic preference. Sf-GDT has the ability to generate designs in constrained and unconstrained design spaces. To obtain distinct and uniformly distributed designs in the design space, designs with space-filling and non-collapsing criteria are favored during the search process. To generate N optimal designs based on these criteria, Jaya algorithm is utilized and modified. Sf-GDT, first, randomly generate a subpopulation of solutions and improves these solutions using search approach of Jaya algorithm. Finally, Sf-GDT is compared with other existing techniques in the literature. The results of this paper show that Sf-GDT outperforms these techniques.

As a future work, we would like to integrate the users' preference and aesthetic judgment into Sf-GDT so that they can create designs based on their preference and aesthetic perception. The proposed technique will be extended for the generative creation of complex 3D character models. Finally, the performance of the different optimization techniques will be studied for this specific problem.

References

[1] Y.-C. Liu, A. Chakrabarti, T. Bligh, Towards an idealapproach for concept generation, Design Studies 24 (4) (2003) 341–355.

[2] J. Wang, Improved engineering design concept selection using fuzzy sets, International Journal of Computer Integrated Manufacturing 15 (1) (2002) 18–27.

[3] G. Pahl, W. Beitz, Engineering design: a systematic approach, Springer Science & Business Media, 2013.

[4] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, G. Fitzmaurice, Dreamsketch: Early stage 3d design explorations with sketching and generative design, in: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, ACM, 2017, pp. 401–414.

[5] N. Umetani, T. Igarashi, N. J. Mitra, Guided exploration of physically valid shapes for furniture design., ACM Trans. Graph. 31 (4) (2012) 86–1.

[6] K. Shea, R. Aish, M. Gourtovaia, Towards integrated performance-driven generative design tools, Automation in Construction 14 (2) (2005) 253–264.

[7] A. Nana, J.-C. Cuillière, V. Francois, Towards adaptive topology optimization, Advances in Engineering Software 100 (2016) 290–307.

[8] M. Turrin, P. von Buelow, R. Stouffs, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, Advanced Engineering Informatics 25 (4) (2011) 656–675.

[9] J. J. L. Kitchley, A. Srivathsan, Generative methods and the design process: A design tool for conceptual settlement planning, Applied Soft Computing 14 (2014) 634–652.

[10] L. Caldas, Generation of energy-efficient architecture solutions applying gene_arch: An evolution-based generative design system, Advanced Engineering Informatics 22 (1) (2008) 59–70.

[11] V. Granadeiro, J. P. Duarte, J. R. Correia, V. M. Leal, Building envelope shape design in early stages of the design process: Integrating architectural design systems and energy simulation, Automation in Construction 32 (2013) 196–209.

[12] S. Krish, A practical generative design method, Computer-Aided Design 43 (1) (2011) 88–100.

[13] Á. E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms, IEEE Transactions on evolutionary computation 3 (2) (1999) 124–141.

[14] R. V. Rao, G. Waghmare, A new optimization algorithm for solving complex constrained design optimization problems, Engineering Optimization 49 (1) (2017) 60–83.

[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, Ocean Engineering 136 (2017) 243–259.

[16] L. Wang, C. Huang, A novel elite opposition-based jaya algorithm for parameter estimation of photovoltaic cell models, Optik-International Journal for Light and Electron Optics 155 (2018) 351–356.

[17] P. D. Michailidis, An efficient multi-core implementation of the jaya optimisation algorithm, International Journal of Parallel, Emergent and Distributed Systems (2017) 1–33.

[18] R. V. Rao, A. Saroj, A self-adaptive multi-population based jaya algorithm for engineering optimization, Swarm and Evolutionary computation 37 (2017) 1–26.

[19] R. V. Rao, D. P. Rai, J. Balic, Multi-objective optimization of abrasive waterjet machining process using jaya algorithm and promethee method, Journal of Intelligent Manufacturing (2017) 1–27.

[20] R. R. Kurada, K. P. Kanadam, Automatic unsupervised data classification using jaya evolutionary algorithm, Adv. Comput. Intell.: Int. J 3 (2016) 35–42.

