Evotype: Evolutionary Type Design

Tiago Martins, João Correia, Ernesto Costa, and Penousal Machado

CISUC, Department of Informatics Engineering,
University of Coimbra, 3030 Coimbra, Portugal
{tiagofm,jncor,ernesto,machado}@dei.uc.pt

Abstract. An evolutionary generative system for type design, Evotype, is described. The system uses a Genetic Algorithm to evolve a set of individuals composed of line segments, each encoding the shape of a specific character, i.e. a glyph. To simultaneously evolve glyphs for the entire alphabet, an island model is adopted. To assign fitness we resort to a scheme based on Optical Character Recognition. We study the evolvability of the proposed approach as well as the impact of the migration in the evolutionary process. The migration mechanism is explored through three experimental setups: fitness guided migration, random migration, and no migration. We analyse the experimental results in terms of fitness, migration paths, and appearance of the glyphs. The results show the ability of the system to find suitable glyphs and the impact of the migration strategy in the evolutionary process.

Keywords: type design, evolutionary design, island model.

1 Introduction

Although conventional computational design tools are effective for precise design tasks during the later phases of the design process, they offer insufficient support to design exploration during the earliest, essentially conceptual, stages of the design process.

We present an evolutionary system for type design — Evotype. Although it is still a work in progress, the system is already able to automatically generate alternative designs for glyphs from scratch. A glyph consists in a specific graphic expression of a given readable character. For the purposes of this article, we focus in the evolution of glyphs for letters of the Roman alphabet.

The main contribution presented herein is a functional prototype of a generative system capable of creating consistent glyphs. Other contributions include: (i) a Genetic Algorithm (GA) with a generic representation wherein individuals are composed by line segments encoded as sequence of numeric values; (ii) a fully autonomous evolutionary approach for the evolution of glyphs; (iii) the use of an island model and the study of the impact of migration on evolution; and (iv) the use of a Machine Learning (ML) approach to guide the evolution and the migration process.

The paper is organised as follows: section 2 presents related work, considering applications of evolutionary techniques in the domain of glyphs design; section 3 thoroughly describes the GA of the proposed system; section 4 describes the experimental setup; section 5 presents the analysis of the experimental results; finally, conclusions and future work is presented in section 6.

2 Related Work

Designers, engineers, artists, and scientists have been using evolution-based techniques to support the creative process and evolve innovative artefacts. Although evolutionary glyph design is a relatively unexplored area, some applications exist.

Butterfield and Lewis [1] employ Interactive Evolutionary Computation (IEC) to explore the creation of fonts. More specifically, they evolve deformations, i.e. the letters of a specific typeface are deformed by a set of implicit surface primitives, which are encoded in the genotypes. Lund [2] also uses IEC to evolve the settings of a parametric typeface system. Each parameter controls the appearance of a given characteristic of the font. Unemi and Soda [3] propose an IEC system for the design of Japanese Katakana glyphs. Schmitz [4] presents the interactive program *genoTyp*, which allows the user to create new fonts through the breeding of existing ones, according to genetic rules and manual manipulation of their genes. The possibility of recombining famous typefaces is exciting, however, the limitations of the representation hinder the quality of the results. Levin et al. [5] use IEC to implement the *Alphabet Synthesis Machine*, a system which allows the creation and evolution of abstract letter forms.

Despite evolutionary systems for glyph design exist, as far as we know all of them rely on user evaluation. Thus, the user interactively iterates a cyclic process of selection and generation until an acceptable solution is obtained. As such, all suffer from the well-known limitations of IEC systems, namely user-fatigue and inconsistency in evaluation. Additionally, they require the creation of a parametric typeface (e.g., [2]), or pre-existing typefaces or skeletons (e.g., [4] and [1], respectively), and are conditioned by these requirements.

3 The Approach

Evotype evolves glyph designs for various characters in a parallel and autonomous way. To achieve this, a GA [6] is implemented to evolve different populations of candidate glyph designs. Each individual is a glyph design. Each population lives in its own island and is composed of individuals that represent a specific character. Thus, to evolve glyphs for 26 characters we use 26 populations in 26 islands. The different islands can communicate with each other, allowing the migration of glyphs among them.

