Applied Soft Computing Journal 89 (2020) 106121

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Generating *Kranok* patterns with an interactive evolutionary algorithm

Nutthanon Leelathakul, Sunisa Rimcharoen [*]

Faculty of Informatics, Burapha University, Thailand

ARTICLE INFO

Article history:
Received 19 March 2019
Received in revised form 6 December 2019
Accepted 15 January 2020
Available online 23 January 2020

Keywords:
Interactive evolutionary algorithm
Thai drawing
Kranok pattern
Bézier curve
Generated art

ABSTRACT

The *kranok* pattern is a classical Thai pattern, often seen in the ornamentation of Thai religious artifacts such as Tripiṭaka cabinets, temple doors and coffins. All Thai trainee artists must practice drawing the *kranok* pattern because it is a fundamental motif in Thai traditional decorative art. Individual skilled artists and instructors have their own preferred ways of drawing the pattern, and their styles differ in ornamental details. Nevertheless, all make use of similar basic structures. Most trainees learn a range of styles by observing experienced artists. After exploring the methods of drawing used by experts and published in textbooks, we designed an algorithm for automatically generating *kranok* patterns. We used an interactive evolutionary algorithm (IEA) to improve the aesthetic appeal of the generated patterns in response to users' feedback. The aim of our work was not to replace artists with machines, but to enhance human artistic expression. Specifically, the work aimed to help users without artistic skills to create *kranok* patterns in their own style. The algorithm facilitated the creation of a variety of personalized *kranok* patterns – diverse in their expressive curvature, refinement and proportions – that were satisfying to the varying preferences of a range of users. We also analyzed the proposed algorithm's behavior in terms of its convergence to generate specific shapes. The proposed method was examined by 28 respondents (27 Thai and 1 foreign) who were selected to include representatives of both sexes as well as experts in both Thai drawing and evolutionary algorithms. The results from our questionnaires showed that all respondents were satisfied with the generated *kranok* patterns: one respondent was 'completely satisfied', seventeen 'very satisfied', seven 'moderately satisfied', and three 'slightly satisfied'.

1. Introduction

The *kranok* pattern is a traditional motif in Thai drawing. It is one of the basic components of Thai decorative art, used in the decoration of artifacts, the ornamentation of shrine-hall gables, and the embroidery of the costumes used in classical Thai dance-drama (*khon*). Wenk [1], who studied the art of mother-of-pearl in Thailand, concluded that the *kranok* pattern is one of the masterpieces of Thai art. Typically, it is necessary to employ a skilled artist to draw *kranok* patterns because the achievement of aesthetically appealing results requires fluid and graceful delineation together with well-proportioned composition.

The exquisite intricacy of *kranok* patterns makes the task of generating them by a machine very challenging. A skilled human hand can intuitively negotiate the many underlying difficulties, but it is hard to produce results of comparable quality by automated processes alone. The particular difficulties of doing this with a *kranok* pattern lie in its use of intricate connecting curves. To construct a mathematical model for generating the patterns automatically, we gathered a collection of patterns from various artists, and then analyzed and extracted their common structures, together with the nuances of individual artists' styles. The results were used to define rules or equations that defined the pattern's components, e.g., curves, spires, notches, etc.

The weakness of such mathematical model is that it may produce patterns of limited aesthetic appeal, as such appeal depends on values that reside as much within the viewers' subjectivity. To overcome this limitation, we adapted an interactive evolutionary algorithm capable of evolving *kranok* patterns through multiple generations in response to feedback from individual human users. Our larger aim in this endeavor was to harness AI to augment conventional artistic expression of humans. Although AI technologies nowadays have huge impacts on human society, most AI applications have been used only for automating chores. The AI community has little knowledge of how to leverage the synergy of humans and AI, where humans could cooperate with AI to achieve beyond-human-level AI. By providing human input in the form of interpretation and verification, we hope our proposed work decreases this gap.

Evolutionary algorithms have been applied to a wide variety of creative arts for decades [2]. In traditional evolutionary algorithms [3], defining proper fitness functions is difficult because

[*] Corresponding author.
E-mail addresses: nutthanon@buu.ac.th (N. Leelathakul), rsunisa@buu.ac.th (S. Rimcharoen).

https://doi.org/10.1016/j.asoc.2020.106121
1568-4946/© 2020 Elsevier B.V. All rights reserved.

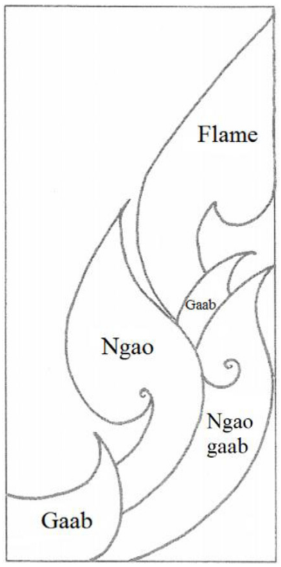


Fig. 1. The composition of the three-headed *kranok* pattern.

each fitness function has to reflect the relevant aesthetic, which is rather subjective and thus may vary with the audience. It seems more appropriate to engage the audience in the evolutionary process: hence, the invention of an IEA. During the IEA process, users participate by giving feedback to the algorithm, which gradually learns their preferences and thereby generates patterns more satisfactory to each individual user.

To the best of our knowledge, this is the first time an evolutionary approach has been used to create Thai patterns. We hope the proposed model will be a good starting point for the future synergy of humans and machines in the generation of personalized *kranok* patterns, which might be produced even by individuals without artistic skills. Such patterns could be further adapted for use as illustrations or decorations on a variety of products, increasing their variety, appeal and cultural impact.

The rest of this paper is organized as follows. Section 2 gives background information about the *kranok* pattern, Bézier curves and the IEA. Section 3 describes the proposed method. Section 4 reports the experiments on convergence. Section 5 shows the survey results and the respondents' feedback regarding the application in general. Section 6 presents the results of a survey concerning user fatigue. Section 7 provides discussions. Section 8 concludes our work and outlines our plan for future work.

2. Backgrounds

2.1. The kranok pattern

The *kranok* is a classical Thai pattern that mimics the shape of *flames*. Thai art students learn to draw this pattern in their very first lesson. It is a good basic exercise in Thai drawing because it is composed of multiple curvy lines [4]. In this paper, we attempt to use a machine to generate a three-headed *kranok* pattern. The main constituents of the three-headed *kranok* pattern (depicted in Fig. 1) are (1) *ngao*: this is the main part of the pattern, located in the middle left; (2) *ngao gaab:* this serves as a base located to the right of and slightly below the *ngao*, and (3) the *flame*: this is the top part of the *kranok* pattern, making the pattern more attractive. Further *gaab*s are often added for decorative effect.

In general, the drawing of the *kranok* pattern follows the steps below.

Step 1: Draw a 2 × 3 grid as shown in the top left of Fig. 2(a) or a 2 × 4 grid as shown in Fig. 2(b).
Step 2: Sketch the main structure of the *kranok* pattern. Two examples are shown in Fig. 3.
Step 3: Draw the three main components (i.e., *ngao*, *ngao gaab*, and *flame*) of the *kranok* pattern as shown in Fig. 4. Add 2 decorative *gaab*s: one at the bottom and the other between the *ngao gaab* and the *flame*. Then customize the details (i.e., add notches around the three main components) and possibly more curves.
Step 4: Add sub-components (such as sub-*ngao*, sub-*ngao gaab*, and sub-*flame*). Finalize the *kranok* pattern as shown in Fig. 5.

In this paper, we focus on generating the three main components of the *kranok* pattern, together with their decorative details (as in Step 3), but not on the sub-components shown in Step 4.

2.2. Bézier curve

Bézier curves are common in computer graphics. They have been widely used to draw or fit scalable smooth curves. For example, Chang and Yan [5] used pairwise cubic Bézier curves for fitting hand-drawn images. Pal et al. [6] approximated digitized curves based on Bézier curves. Masood and Sarfraz [7] used cubic Bézier curves to capture outlines of 2D objects. Han and Guo [8] approximated conic sections by using quadratic Bézier curves. Our work also uses cubic Bézier curves as a basis for generating the *kranok* patterns. Four fixed points are needed to manipulate the plotting of the cubic Bézier curve. Let P_0, P_1, P_2 and P_3 denote the first end point, two middle control points, and the last end point respectively. The curve $B(t)$ is represented by Eq. (1) (where $t \in [0, 1]$).

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3, \; 0 \le t \le 1 \quad (1)$$

Fig. 6 is an example of the cubic Bézier curve. Given P_0 and P_3 as the first and the last end point, as t is varied from 0 to 1, the curve bends toward the two middle points P_1 and P_2.

2.3. Interactive evolutionary algorithm

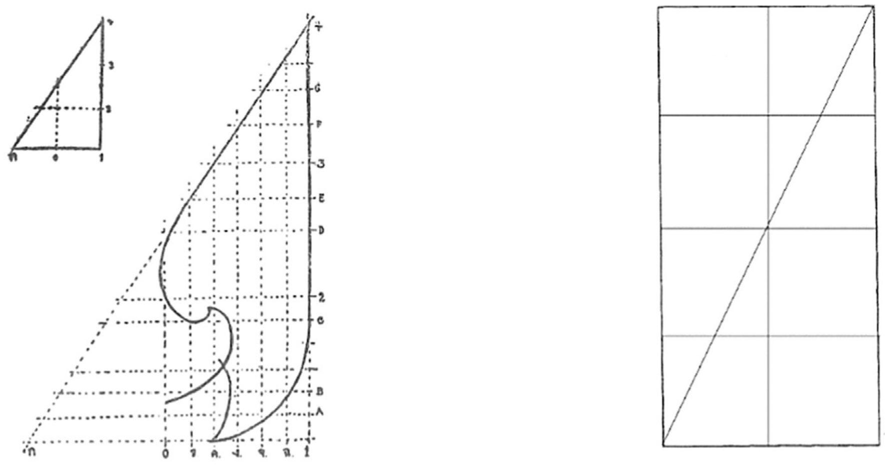
Interactive evolutionary algorithms (IEAs) are variations of evolutionary algorithms. They differ from traditional evolutionary algorithms in the fitness evaluation process. Instead of using a pre-defined fitness function, each of the IEAs lets users interact with the algorithm by giving feedback (as the fitness values) to each candidate solution, allowing the algorithm to determine the potential candidates specifically for each user. The IEAs are applied to various application domains where it is hard or almost impossible to define the proper fitness functions. Fig. 7 shows the process overview of the IEAs.

We review and categorize related works (which have applied the IEA to solve various problems) into two main domains: (1) design and art, and (2) user-satisfaction optimization.

(1) Interactive evolutionary algorithm for design and art

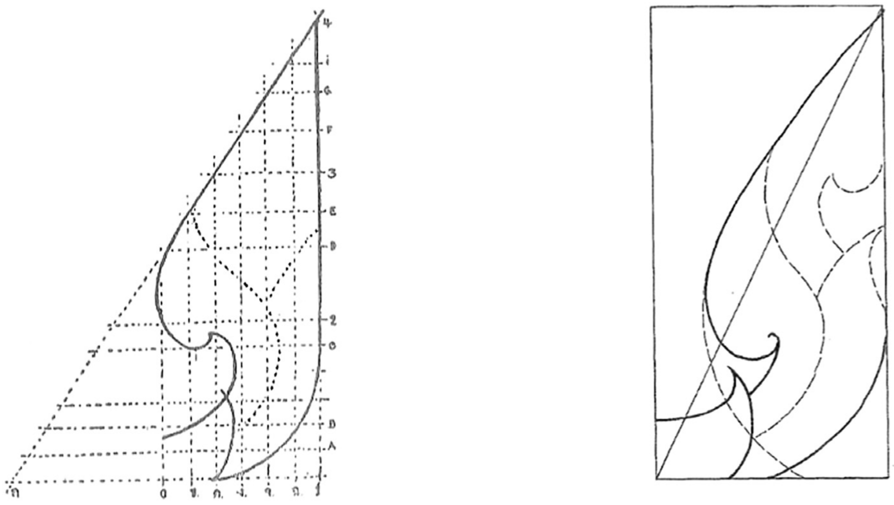
- Fashion design

Kim and Cho [9] created 3-D models with OpenGL and GLUT libraries for designing women's dresses (by varying bodies, necks, sleeves and skirts). The wide variety of dress styles (so called populations) were shown on a screen and users assigned fitness values to each pattern. The algorithm generated offspring using information from the given fitness value of each design. Gong et al. [10] proposed an interactive genetic algorithm with a multi-population adaptive hierarchy for designing women's dresses (by varying sleeves, necklines, and skirts). They reported that their proposed hierarchical structure helps decrease the search space and the convergence time. Mok et al. [11] used an interactive genetic algorithm for designing sketches of skirt silhouettes, waist



(a) *Kranok* proportions [45] (b) *Kranok* proportions [4]

Fig. 2. Grids for sketching the *kranok* pattern.