[21] S. P. Singh, T. Prakash, V. Singh, M. G. Babu, Analytic hierarchy process based automatic generation control of multi-area interconnected power sys-

tem using jaya algorithm, Engineering Applications of Artificial Intelligence 60 (2017) 35–44.

[22] R. Buddala, S. S. Mahapatra, Improved teaching–learning-based and jaya optimization algorithms for solving flexible flow shop scheduling problems, Journal of Industrial Engineering International (2017) 1–16.

[23] S. Degertekin, L. Lamberti, I. Ugur, Sizing, layout and topology design optimization of truss structures using the jaya algorithm, Applied Soft Computing.

[24] E. Gunpinar, S. Gunpinar, A shape sampling technique via particle tracing for cad models, Graphical Models 96 (2018) 11–29.

[25] S. Khan, E. Gunpinar, Sampling cad models via an extended teaching–learning-based optimization technique, Computer-Aided Design 100 (2018) 52–67.

[26] A. Runions, M. Fuhrer, B. Lane, P. Federl, A.-G. Rolland-Lagan, P. Prusinkiewicz, Modeling and visualization of leaf venation patterns, ACM Transactions on Graphics (TOG) 24 (3) (2005) 702–711.

[27] M. Ruiz-Montiel, J. Boned, J. Gavilanes, E. Jiménez, L. Mandow, J.-L. PéRez-De-La-Cruz, Design with shape grammars and reinforcement learning, Advanced Engineering Informatics 27 (2) (2013) 230–245.

[28] E. Kalogerakis, S. Chaudhuri, D. Koller, V. Koltun, A probabilistic model for component-based shape synthesis, ACM Transactions on Graphics (TOG) 31 (4) (2012) 55.

[29] P. Prusinkiewicz, M. Shirmohammadi, F. Samavati, L-systems in geometric modeling, International Journal of Foundations of Computer Science 23 (01) (2012) 133–146.

[30] M. C. A. H. H. CHAU, A. MCKAY, A. DE PENNINGTON, Combining evolutionary algorithms and shape grammars to generate branded product

design, in: Design Computing and Cognition06, Springer, 2006, pp. 521–539.

[31] J. Cui, M.-X. Tang, Integrating shape grammars into a generative system for zhuang ethnic embroidery design exploration, Computer-Aided Design 45 (3) (2013) 591–604.

[32] S. C. Chase, Generative design tools for novice designers: Issues for selection, Automation in Construction 14 (6) (2005) 689–698.

[33] G. Kelly, H. McCabe, Interactive generation of cities for real-time applications, in: ACM SIGGRAPH 2006 research posters, ACM, 2006, p. 44.

[34] W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, P. Prusinkiewicz, Self-organizing tree models for image synthesis, ACM Transactions on Graphics (TOG) 28 (3) (2009) 58.

[35] F. Fuerle, J. Sienz, Formulation of the audze–eglais uniform latin hypercube design of experiments for constrained design spaces, Advances in Engineering Software 42 (9) (2011) 680–689.

[36] D. Draguljić, T. J. Santner, A. M. Dean, Noncollapsing space-filling designs for bounded nonrectangular regions, Technometrics 54 (2) (2012) 169–178.

[37] M. W. Trosset, Approximate maximin distance designs, in: Proceedings of the Section on Physical and Engineering Sciences, 1999, pp. 223–227.

[38] E. Stinstra, D. den Hertog, P. Stehouwer, A. Vestjens, Constrained maximin designs for computer experiments, Technometrics 45 (4) (2003) 340–346.

[39] P. Audze, V. Eglais, New approach for planning out of experiments, Problems of dynamics and strengths 35 (1977) 104–107.

[40] V. R. Joseph, E. Gul, Maximum projection designs for computer experiments, Biometrika 102 (2) (2015) 371–380.

[41] J. D. Camba, M. Contero, P. Company, Parametric cad modeling: An analysis of strategies for design reusability, Computer-Aided Design 74 (2016) 18–31.