The system is schematically represented in figure 1 and behaves as follows. The evolutionary process begins with the initialisation of all populations with randomly created glyphs. The individuals are evaluated and then selected for mating according to their fitness. Recombination and mutation operators are applied to generate offspring. The selection stage follows, determining which individuals proceed to the next generation. The next step is migration, where

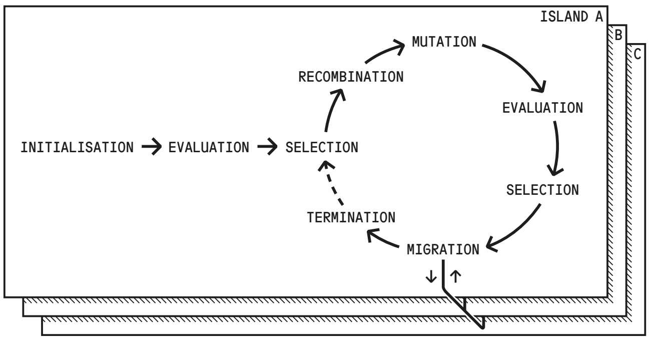


Fig. 1. Overview of Evotype.

individuals may be moved to or received from other islands. As depicted in figure 1 the process is cyclically repeated. The termination criterion is based on the number of generations. In the following subsections we detail the genetic mechanisms employed in Evotype.

3.1 Representation

The genotype consists of a sequence of genes encoding a glyph. Each gene codifies a two-dimensional line segment that is composed by a sequence of five numbers, wherein the first four are the coordinates of its end points and last one correspond to its thickness (see figure 2). The genotype' length may vary from individual to individual, thus different individuals can be composed of different number of line segments.

A mapping mechanism, normally referred to as *embryogenesis*, is in charge for the expression of the genotype into a perceptible artefact—the phenotype. The phenotype consists in a graphical representation of the genotype, i.e. a glyph created from the encoded parameters. The expression process consists in the drawing of dark line segments, defined by the genotype, on a white canvas, as illustrated in figure 3.

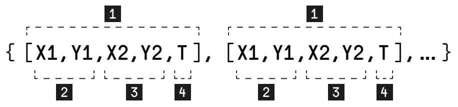
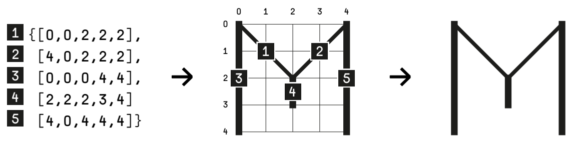


Fig. 2. Encoding of the genotype, composed of genes that codify line segments (1) defined by their end points (2 and 3) their thickness (4).

The dimension of the search space is reduced through the use of a rectangular grid that constraints the coordinates of the line segments' end points, which must adhere to the grid points. The density of the grid is configured by



```
1 {[0,0,2,2,2],
2  [4,0,2,2,2],
3  [0,0,0,4,4],
4  [2,2,2,3,4]
5  [4,0,4,4,4]}
```

Fig. 3. Mapping process from genotype to phenotype. On the left, the genotype; On the middle, an intermediate representation depicting the grid and the correspondence between genes and line segments; On the right, the phenotype.

the user.The representation ensures high locality. Neighbouring genotypes are mapped to similar phenotypes, meaning that small modifications of the genetic code induce small changes in the phenotypic space.

3.2 Initialisation

The initial populations are seeded with randomly generated glyphs. Each glyph of the first population is composed of a single line segment, with all the gene values set by uniform random selection over the admissible interval for each parameter. All islands receive identical initial populations. This initialisation setup provides equality and simplicity among all initial populations, enabling us to access the ability of the system to evolve glyphs for different characters, starting from the same set of random glyphs.