(a) Three-headed *kranok* draft [45] (b) Three-headed *kranok* draft [4]

Fig. 3. Sketching a structure of the *kranok* pattern.

styles, and other key style elements (such as darts, yokes, pleats, panels, and gathers). Sugahara et al. [12] applied an interactive genetic algorithm to help design the traditional Japanese garment called Yukata. The algorithm created a variety of colors and patterns to fit each user's preference.

- *Product and room design*

Brintrup et al. [13] proposed an interactive multi-objective genetic algorithm to help design ergonomic chairs. They considered both qualitative and quantitative objectives: "how suitable the chair looks for a particular posture" (from the designer's point of view) and "how closely the chair fits the sitter's posture". Bandte [14] used an IEA for aircraft design. Thirty-five variables were grouped into five main categories: vehicles, wings, fuselages, empennages (tail assemblies) and engines. Akase and Okada [15] presented a 3D room layout design by incorporating user interaction. Hernandez et al. [16] presented an interactive genetic algorithm for solving the unequal area facility layout problem. Dou et al. [17] proposed a multi-stage interactive genetic algorithm and applied it to car console design. It helped obtain users' requirements with the aim of improving the product configuration design and customization. Dou et al. [18] proposed an interactive genetic algorithm for customer-collaborative car console design.

- *Software and user-interface design*

Simons and Parmee [19] proposed an IEA to design object-oriented software. Monmarche et al. [20], Yokoyama et al. [21] and Sorn et al. [22] used interactive genetic algorithms to generate HTML pages for websites. Yoon et al. [23] focused on generating 3D game models and textures. They provided a web-based interactive genetic algorithm that allows users to change shapes and image textures. Martinez et al. [24] presented a method to estimate and predict user perceptions. They selected a generated artwork as a case study, and used Software Product Line (SPL) techniques to collect users' feedback. Martinez et al. [25] proposed an approach to the management of user-interface design. They employed an interactive genetic algorithm to reduce the number of involved end users in assessing user-interface variants.

- *Media design*

Miki et al. [26] made use of an interactive genetic algorithm that evolves melodies by taking into account the listener's comfort. The generated melodies could be used to warn users when specified events (e.g., a microwave finishing the food warming) occur — without annoying the listeners. Hastings et al. [27]

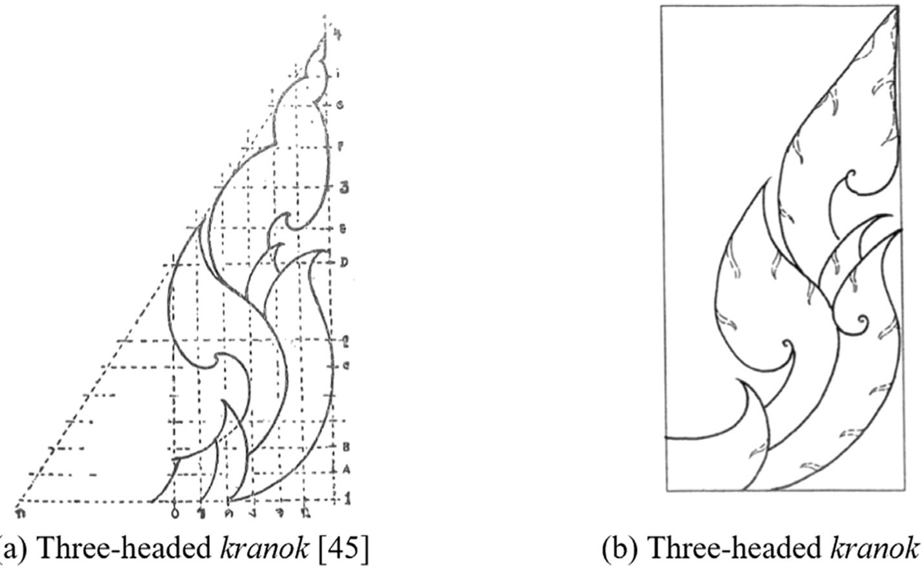


Fig. 4. Sketching the three-headed *kranok* pattern together with the ornamental details.

proposed a content-generating Neuro-evolution of Augmenting Topologies (cgNEAT) to generate online game content automatically. Cardamone et al. [28] proposed an IEA to generate tracks for a car-racing game. Yoshida et al. [29] presented an interactive genetic algorithm to help users (without any professional skills) generate fonts. Lv et al. [30] used an interactive genetic algorithm to synthesize emotionally expressive speech by adjusting prosody parameters: the resulting algorithm can synthesize speech expressing six emotions, namely joy, anger, fear, disgust, surprise, and grief.

- Arts

Eiben [31] used an IEA to produce images in the styles of the painters Mondrian and Escher. The images were displayed at the City Museum in The Hague, the Netherlands, and received very positive feedback. In his concluding remarks, Eiben mentioned two interesting points, namely that (1) the chromosome representation is one of the most important sources of creativity, and (2) there is a great challenge in using evolutionary processes to mimic existing artworks. Lv et al. [32] generated Batik-style patterns using an interactive genetic algorithm incorporated with a back-propagation neural network and fuzzy interval fitness. Hailemariam et al. [33] presented an interactive genetic algorithm to evolve facial animations on a 3D face model. They also used an elitism strategy to minimize user fatigue during the evolution.

(2) Interactive evolutionary algorithm for optimization of user satisfaction

- Marketing

Madera et al. [34] applied an IEA and fuzzy logic to optimize advertising texts and increase the effectiveness of advertising campaigns. They provided a web interface for each user to choose the most attractive texts of the current generation. The algorithm then evolved new advertising texts with more efficacy.

- Image

Arevalillo-Herráez et al. [35] proposed a content-based image retrieval technique to fetch pictures from large repositories. They incorporated an IEA and distance-based strategies within a relevant feedback algorithm. Solomon et al. [36] used an IEA to generate facial composites of suspects in criminal investigations. They developed the technique to help witnesses to produce facial likenesses of suspects. Mist and Gibson [37] used an IEA to optimize weighted vector directional filters for image processing. They allowed users to select a single member of the population to be seeded in the following generation. Their perceptual image quality outperforms that of previous techniques. Bergen [38] evolved stylized images (made up of primitive geometric forms so as to make them look as close as possible to the original images) using an interactive genetic algorithm. Users can control a variety of options and parameters, and are able to edit the chromosomes directly.

- Engineering

Ruiz et al. [39] proposed a preference-based evolutionary multiobjective optimization for improving the auxiliary services of power plants. Ismail et al. [40] proposed a new IEA for the vehicle routing problem. Users could visually identify interesting route segments by expressing two kinds of preferences: I like and I do not like. The candidate solutions preferred by each user have a higher chance of surviving in the next generation. Quiroz et al. [41] introduced a collaborative evolution for floor planning. They allowed a group of collaborative users to participate in the evolution of floorplans by reflecting the peer group's expertise and preferences.

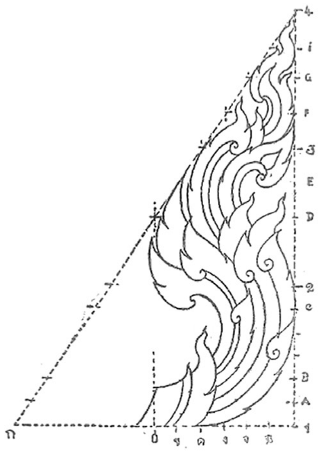
- Medical

Takagi and Ohsaki [42] presented a digital hearing aid fitting for hearing-impaired people. They applied an IEA to optimize hearing based on feedback from users. Lameijer et al. [43] created a tool called Molecule Evaluator to help chemists design a new drug. The tool evolved the new drugs by taking into account both the biochemical knowledge of molecular structures and the chemists' feedback.

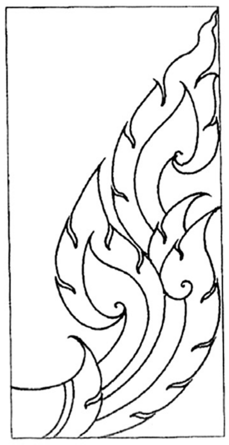
The researches mentioned above and the references in Takagi's survey [44] confirm the practicality of the IEA in various applications that demand human judgment during the evolutionary process. Thus, our work aims to adapt the IEA for creating and evolving *kranok* patterns by taking into account each user's satisfaction. The pattern selected as the focus of this paper is the three-headed *kranok* pattern, which is the archetype of other Thai patterns. Details of the *kranok*-pattern generation will be introduced in the next section.

3. Methodology

This section presents our methodology for generating the three-headed *kranok* patterns using the IEA. We explain in detail (a) how to encode our proposed chromosome, and (b) our proposed six-step IEA.



(a) Finalized three-headed *kranok* with details [45]



(b) Finalized three-headed *kranok* with details [4]

Fig. 5. Detailed decoration in the *kranok* pattern.

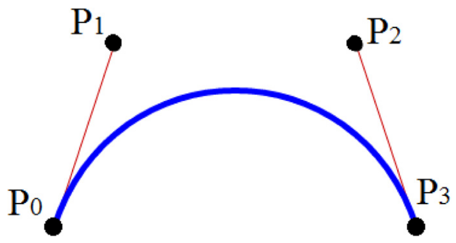


Fig. 6. Cubic Bézier curve.

3.1. Chromosome encoding

First, we encode the three-headed *kranok* pattern into a chromosome structure, which stores data for the evolutionary computation. During the computation process, new chromosomes are randomly generated using the structure. The values in the chromosomes are parameters for drawing new *kranok* patterns, which then are evaluated by users. The way in which various aspects of *kranok* patterns (i.e., curves, positions, etc.) are encoded into a chromosome is as follows.

To design the chromosome structure, we studied various *kranok* patterns illustrated in two textbooks of Thai drawing [4,45], as well as consulting an expert in the field. Various characteristics (such as the number of curves, curvature, component positions, and component proportions) were analyzed. Our proposed model for representing the *kranok* patterns consisted of 13 Bézier curves, illustrated in Fig. 8. Fig. 8(a) and (b) show the five components of the *kranok* pattern within a predefined boundary. Fig. 8 (c) assigns a number to each of the curves, each of which can be represented by a cubic Bézier curve.

Each curve can be ornamented with a few edge-cuts (hereafter called notches), as shown in Fig. 9(a). Generally, the notches are cuts at equal intervals along a curve. There are two notch types: inward and outward. The inward notches are cuts whose arcs lie within the overall outline of the curve, while the outward notches' obtrude from that outline. The inward and outward notches are shown respectively on the left and on the right of Fig. 9(b). In this work, we encode the notch type as a parameter in our proposed chromosome, so as to generate both inward and outward notches.

Based on the pattern analysis above, we devised a chromosome structure as shown in Fig. 10. The chromosome was composed of 13 curve parameter sets, each of which included four Bézier points, the number of notches (ranging from 0 to 5), the notch type (inward or outward), and the curvature level (ranging from 60 to 80 degrees).

We set the canvas size at 282 $\times$ 324 pixels, and the coordinate pair (0, 0) was at the top-left corner of the canvas. The positions of all Bézier points were randomized within the ranges defined in Table 1. For example, the first point of curve$_0$ (denoted as $Curve_0.P_0(x, y)$ in Table 1) was randomly chosen within a range from 69 to 81 for the x-coordinate, and within a range from 219 to 231 for the y-coordinate.

3.2. Generating kranok patterns

This section describes our IEA methodology for generating a variety of *kranok* patterns. It comprised six steps: shape generation, shape-feedback collection, shape evolution, ornamental-detail generation, collection of feedback on ornamental detail, and ornamental-detail evolution. The process overview is shown in Fig. 11.

(1) Shape generation

The evolutionary process starts with generating the first generation of *kranok* patterns (without ornamental details) as a pool of candidate solutions (called the population). The shapes of the *kranok* patterns are generated by decoding the initial randomly created chromosomes containing the parameters of 13 Bézier curves. Each curve is generated, one after another, until 13 curves are produced, subject to the conditions that each curve must be connected with others, and that the curves must together form a shape similar to the one in Fig. 8(c). For example, P_0 of curve$_0$ ($Curve_0.P_0$) and of curve$_1$ ($Curve_1.P_0$) must be at the same location, $Curve_2.P_3$must be on curve$_0$, and $Curve_3.P_3$must be at the same place as $Curve_2.P_{0,}$ and so on.

When our developed application starts, it generates nine *kranok* patterns complying with the initial nine chromosomes, which are then displayed on the screen, as shown in Fig. 12. Because of each point's randomized position, the application can deliver various styles of patterns. For instance, their heights, degrees of angle, and curvatures might be different. Underneath each generated pattern, the application provides feedback options for users to specify their preference interactively, as detailed in the next step.

(2) Shape-feedback collection

After a pool of candidate solutions is generated, the user evaluates all of them so that the process can learn which styles

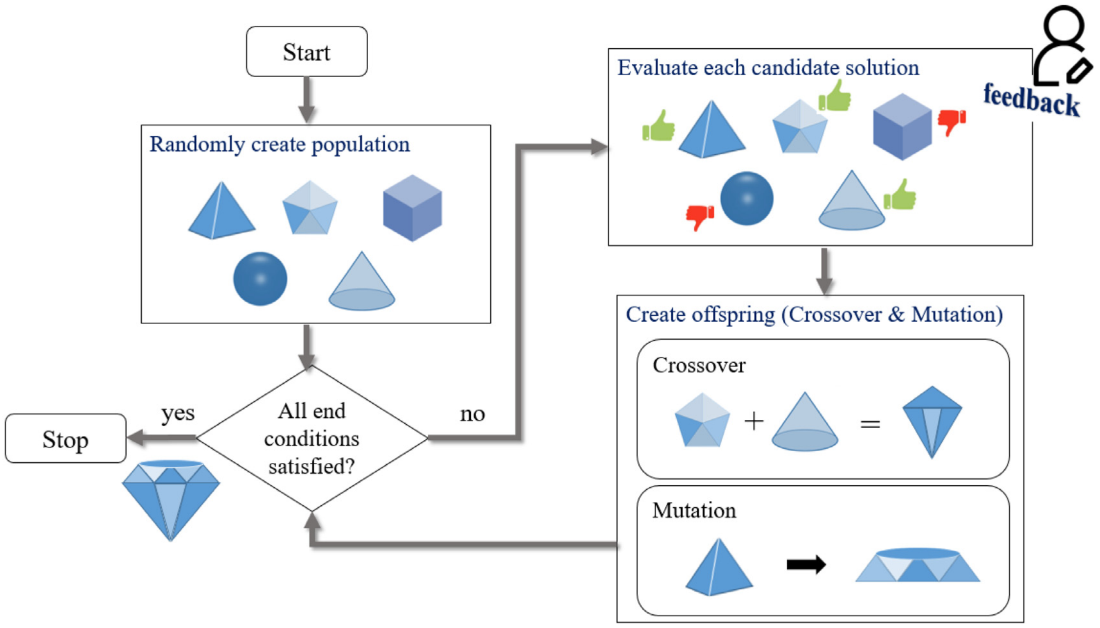

Start
Randomly create population
Evaluate each candidate solution
feedback
Create offspring (Crossover & Mutation)
Crossover
+
=
Mutation
All end conditions satisfied?
yes
no
Stop

Fig. 7. Interactive evolutionary algorithm.

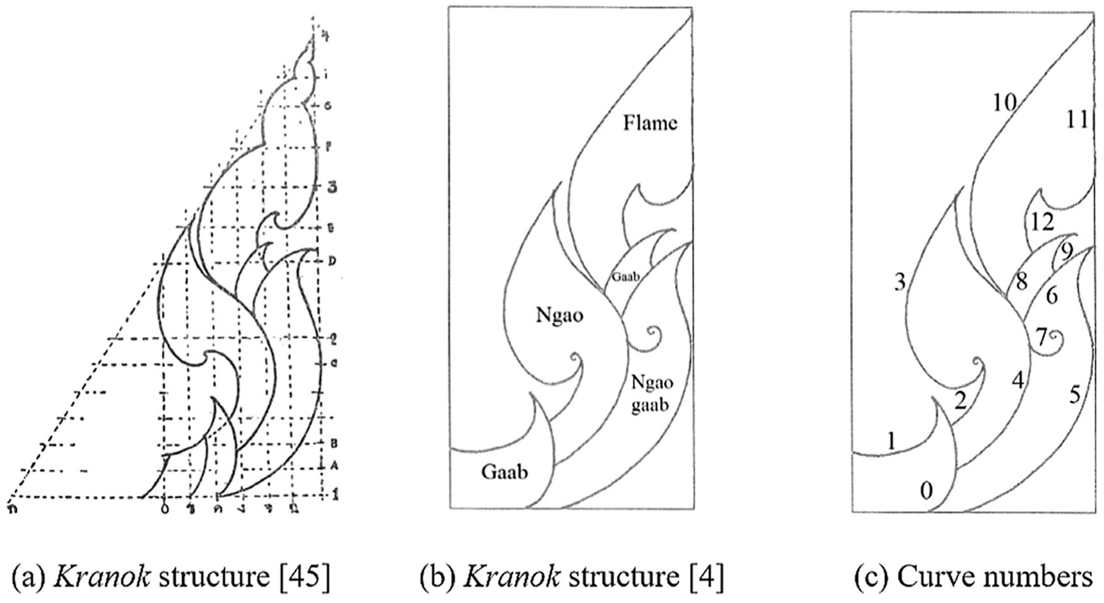


Fig. 8. Structure of the *kranok* pattern: (a) the original *kranok* pattern from [45], (b) the one from [4], (c) assignment of identifying numbers to component curves of the *kranok* pattern.

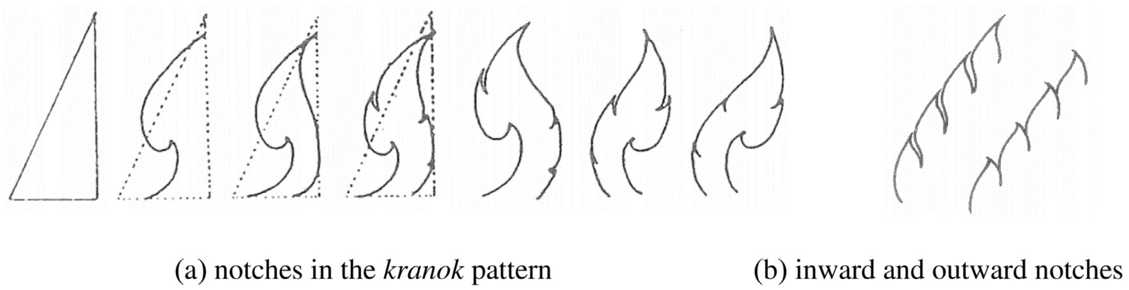

(a) notches in the kranok pattern
(b) inward and outward notches

Fig. 9. Examples of notches in the *kranok* pattern [46].

the user prefers. The IEA plays an important role in allowing users to input their aesthetic judgment of the *kranok* patterns.

There are two buttons below each pattern: *Rand* and *SaveImg* buttons. If the user does not like a specific generated *kranok*

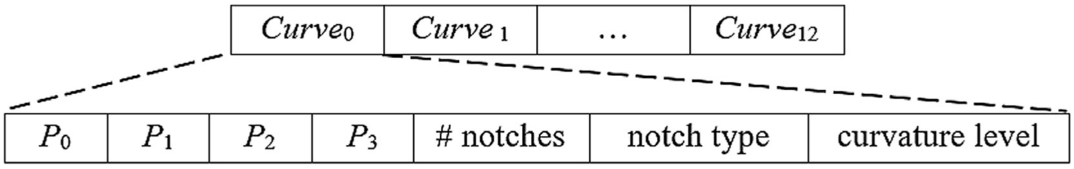


Fig. 10. Chromosome structure.

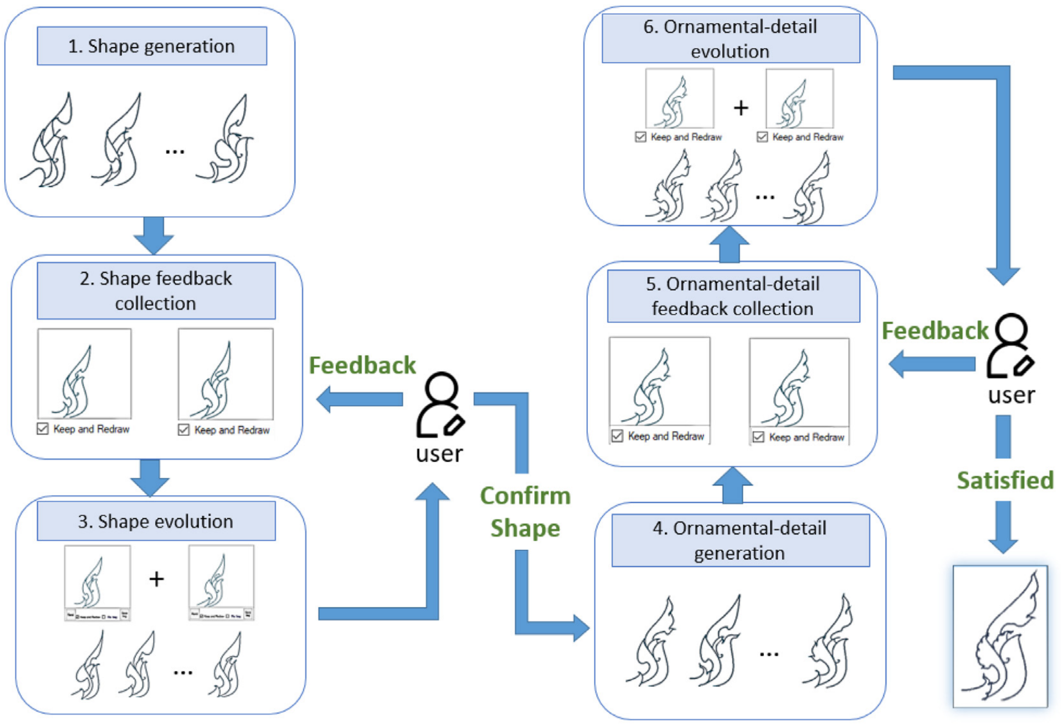


Fig. 11. The process for generating *kranok* patterns.

Table 1
Coordinate constraints for generating *kranok* patterns.

	$P_0(x, y)$		$P_1(x, y)$		$P_2(x, y)$		$P_3(x, y)$	
	x	y	x	y	x	y	x	y
$Curve_0$	75 ± 6.0	225 ± 6.0	$[Curve_0.P_0 - 6, Curve_0.P_0 + 18]$	225 ± 6.0	93 ± 6.0	250 ± 6.0	70 ± 6.0	267 ± 6.0
$Curve_1$	$Curve_0.P_0$	$Curve_0.P_0$	87 ± 6.0	225 ± 6.0	75 ± 6.0	250 ± 6.0	37 ± 6.0	250 ± 6.0
$Curve_2$	90 ± 6.0	202 ± 6.0	105 ± 6.0	210 ± 6.0	110 ± 6.0	232 ± 6.0	$Curve_0$	$Curve_0$
$Curve_3$	87 ± 6.0	147 ± 6.0	50 ± 6.0	180 ± 6.0	77 ± 6.0	222 ± 6.0	$Curve_2.P_0$	$Curve_2.P_0$
$Curve_4$	$Curve_3.P_0$	$Curve_3.P_0$	72 ± 6.0	162 ± 6.0	177 ± 6.0	227 ± 6.0	$Curve_0$	$Curve_0$
$Curve_5$	147 ± 6.0	147 ± 6.0	115 ± 6.0	152 ± 6.0	197 ± 6.0	252 ± 6.0	93 ± 6.0	267 ± 6.0
$Curve_6$	$Curve_5.P_0$	$Curve_3.P_0$	135 ± 6.0	142 ± 6.0	101 ± 6.0	169 ± 6.0	$Curve_4$	$Curve_4$
$Curve_7$	$[Curve_6.P_3+6.3, Curve_6.P_3+7.8]$	$[Curve_6.P_3+3.4, Curve_6.P_3+3.6]$	$[Curve_6.P_3+17, Curve_6.P_3+18.5]$	$[Curve_6.P_3+9.2, Curve_6.P_3+9.8]$	$[Curve_6.P_3+8.7, Curve_6.P_3+10]$	$[Curve_6.P_3+8.8, Curve_6.P_3+11.5]$	$Curve_6.P_3$	$Curve_6.P_3$
$Curve_8$	128 ± 6.0	143 ± 6.0	120 ± 6.0	138 ± 6.0	93 ± 6.0	162 ± 6.0	$Curve_4$	$Curve_4$
$Curve_9$	$Curve_8.P_0$	$Curve_8.P_0$	120 ± 6.0	143 ± 6.0	118 ± 6.0	150 ± 6.0	$Curve_6$	$Curve_6$
$Curve_{10}$	152 ± 12.0	45 ± 12.0	137 ± 6.0	100 ± 6.0	77 ± 6.0	125 ± 6.0	$Curve_4$	$Curve_4$
$Curve_{11}$	$Curve_{10}.P_0$	$Curve_{10}.P_0$	160 ± 6.0	160 ± 6.0	127 ± 6.0	130 ± 6.0	131 ± 6.0	125 ± 6.0
$Curve_{12}$	$Curve_{11}.P_3$	$Curve_{11}.P_3$	114 ± 6.0	128 ± 6.0	107 ± 6.0	140 ± 6.0	$Curve_8$	$Curve_8$

pattern, she clicks the *Rand* button. (The Rand button's pros and cons are discussed in Section 7.3.) Then the application randomly generates a totally new pattern that does not inherit any parts from the previous patterns (called predecessors). The user can export the image (in the png format) of the preferred pattern by clicking the *SaveImg* button. There are two checkboxes between the two buttons. If the user decides to keep the shape of a specific *kranok* pattern as a predecessor of the next generation, she checks the *Keep and Redraw* or *Fix Img* checkbox. The *Keep and Redraw* checkbox is checked when the user is satisfied with only some parts of the generated pattern, and chooses to replace it with a new but similar one. The *Keep and Redraw* option plays an important role in the evolutionary selection process as it enables the selected patterns to be propagated to the next generation. The *Fix Img* checkbox is checked when the user is satisfied with the whole pattern, and decides to keep it. This is in accordance with the elitism concept in the evolutionary approach as it preserves and places the chosen candidate solutions in the mating pool to

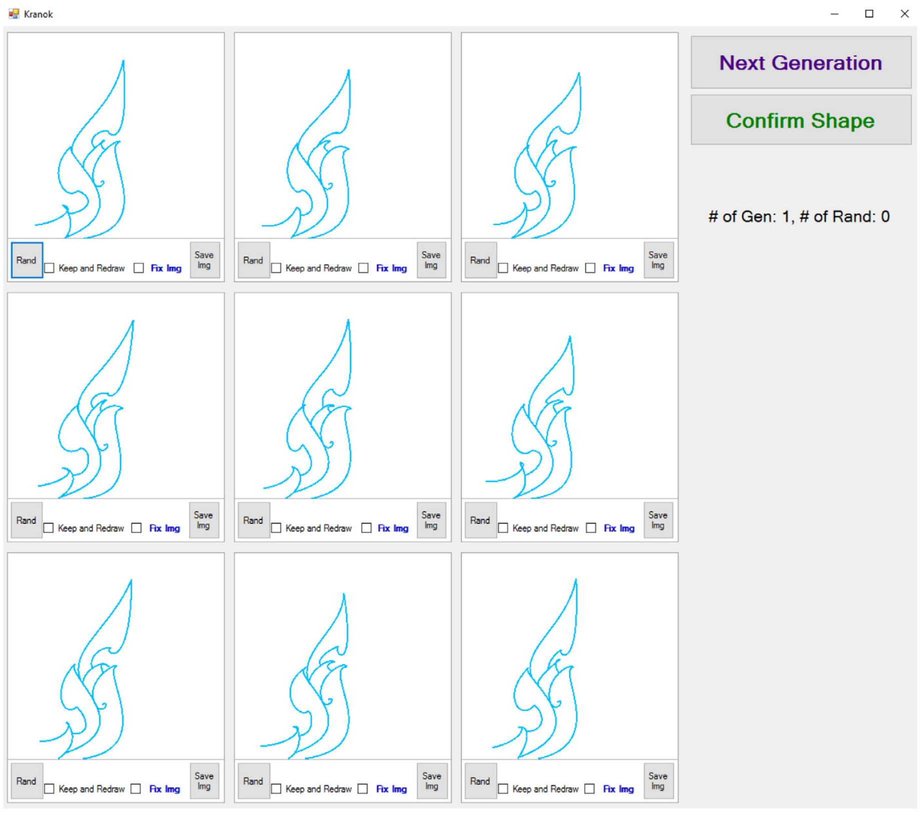


Fig. 12. The user interface of the application.

create the next generation. From each pool of candidate solutions a user can choose to keep as many patterns as she prefers (up to nine patterns). In the next generation, the algorithm replaces the previous patterns (except the one with the checked *Fix Img* checkbox) with newly generated ones, of which certain parts might be inherited from the chosen ones in the last generation.

(3) Shape evolution

The algorithm takes into account the user's feedback to generate the next generation of candidate solutions in the following manner.

- Each *kranok* pattern will be replaced with a new one if its corresponding *Fix Img* option is unchecked.

- Each new pattern (or new candidate) is created by combining arbitrary curves of a pair of previous *kranok* patterns for which the user has checked the corresponding *Keep and Redraw* and/or *Fix Img* options. (This method of generating new patterns is the so-called crossover.) In this paper, we adopt the uniform crossover method [47], which selects each of the curves either from the first or the second parent with equal probability, resulting in more diverse patterns. Then, each candidate may be mutated, with probability p, by replacing its randomly-selected curve with a new randomly-generated one. In our application, the crossover rate is 0.6, and the mutation rate is 0.3. Setting the crossover rate too low makes the new generation too similar to the previous one (i.e., no diversity), while setting it too high may lead to premature convergence. It is common to keep the mutation rate low. The higher the mutation rate, the longer the convergence time.

- When all of the candidates are generated, the algorithm displays them on the application's user interface and waits for more feedback.

The process repeats until the user presses the *Confirm Shape* button, indicating that she is now satisfied with the evolved overall shape, and is ready to proceed to the generation of ornamental details as the next step.

(4) Ornamental-detail generation

The ornamental details of the shapes are generated by following the steps below:

Step 4.1: Randomly select curves to be decorated with notches, and then determine the length (L) of each curve by summing the lengths of all sub-curves, each of which connects a pair of neighboring points on the curve. The lengths of all the sub-curves are approximated according to Eq. (2) [48] below:

$$Arc\ Length = \int_a^b \sqrt{1+\left(\frac{dy}{dx}\right)^2}\,dx \qquad (2)$$

Step 4.2: Randomize the number of notches (n) in each curve.

Step 4.3: Randomize a parameter (prm) from the range of [60, 80] to specify the level of each notch' curvature.

Step 4.4: Divide each curve of length L into n segments, each of which has a notch. P_1 and P_8 denote the first and the last points of each segment respectively (and also the first and the last of each notch). We finally draw each segment with a notch by conjoining three Bézier curves (P_1, P_2, P_3, P_4), (P_4, P_5, P_6, P_7), and (P_6, P_7, P_7, P_8), as depicted in Fig. 13.

To conjoin the three cubic Bézier curves and make a notch, we need to know the coordinates of all the eight points. As P_1 and P_8 are the first and the last points of the segment, we already know their exact coordinates. The coordinates of the other points (P_2 to P_7) can be determined according to Table 2. We start by calculating the angle θ. Let l_1 be the height of the red triangle (in Fig. 13), and l_2 be the Euclidean distance between the point P_1 and P_8. The angle θ is $\arccos(l_1/l_2)$.

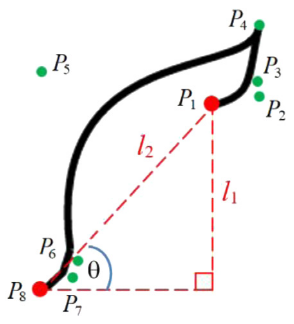


Fig. 13. One segment with a notch consisting of three *Bézier* curves.

Table 2
Control points for generating the ornamental details.

Control points	Equations
P_1	$x = x_1$ $y = y_1$
P_2	$x = x_1 + \cos(\theta + prm + 225) \times (L/6n)$ $y = y_1 + \sin(\theta + prm + 225) \times (L/6n)$
P_3	$x = x_1 + \cos(\theta + prm + 202.5) \times (L/6n)$ $y = y_1 + \sin(\theta + prm + 202.5) \times (L/6n)$
P_4	$x = x_1 + \cos(\theta + prm + 180) \times (L/3n)$ $y = y_1 + \sin(\theta + prm + 180) \times (L/3n)$
P_5	$x = x_1 + \cos(\theta + prm + 80) \times (L/1.5n)$ $y = y_1 + \sin(\theta + prm + 80) \times (L/1.5n)$
P_6	$x = x_8 + \cos(\theta + prm + 180) \times (L/6n)$ $y = y_8 + \sin(\theta + prm + 180) \times (L/6n)$
P_7	$x = x_8 + \cos(\theta + prm + 195) \times (L/12n)$ $y = y_8 + \sin(\theta + prm + 195) \times (L/12n)$
P_8	$x = x_8$ $y = y_8$

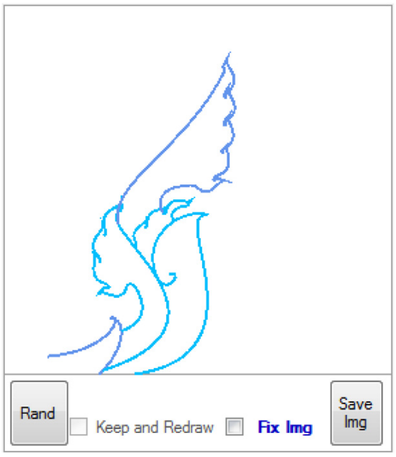


Fig. 14. Selecting the ornamental details.

Step 4.5: Repeat Step 4.1–4.4 on the remaining shapes.

(5) Ornamental-detail feedback collection

The nine patterns, now including ornamental details, are shown in the user interface and evaluated by the user. Each generated pattern contains 5 sub-bodies (*gaabs*). The user can double click on any *gaabs* to pinpoint which ornamental details she prefers. The example in Fig. 14 shows that, after the user selects the top and bottom *gaabs*, their edge color changes to darker blue. The user may keep as many *gaabs* (whose ornamental details are preferred) as she wishes.

(6) Ornamental-detail evolution

In the next generation, only ornamental details of the unselected *gaabs* are replaced with new ones, which are generated on the basis of those selected in the previous round. (The *gaabs*' shapes remain the same: only their ornamental pattern changes.) The user can repeatedly choose preferred patterns and click "Next Generation" to replace unsatisfactory patterns with newly generated ones until she is satisfied.

4. Experiments on convergence

As explained in the previous section, there are two main phases in the process of generating the *kranok* patterns: shape evolution and ornamental-detail evolution. In this section, we present experimental results and analyze the algorithm's behaviors in both phases. For all the experiments, we are the ones who gave the application the feedback according to the predefined objectives: users prefer (1) tall patterns, (2) wide patterns, (3) patterns with all *gaabs* notched, and (4) patterns with only one *gaab* (at the top) notched.

4.1. Phase 1: Evolving the shapes

We designed two scenarios to analyze the evolution behaviors of the algorithm. Specifically, we aimed to study how rapidly the IEA could learn each user's preferences, and generate satisfying patterns. In the first scenario, we assumed that users always preferred tall shapes, and in the second that they preferred wide shapes. We monitored the number of generations that had to be evolved before the desired shapes were generated (if at all).

Scenario I: Users prefer tall shapes

In this experiment, we simulated the scenario by running the algorithm five times, in each of which we selected certain generated shapes based on their height: the tallest shape(s) were kept as the parents of the shapes in the next generation. We recorded the heights of all patterns during the evolutionary process in order to monitor the algorithm's convergence. Each line in Fig. 15 shows the height averages of the shapes in all generations. The results of all five runs confirmed that the average heights gradually increased until the algorithm converged. The hypothesis test (one sample t-test) on the average convergence time is provided below.

H_0: mean convergence time (μ_c) is more than or equal to 8 generations

H_1: mean convergence time (μ_c) is less than 8 generations

The average convergence time (avg_c) was 6.0, the standard deviation (s_c) was 1.41, and the t-test statistic (t) $= -3.16$. With 95% confidence and the sample size (n) of 5, the critical value ($t_{0.05,4}$) $= -2.13$. The t-test statistic fell below the critical value. Therefore, we reject H_0 and can conclude with 95% confidence that the algorithm takes less than 8 generations to converge.

Scenario II: Users prefer wide shapes

In this experiment, the widest shape(s) were kept as the parent(s) for the shapes in the next generation. We recorded the widths of all the patterns in each generation. The average widths are shown in Fig. 16. All results obtained from the five runs in Fig. 16 confirm that the average widths of the patterns gradually increase until the algorithm converges. The hypothesis testing on the average convergence time is provided below.

$$H_0 : \mu\_c \geq 7$$

$$H_1 : \mu\_c < 7$$

For this experiment, where $n = 5$, $avg\_c = 6.0$, and $s\_c = 0.71$, the t-test statistic (t) is -3.16, falling below the critical value of -2.13. Therefore, we reject H_0 and can conclude with 95% confidence that the algorithm takes less than 7 generations to converge.

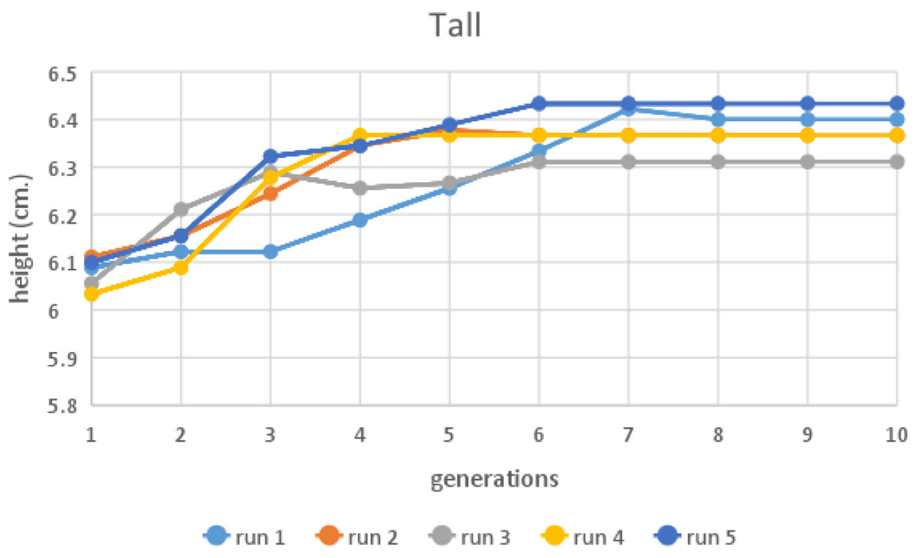


Fig. 15. Shape convergence in the scenario where tall shapes are preferred.

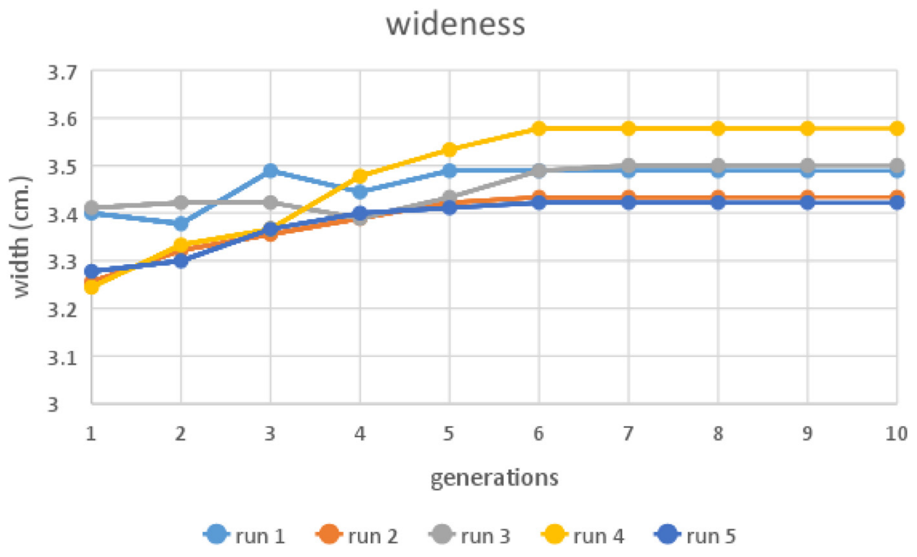


Fig. 16. Shape convergence in the scenario where wide shapes are preferred.

As shown in Figs. 15 and 16, the values of heights and widths grow and are saturated quickly after 6 generations because (1) all the canvases on the application interface are fixed in size, and (2) the small population size of 9 patterns allows fast convergence.

4.2. Phase 2: Evolving the ornamental details

As in Phase 1, we designed two scenarios to study how the proposed algorithm evolved the ornamental details. In the first scenario, we assumed users always preferred patterns in which all *gaabs* are notched. In the second scenario, we assumed users always preferred patterns with only one notched *gaab* that is located at the top of the pattern.

Scenario I: Users preferring patterns with all gaabs notched

In this experiment, we ran the algorithm five times and gave it feedback based on the number of *gaabs* to be notched. According to the assumption, the notched *gaabs* are kept to create children in successive generations. In each run, we recorded the number of the notched *gaabs* and the number of *kranok* patterns with all *gaabs* notched, with the aim of analyzing these records to understand the behavior of the algorithm. The graphs in Fig. 17 show (a) the number of notched *gaabs* and (b) the number of *kranok* patterns in which all *gaabs* are notched in each generation. The line plots show the convergence behavior in the five runs. The results obtained from all runs confirm that the number of notched *gaabs* gradually increases until the algorithm converges. The number of notched *gaabs* reached 45 (i.e., all five *gaabs* were notched in all nine patterns) by evolving only 4 generations. The hypothesis testing on the average convergence time is provided below.

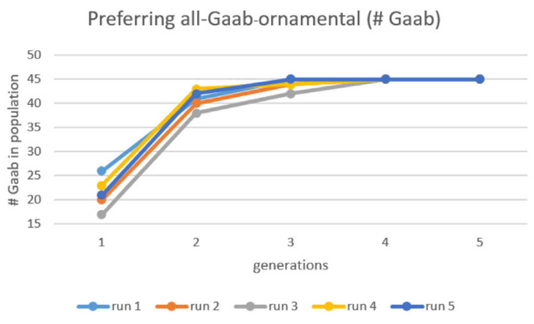
$H_0 : \mu\_c \geq 4$

$H_1 : \mu\_c < 5$

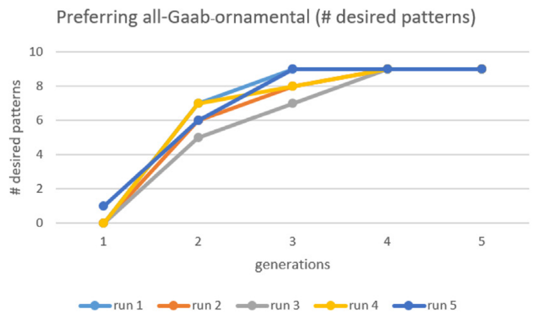
For these two experiments (a) and (b), where $n = 5$, $avg\_c = 3.6$, and $s\_c = 0.55$, the t-test statistic (t) is -5.72, falling below the critical value of -2.13. As a result, we reject H_0 and can conclude with 95% confidence that the algorithm takes less

Scenario II: Users preferring the pattern with only one gaab (at the top) being notched

In this experiment, we assumed that users preferred only one *gaab* being notched. Specifically, we assumed they desired only the top *gaab* to be notched. In each generation, the *gaabs*



(a)



(b)

Fig. 17. Convergence in the scenario where the shapes with all *gaabs* notched are preferred.

conforming to that assumption were kept as the parents of the next generation. We recorded (1) the number of shapes in which the top *gaab* (and the others, if any) was notched, and (2) the number of shapes, in which the only notched *gaab* was the one at the top. The former numbers are shown in Fig. 18(a) and the latter in Fig. 18(b). The results obtained from all five runs confirm that the algorithm converges to the shapes with the desired notched *gaab*s. The number of shapes with the top *gaab* notched increases to nine within 3 generations. Also, in Fig. 18(b), the number of shapes with only one notched *gaab* at the top converges to the maximum possible (i.e., (9) within 5 generations. The hypothesis tests on the average convergence time are provided below.

(a) Testing on the convergence time when users prefer the top *gaab* being notched (no matter whether the other *gaab*s are notched or not)

$H_0 : \mu\_c \geq 3$

$H_1 : \mu\_c < 3$

For this test, where $n = 5$, $avg\_c = 2.4$, and $s\_c = 0.55$, the t-test statistic (t) was -2.45, falling below the critical value of -2.13. Therefore, we reject H_0 and can conclude with 95% confidence that the algorithm takes less than 3 generations to converge.

(b) Testing on the convergence time when users prefer patterns in which only the top *gaab* was notched

$H_0 : \mu\_c \geq 5$

$H_1 : \mu\_c < 5$

For this test, where $n = 5$, $avg\_c = 3.8$, and $s\_c = 0.84$, the t-test statistic (t) is -3.21, falling below the critical value of -2.13. Therefore, we reject H_0 and can conclude with 95% confidence that the algorithm takes less than 5 generations to converge.

From the above studies on the algorithm's behaviors, it can be seen that the IEA quickly learns users' preferences without requiring them to put too much effort into pattern evaluation. These behaviors are different from those of traditional evolutionary algorithms, which need many more generations until they converge. These results confirm that the application is likely to generate desirable *kranok* patterns tailored for each user before she feels exhausted by the evaluation process. The study results on user fatigue are reported in Section 6.

Moreover, if users provide feedback with a consistent direction to their preference (as in Scenario I and II), the algorithm converges rapidly within 3–6 generations. This convergence rate conforms to the survey results detailed in the next section. The respondent who is an expert in Thai drawing used 6–10 generations to achieve his desired *kranok* shapes and ornamental details. We infer that the Thai-drawing expert had a clear target pattern in his mind (and evaluated the generated patterns on the basis of that target). This contrasted with the non-expert respondents, who (we infer) probably exhibited uncertainty, producing inconsistency in their choices, leading to a rather slower convergence.

5. Questionnaire survey on the application in general

5.1. Overview of the survey

The proposed method was implemented and made available online at http://staff.informatics.buu.ac.th/~rsunisa/kranok/. We collected feedback from 28 respondents who downloaded and tested the application. They included an expert in Thai art and an expert in evolutionary algorithms, together with 26 respondents with no relevant expertise. From the survey, most of the respondents had moderate skills as shown in Fig. 19.

After evaluating the application, the respondents answered the following survey questions:

- Q1: How satisfied are you with the generated *kranok* patterns?
- Q2: How much effort do you have to make to create a satisfying *kranok* pattern?
- Q3: How many generations (rounds) did you spend on attaining a satisfying *kranok* pattern?
- Q4: Overall, how would you rate the application?
- Q5: Do you have any other comments and suggestions?

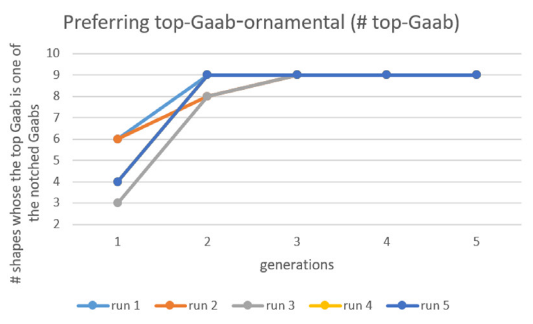
5.2. Results

Q.1: How satisfied are you with the generated kranok patterns?

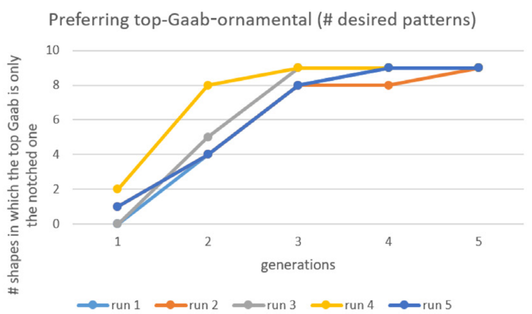
Out of the 28 respondents, 14 (51.9%) were very satisfied, 6 (22.2%) were neutral, 5 (18.5%) were unsatisfied, 1 (3.7%) was very satisfied, 1 (3.7%) was very unsatisfied, and 1 (3.7%) provided no answer to this question. According to the survey results shown in Fig. 20(a) and 20(b), the experts in Thai drawings and in evolutionary algorithms were very satisfied with the generated shapes, and the respondents unfamiliar with Thai drawings seemed somewhat satisfied. The five unsatisfied respondents were the ones who had seen Thai drawings but were amateurs in the execution of such drawings. The very unsatisfied respondent was the one who was unfamiliar with both Thai drawings and evolutionary algorithms.

Q.2: How much effort do you have to make to create a satisfying kranok pattern?

Of the 28 respondents, 12 (42.9%) expended a lot of effort, 7 (25%) some effort, 6 (21.4%) moderate effort, and 3 (10.7%) zero effort, as illustrated in Fig. 21. From the results, we found that the respondents who expended a lot of effort on creating



(a)



(b)

Fig. 18. Convergence in the scenario where shapes with only one notched *gaab* (at the top) are preferred.

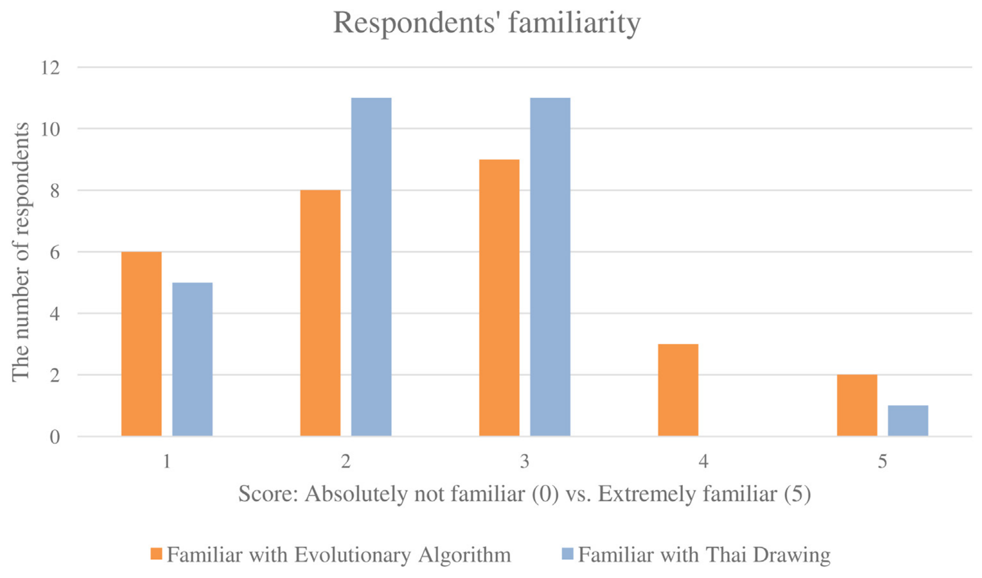
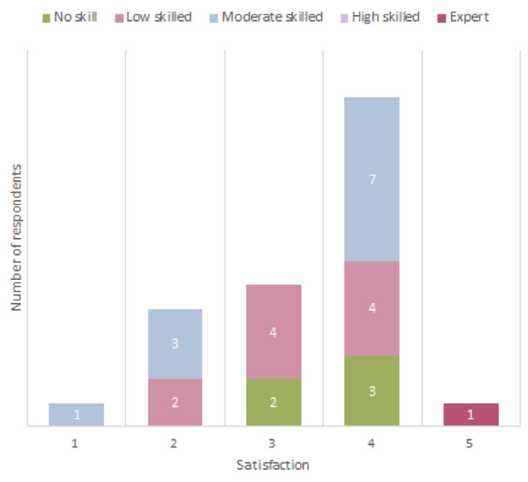
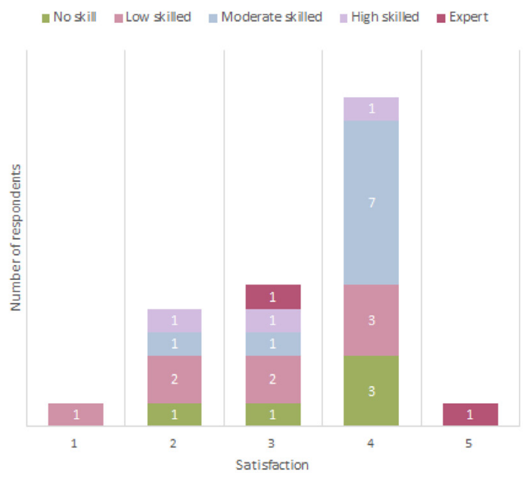


Fig. 19. Respondents' familiarity.

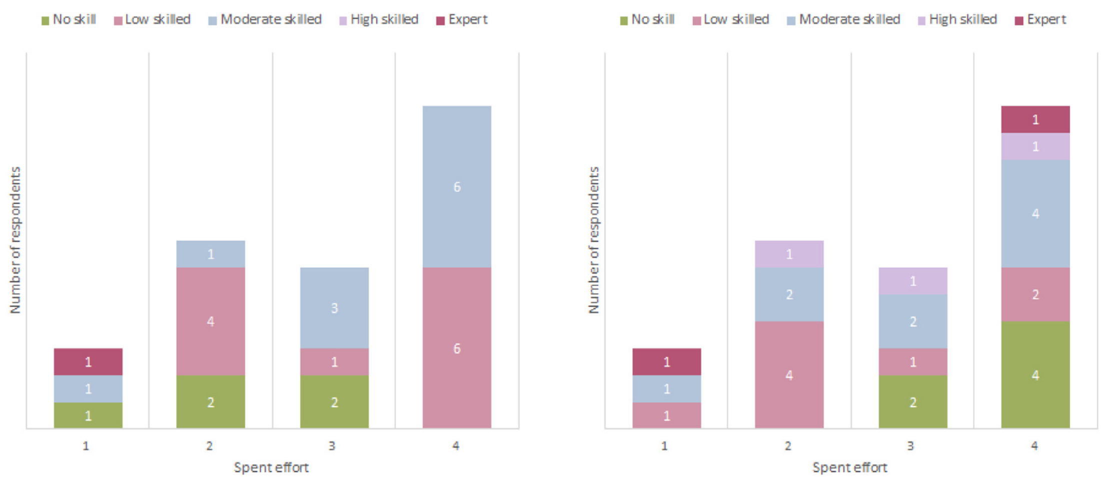


(a) Thai-drawing expertise



(b) Evolutionary algorithm expertise

Fig. 20. Respondents' satisfaction.



(a) Thai-drawing expertise (b) Evolutionary algorithm expertise

Fig. 21. Respondents' satisfaction.

satisfying shapes were the ones who were amateurs in Thai drawings (regardless of how familiar they were with evolutionary algorithms). (It should be noted that, by the term "effort" in this question, we made it clear to all the respondents that we meant usability.)

Q.3: How many generations (rounds) did you spend on attaining a satisfying kranok pattern?

Of the 28 respondents, 14 (50%) attempted more than 15 generations, 10 (35.7%) needed 6–10 generations, 3 (10.7%) used less than or equal to 5 generations, and 1 (3.6%) spent 11–15 generations. On average, the respondents spent 11.14 generations before obtaining satisfactory patterns.

We found that the expert in Thai drawings (level (5) spent 6–10 generations to evolve satisfactory *kranok* patterns. The respondents who spent more than 15 generations were not quite familiar with Thai drawings (having an art-skill level of 2.5 on average) as illustrated in Fig. 22. Most respondents who were familiar with (or expert in) evolutionary algorithms (level 4–5) spent more than 15 generations to achieve at least one desirable pattern. These survey results indicate that experience in Thai drawings helped respondents to spend fewer generations in evolving a satisfactory *kranok* pattern.

Q.4: Overall, how would you rate the application?

Of the 28 respondents, 17 (60.7%) rated the application as high quality, 7 (25%) rated it as moderate quality, 3 (10.7%) rated it as low quality, and 1 (3.6%) rated it as very high quality, as illustrated in Fig. 23.

Q.5: Any other comments and suggestions?

The respondents gave useful comments and suggestions summarized as follows:

- Some patterns contain too many overlapping lines. If a *gaab* already overlaps another, it should not be overlapped by a third. The complicated curves make the *kranok* patterns messy. For example, the curves in the middle of the *kranok* pattern in Fig. 24 are less attractive.

- The respondents liked the creativity in the application (especially the idea of allowing users to influence the evolutionary process).

- The respondents agreed that users without any relevant expertise could use the application.

- The population size of 9 was too large: a size of 1–6 was suggested. Two respondents found it hard to consider as many as nine figures at the same time.

- The application should provide a wider variety of patterns.

- The style of the notch was quite inaccurate according to traditional standards of craftsmanship. (Note: this comment was given by only one out of twenty-eight respondents. The view might perhaps be attributable to a conservative aesthetic taste rather than a perception of 'inaccuracy' in the strict sense of the word. In this work, we focus on using machines to generate various styles of *kranok* patterns, so it is not surprising that the style of the notches diverged from traditional models. Our aim is not to replace artists by machines, but to deal with the challenges of generating Thai artworks using machines, helping users to express their artistic skills. Nevertheless, we will embrace this comment and improve the algorithm in our future work in such a way as to generate *kranok* patterns of a more traditional character.)

5.3. Submitted patterns

After creating the patterns, the respondents were asked to upload their most favored generated patterns to our server. The 14 submitted *kranok* patterns are illustrated in Fig. 25 below. We divided the respondents into three major categories: (1) those who prefer *kranok* patterns without any ornamental details (as in h, i, and j), (2) those who favor patterns with limited ornamental details, i.e., just a few notched *gaabs* (as in b, c, d, g, and m), and (3) those who incline toward patterns with more numerous ornamental details (as in a, e, f, k, and n).

The respondent who identified herself as an expert in both Thai arts and evolutionary algorithms sent us the pattern shown in Fig. 25(h), which is entirely without notches. (As mentioned in Section 5.2, she declared herself very pleased with the pattern.) For the respondents in the second and third categories, we observe that they share a common preference: the curves numbered 3 and 11 are notched. (See Fig. 8(c) for the curve-number reference).

6. Survey results on user fatigue and rand button

6.1. User fatigue

User fatigue is one of the most important issues in the design of Interactive Evolutionary Algorithms. We designed our application in such a way that users would not spend too much effort to obtain satisfactory patterns. Specifically, during the human-in-the-loop evaluation, users are not required to give a score to each

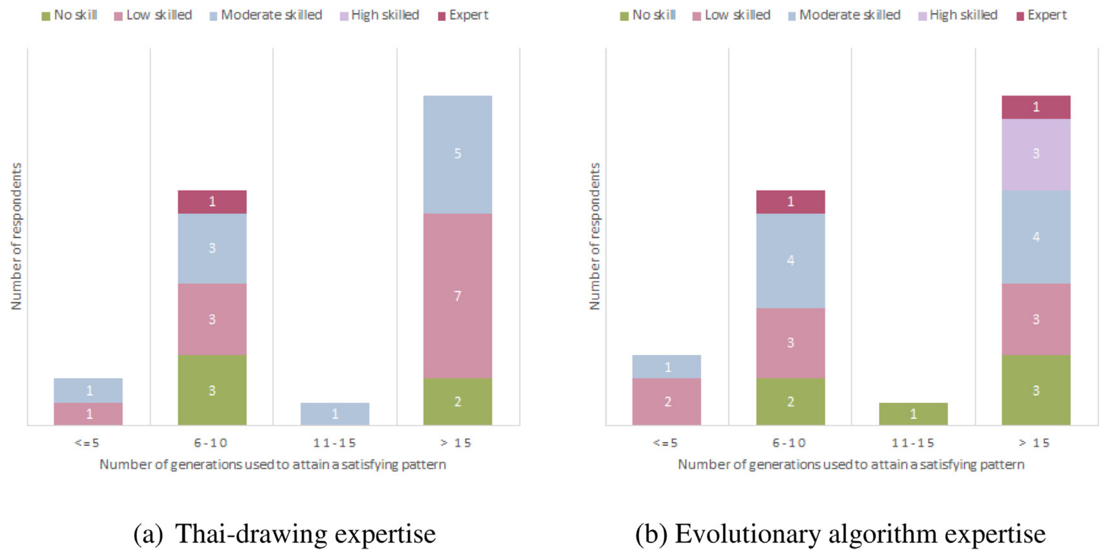


Fig. 22. Number of generations used to achieve satisfying patterns.

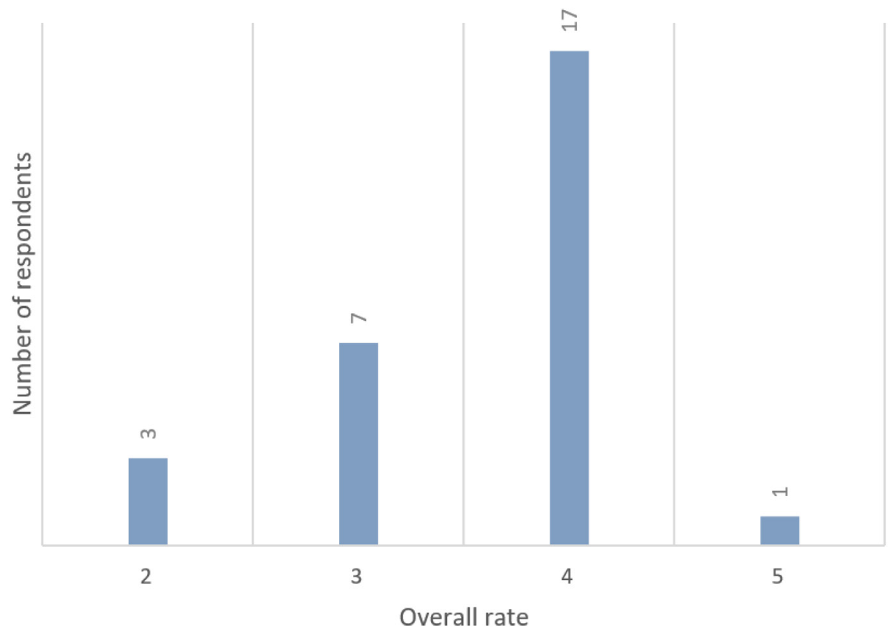


Fig. 23. The application's overall rate.

generated pattern (in contrast to most IEA applications [9,10,12, 22,26,28,30]). Checkboxes in the application allow users to choose the patterns they like within a shorter time.

To study user fatigue, we asked the respondents to what extent they felt tired while attempting to generate satisfactory patterns in each of the two phases, i.e., shape evolution and evolution of ornamental details. Users were asked to measure their tiredness on a five-point scale, viz., not tired, slightly tired, moderately tired, very tired, and utterly exhausted. All 28 respondents responded to this question, and reported their degrees of fatigue in both phases as detailed below.

During the shape-evolving phase, out of the 28 respondents, 19 (67.9%) did not feel tired, 7 (25.0%) felt slightly tired, and 2 (7.1%) felt moderately tired.

During the ornament-evolving phase, out of the 28 respondents, 15 (53.6%) did not feel tired, 8 (28.6%) felt a slightly tired, 3 (10.7%) felt moderately tired, and 2 (7.1%) felt very tired.

As shown in the results, more than half of the respondents did not feel tired at all after using the application. No respondent felt very tired or utterly exhausted during the first phase. In the second phase (generating ornamental details), there were five who felt tired or very tired. The user fatigue during this second phase is not surprising because the ornament-evolving process is designed to generate more intricate details on *kranok* curves.

6.2. *Rand button*

In this work, the *Rand* button is introduced to help users replace an unsatisfactory pattern with a totally new one. We conducted two experiments to compare the users' fatigue and satisfaction when they are either allowed or not allowed to click the Rand button. Out of the 28 participants, 25 (89.3%) agreed that the application should provide the Rand button because it allowed individual users to generate new patterns closer to their preferences. On the other hand, 3 respondents (10.7%) were not in favor of using the button as it caused patterns with certain desirable parts to disappear, and this required further generations to obtain parts with equal aesthetic appeal. We surveyed the respondents' experience when they interacted with the application with and without the Rand button. There were 18 respondents (64.3%) who felt uncomfortable when the application interface lacked the Rand button. In addition, 18 respondents (64.3%) felt less tired when using the application with the Rand button. The results imply that the Rand button is a promising option that assists the algorithm to escape instantly from local optima, enabling users to find satisfactory patterns for being potential candidates in the current population, allowing the evolutionary process to generate more attractive patterns faster.

We also studied users' behavior when using the Rand button. In the process of evolving *kranok* shapes, the average number of times that respondents pressed the Rand button was 21. In the process of evolving ornamental details, the average number of times was 36.7. Thus the respondents pressed the Rand button more often in the ornament-evolving stage than in the shape-evolving stage. This indicates that it is harder for the algorithm to generate ornamental details that satisfy users (i.e., the chances of becoming trapped in the local optima are higher) thus requiring the respondents to press the Rand button more frequently.

Furthermore, we asked the respondents in what situations they clicked the Rand button. Of the 28 respondents, 17 (60.7%) pressed it after 3 generations, 8 (28.6%) during the first 3 generations, 3 (10.7%) used the Rand button at the final generation right before they confirmed the shapes. Almost all (24 respondents) told us that they clicked the Rand button when they were unsatisfied with the current generated patterns, and attained more desirable patterns after clicking it.

7. Discussion

In this section, we provide discussion and guidance based on the study results. There are four subtopics in this discussion, as follows.

7.1. Population size

Determining the appropriate population size is one of the most important tasks of practitioners in the evolutionary computing (EC) field. Generally, EC practitioners tend to increase the population size until they discover the ones appropriate for each specific problem. Therefore, a large population size is usually chosen in actual use. In the case of interactive evolutionary algorithms (IEA) however, the method of choosing population size is different. Here, the population size should be small enough to provide user-friendly interfaces and to avoid user fatigue.

In this work, the question of population size was one of our design concerns. To the best of our knowledge, there is no research literature addressing the appropriate population size for an IEA. Our application was initially designed with a population size of 4 candidate *kranok* patterns. However, after the preliminary experiments, we found that the generated patterns were not sufficiently diverse. The population size was therefore gradually increased to 9 in order to generate a wider variety of patterns. We consider this an appropriate size because a larger population could cause user fatigue and make the application interface unfriendly to users.

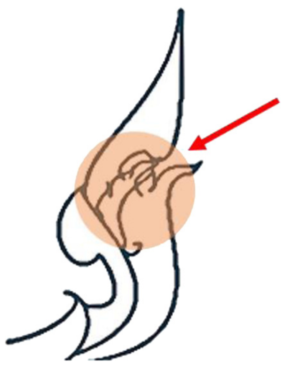


Fig. 24. Overlapping curves.

7.2. The initial population

In IEA, there is a trade-off between user fatigue and variety. If the algorithm is allowed freely and randomly to generate patterns, users might tire too early in the processes of evaluating and giving feedback, without yet having obtained satisfactory patterns. For example, if the algorithm generated the Bézier curves randomly and independently, the resulting images might include eccentric and unappealing shapes that are remote from the *kranok* tradition. If, contrariwise, we adopted a biased initial population (e.g., using popular *kranok* patterns), the searching time might be shorter, but the generated patterns would probably be too similar to the original standard ones, thus blocking the creativity and artistic invention of the user.

To avoid these extremes, we attempted to keep a balance between a random and a biased initial population. The first population was not initialized by seeding famous *kranok* patterns, which most people would probably prefer, on the grounds that such a strategy might trap the algorithm in a local optimum, not promoting variety. Instead, our initial population was comprised of *kranok* patterns randomly generated under predefined conditions. For example, certain curves must have their starting points on a part of another curve or the end of another line. Some curves must be above others. In summary, we chose to create randomly a variety of initial patterns (under predefined conditions) instead of starting with famous patterns because our aim was to help users create *kranok* patterns personalized to their own preferences.

7.3. The effect of the Rand button

The Rand button is located at the bottom-left of each *kranok* pattern presented in the application. When users click this button, the associated pattern will be replaced with a new randomly generated one. This feature is quite uncommon in the IEA field. Allowing users to explore freely the search space during the evolutionary process has its pros and cons. The advantage of the Rand button is that it helps users eliminate unsatisfactory candidates from the population and replace them with more favored ones. Users do not have to waste time waiting for the evolutionary process to gradually phase out the unwanted patterns. The button can also help reduce user fatigue (according to our experiment results in Section 6.2). The disadvantage of using the Rand button is the destruction of potentially valuable chromosome information. Users might remove patterns that, while unsatisfactory overall, possess potentially satisfactory elements, e.g., elegant curves. However, this would not be a major problem if users eliminate only those patterns that they deem unsatisfactory in all particulars. Without the Rand button, such chromosome information will eventually be culled out via the selection process, which might take a few (or more) generations.

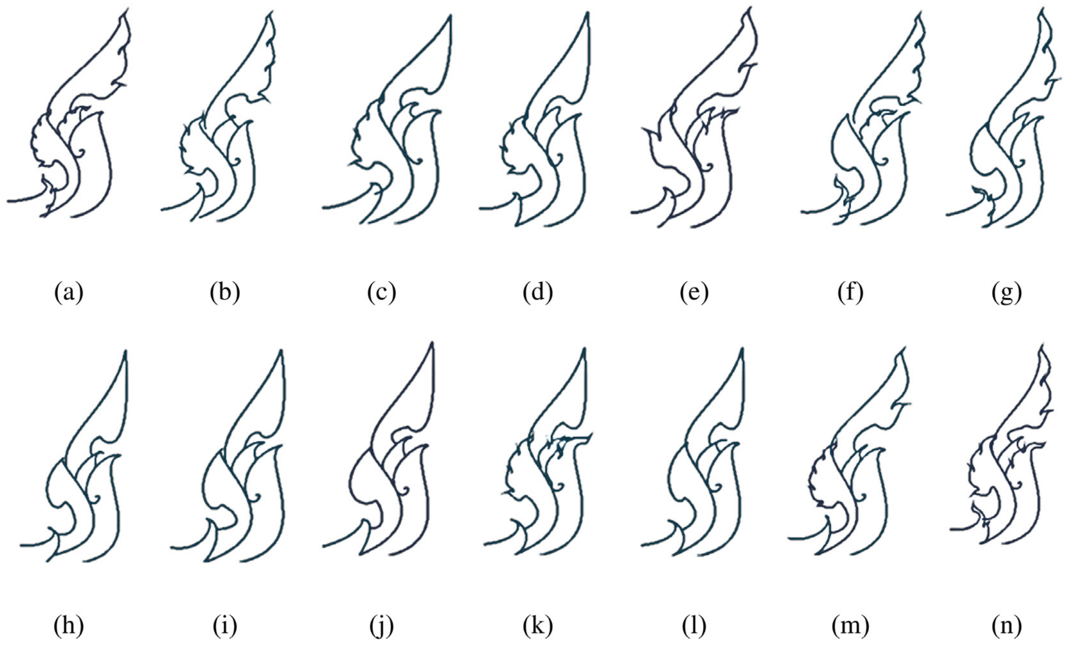


Fig. 25. Evolved *kranok* patterns submitted by the respondents.

7.4. Limitation of the proposed application

In this work, we focused on generating the main components of *kranok* patterns and their decorative details, not other delicate subcomponents. Evolving these fine details without tiring users is challenging but is included in our plan for future work.

For the ease and convenience of users with limited artistic skills, the values of certain parameters (such as the population size, the maximum number of notches, crossover rate and mutation rate) were fixed. We encourage future developers to include an option (by keeping user fatigue in mind) to allow experienced users greater freedom to assign values to a wider range of parameters.

8. Conclusion

We applied an interactive evolutionary algorithm (IEA) to generate traditional Thai patterns of the *kranok* type. It is challenging to design an automated process that can emulate the creative work of skilled artists. The principal difficulty is to generate patterns that, while being novel and distinctive, suffer no loss in aesthetic appeal, the latter quality being so closely associated with human sensibility in general and the individual subjectivity of the creative artist in particular. We therefore designed the IEA in such a way as to allow users to participate in the evolutionary process of pattern generation. Users were enabled to give feedback on the generated patterns: they could retain specific aspects that they found satisfactory, while replacing other aspects; or they could choose to generate totally new patterns. In this way, the IEA rapidly learned to generate new patterns that could satisfy the preferences of individual users. Our studies showed that the IEA required only a small number of generations to converge: users could obtain their desired *kranok* patterns with ease. Furthermore, most users declared themselves highly satisfied with the application. In sum, we have found that it is possible for a machine to produce a variety of delicate, aesthetically pleasing *kranok* patterns. This work could be developed further, with the aims of enhancing the aesthetic qualities of the generated *kranok* patterns, and of extending the IEA to the production of other types of traditional Thai patterns.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106121.

CRediT authorship contribution statement

Nutthanon Leelathakul: Conceptualization, Software, Validation, Formal analysis, Investigation, Writing - review & editing, Supervision, Funding acquisition. **Sunisa Rimcharoen:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Visualization, Supervision, Project administration.

Acknowledgments

This work was supported by the faculty of Informatics, Burapha University, Thailand [grant number 2/2557]. Special thanks to Mr. Napapol Aroonsirilert for his help in programming.

References

[1] K. Wenk, Notes on the history of the art of mother-of-pearl in Thailand with particular reference to the doors on the Ubosot of Wat Phra Chetuphon, J. SIAM Soc. 88 (1 & 2) (2000) 1–14.
[2] P.J. Bentley, D.W. Corne, Creative Evolutionary Systems, Morgan Kaufmann, 2002.
[3] V.G. Ivancevic, T.T. Ivancevic, Computational mind: A complex dynamics perspective, in: Series in Computational Intelligence, vol. 60, Springer, Berlin, Heidelberg, 2007, p. 243.
[4] S. Kanchanakun, Sen Sai Lai Thai: Chut Lai Thai Chabap Somboon 1, Setthasin Publishing, Bangkok, 2004.
[5] H.-H. Chang, H. Yan, Vectorization of hand-drawn image using piecewise cubic Bézier curves fitting, Pattern Recognit. 31 (11) (1998) 1747–1755.
[6] S. Pal, P. Ganguly, P.K. Biswas, Cubic Bezier approximation of a digitized curve, Pattern Recognit. 40 (2007) 2730–2741.
[7] A. Masood, M. Sarfraz, Capturing outlines of 2D objects with Bézier cubic approximation, Image Vis. Comput. 27 (2009) 704–712.
[8] X. Han, X. Guo, Optimal parameter values for approximating conic sections by the quartic Bézier curves, J. Comput. Appl. Math. 322 (2017) 86–95.
[9] H.S. Kim, S.B. Cho, Application of interactive genetic algorithm to fashion design, Eng. Appl. Artif. Intell. 13 635–644.

[10] D.-W. Gong, G.-S. Hao, Y. Zhou, X.-Y. Sun, Interactive genetic algorithms with multi-population adaptive hierarchy and their application in fashion design, Appl. Math. Comput. 185 (2007) 1098–1108.

[11] P.Y. Mok, X.X. Wang, J. Xu, Y.L. Kwok, Fashion sketch design by interactive genetic algorithms, in: Proceedings of the Sixth Global Conference on Power Control and Optimization, AIP Conference Proceedings, vol. 1499, pp. 357–364.

[12] M. Sugahara, M. Miki, T. Hiroyasu, Design of Japanese Kimono using interactive genetic algorithm, in: Proceedings of International Conference on Systems, Man and Cybernetics, 2008, pp. 185–190.

[13] M. Brintrup, J. Ramsden, H. Takagi, A. Tiwari, Ergonomic chair design by fusing qualitative and quantitative criteria using interactive genetic algorithms, IEEE Trans. Evol. Comput. 12 (2008) 343–354.

[14] O. Bandte, A broad and narrow approach to interactive evolutionary design – An aircraft design example, Appl. Soft Comput. 9 (2009) 448–455.

[15] R. Akase, Y. Okada, 3D room layout system using IEC (Interactive evaluational computation), in: Encyclopedia of Computer Graphics and Games, Springer International Publishing, 2015, pp. 1–17.

[16] L.G. Hernandez, L.S. Morera, A.A. Azofra, An interactive genetic algorithm for the unequal area facility layout problem, in: Advances in Intelligent and Soft Computing, vol. 87, Springer, Berlin, Heidelberg, 2011, pp. 253–262.

[17] R. Dou, C. Zong, G. Nan, Multi-stage interactive genetic algorithm for collaborative product customization, Knowl.-Based Syst. 92 (2016) 43–54.

[18] R. Dou, C. Zong, M. Li, An interactive genetic algorithm with the interval arithmetic based on hesitation and its application to achieve customer collaborative product configuration design, Appl. Soft Comput. 38 (2016) 384–394.

[19] C.L. Simons, I.C. Parmee, Elegant object-oriented software design via interactive evolutionary computation, IEEE Trans. Syst. Man Cybern. C 42 (6) (2012) 1797–1805.

[20] N. Monmarche, G. Nocent, M. Slimane, G. Venturini, P. Santitni, Imagine: a tool for generating HTML style sheets with an interactive genetic algorithm based on genes frequencies, in: Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, 1999, pp. 640–645.

[21] T. Yokoyama, H. Takenouchi, M. Tokumaru, N. Muranaka, Website design system based on an interactive genetic algorithm using tournament evaluation by multiple people, in: Proceedings of the 6th International Conference on Soft Computing and Intelligent Systems and the 13th International Symposium on Advanced Intelligent Systems, 2012, pp. 2260–2263.

[22] D. Sorn, S. Rimcharoen, Web page template design using interactive genetic algorithm, in: Proceedings of the International Conference on Computer Science and Engineering Conference, 2013, pp. 206–211.

[23] D. Yoon, K.-J. KIM, 3D game model and texture generation using interactive genetic algorithm, Comput. Entertain. 14 (1) (2017) 1–16.

[24] J. Martinez, J.-S. Sottet, A.G. Frey, T.F. Bissyandé, T. Ziadi, J. Klein, P. Temple, M. Acher, Y. le Traon, Towards estimating and predicting user perception on software product variants, in: Lecture Notes in Computer Science, vol. 10826, Springer, Cham, 2018, pp. 23–40.

[25] J. Martinez, J.-S. Sottet, A.G. Frey, T.F. Bissyandé, T. Ziadi, T. Bissyandé, J. Vanderdonckt, J. Klein, Y. le Traon, Variability management and assessment for user interface design, in: Human Centered Software Product Lines, in: Human–Computer Interaction Series, Springer, Cham, 2017, pp. 81–106.

[26] M. Miki, H. Orita, S.H. Wake, T. Hiroyasu, Design of sign sounds using an interactive genetic algorithm, in: Proceedings of the International Conference on Systems, Man, and Cybernetics, IEEE, 2006, pp. 3486–3490.

[27] E.J. Hastings, R.K. Guha, K.O. Stanley, Automatic content generation in the galactic arms race video game, IEEE Trans. Comput. Intell. AI Games 1 (4) (2009) 245–263.

[28] L. Cardamone, D. Loiacono, P.L. Lanzi, Interactive evolution for the procedural generation of tracks in a high-end racing game, in: Proceedings of Genetic and Evolutionary Computation Conference, 2011, pp. 395–402.

[29] K. Yoshida, Y. Nakagawa, M. Köppen, Interactive genetic algorithm for font generation system, in: World Automation Congress, 2010, pp. 1–6.

[30] S. Lv, S. Wang, X. Wang, Emotional speech synthesis by XML file using interactive genetic algorithms, in: Proceedings of the World Summit on Genetic and Evolutionary Computation, 2009, pp. 907–910.

[31] A.E. Eiben, Evolutionary reproduction of Dutch masters: the Mondriaan and Escher evolvers, in: The Art of Artificial Evolution, in: Natural Computing Series, Springer, Berlin, Heidelberg, 2008, pp. 211–224.

[32] J. Lv, M. Zhu, W. Pan, X. Liu, Interactive genetic algorithm oriented toward the novel design of traditional patterns, Information 10 (2) (2019) 36.

[33] M. Hailemariam, B. Goertzel, T. Yohannes, Evolving 3D facial expressions using interactive genetic algorithms, in: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 274, Springer, Cham, 2019, pp. 492–502.

[34] Q. Madera, O. Castillo, M. García-Valdez, A. Mancilla, A method based on interactive evolutionary computation and fuzzy logic for increasing the effectiveness of advertising campaigns, Inf. Sci. 414 (2017) 175–186.

[35] M. Arevalillo-Herráez, F.J. Ferri, S. Moreno-Picot, Distance-based relevance feedback using a hybrid interactive genetic algorithm for image retrieval, Appl. Soft Comput. 11 (2011) 1782–1791.

[36] C.J. Solomon, S.J. Gibson, J.J. Mist, Interactive evolutionary of facial composites for locating suspects in criminal investigations, Appl. Soft Comput. 13 (2013) 3298–3306.

[37] J.J. Mist, S.J. Gibson, Optimization of weighted vector directional filters using an interactive evolutionary algorithm, in: Proceedings of Genetic and Evolutionary Computation Conference, 2013, pp. 1691–1694.

[38] S.R. Bergen, Evolving stylized images using a user-interactive genetic algorithm, in: Proceedings of Genetic and Evolutionary Computation Conference, 2009, pp. 2745–2748.

[39] A.B. Ruiz, M. Luque, F. Ruiz, R. Saborido, A combined interactive procedure using preference-based evolutionary multiobjective optimization. Application to the efficiency improvement of the auxiliary services of power plants, Expert. Syst. Appl. 42 (2015) 7466–7482.

[40] S.B. Ismail, F. Legras, G. Coppin, A new interactive evolutionary algorithm for the vehicle routing problem, in: Proceedings of Genetic and Evolutionary Computation Conference, 2012, pp. 661–662.

[41] J.C. Quiroz, A. Banerjee, S.J. Louis, IGAP: Interactive genetic algorithm peer to peer, in: Proceedings of Genetic and Evolutionary Computation Conference, 2008, pp. 1719–1720.

[42] H. Takagi, M. Ohsaki, Interactive evolutionary computation-based hearing aid fitting, IEEE Trans. Evol. Comput. 11 (2007) 414–427.

[43] E.W. Lameijer, J.N. Kok, T. Back, A.P. IJzerman, The molecule evaluator. an interactive evolutionary algorithm for the design of drug-like molecules, J. Chem. Inf. Model 46 (2) (2006) 545–552.

[44] H. Takagi, Interactive evolutionary computing: fusion of the capacities of EC optimization and human evaluation, Proc. IEEE 89 (9) (2001) 1275–1296.

[45] C. Salelanon, Wi Cha Wat Kian Pap Thai, Jan Wa Publishing, Bangkok, 1950.

[46] B. Changchaya, Sin la Bpa Lai Thai, Chomromdek Publishing, Bangkok, 2002.

[47] W.M. Spears, K.A. De Jong, On the virtues of parameterized uniform crossover, in: Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufman, 1991, pp. 230–236.

[48] R. Larson, R.P. Hostetler, B.H. Edwards, Calculus, fourth ed., D. C. Heath and Company, 1990.