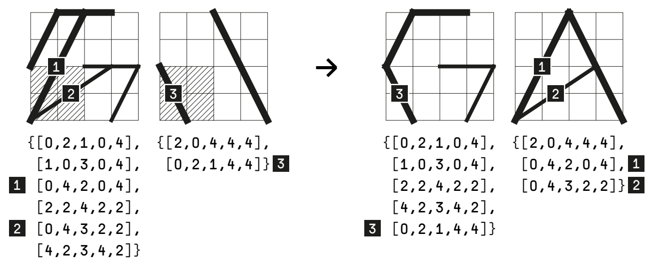
3.3 Crossover

The crossover operation consists in the exchange of line segments between parents. Crossover operates on gene boundaries, preserving the integrity of line segments. The operator proceeds as follows: randomly select a rectangular area of the grid; determine, for both parents, the line segments whose middle points are inside the rectangle; exchange those line segments between parents. As illustrated in figure 4, crossover may be asymmetric, in the sense that the number of line segments a genotype "receives" may be different from the one it "donates".

3.4 Mutation

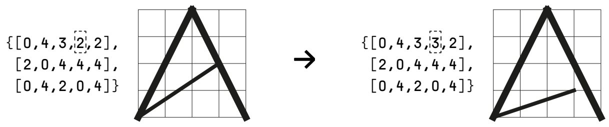
Mutation also operates on a gene basis. Thus, the mutation of a gene implies changing one of its five parameters by a value of one, as illustrated in figure 5. At the phenotype level, this variation results in the minimum translation of one of the end points of the line segment in one of the four possible directions—up, down, left, or right—or the minimum variation of its thickness. The impact of this change at the phenotype level depends on the density of the grid. A denser grid allows smaller visual variations. The probability of mutation is defined per gene, meaning that multiple genes may be mutated.

A second type of mutation exists, gene deletion and insertion, allowing the variation of the size of the genotype. There is a probability of deleting a randomly selecting gene and inserting a randomly created one.



```
  {[0,2,1,0,4],  {[2,0,4,4,4],          {[0,2,1,0,4],  {[2,0,4,4,4],
   [1,0,3,0,4],   [0,2,1,4,4]} 3         [1,0,3,0,4],   [0,4,2,0,4], 1
1  [0,4,2,0,4],                          [2,2,4,2,2],   [0,4,3,2,2]} 2
   [2,2,4,2,2],                          [4,2,3,4,2],
2  [0,4,3,2,2],                       3  [0,2,1,4,4]}
   [4,2,3,4,2]}
```

Fig. 4. Crossover process. On the left, two parents, the corresponding genotypes, and a randomly selected rectangular area (shaded region). The selected area determines the line segments that will be exchanged (1, 2, and 3). On the right, the results of the crossover operation at the genotype and phenotype level.



```
{[0,4,3,2,2],          {[0,4,3,3,2],
 [2,0,4,4,4],           [2,0,4,4,4],
 [0,4,2,0,4]}           [0,4,2,0,4]}
```

Fig. 5. Mutation process. On the left, the original genotype and phenotype; On the right, the results of the mutation operator.

Unfeasible variations are prevented during the mutation, including translations of coordinates that (i) go beyond the grid limits, (ii) create line segments with null length, or (iii) generate two line segments which are defined by the same end points.

3.5 Evaluation

As previously mentioned, each population (i.e. each island) is composed of individuals that are candidate graphic representations of a specific character. As such, the fitness of an individual depends on the environment, i.e. the island were it lives. In the scope of this paper we use Optical Character Recognition (OCR) to assign fitness. The details of the fitness assignment scheme are described in section 4.1.

3.6 Migration

Migration can occur once per each island in each generation. The probability of occurrence is determined by the migration rate. To study the influence of migration in the evolutionary process we considered three migration mechanisms:

No Migration – As the name indicates, no migration is used. The islands are isolated.

Random – When migration occurs, each island selects one random individual among the ones living on different islands. A copy of the individual is added to the population of the island, replacing the individual with worst fitness.

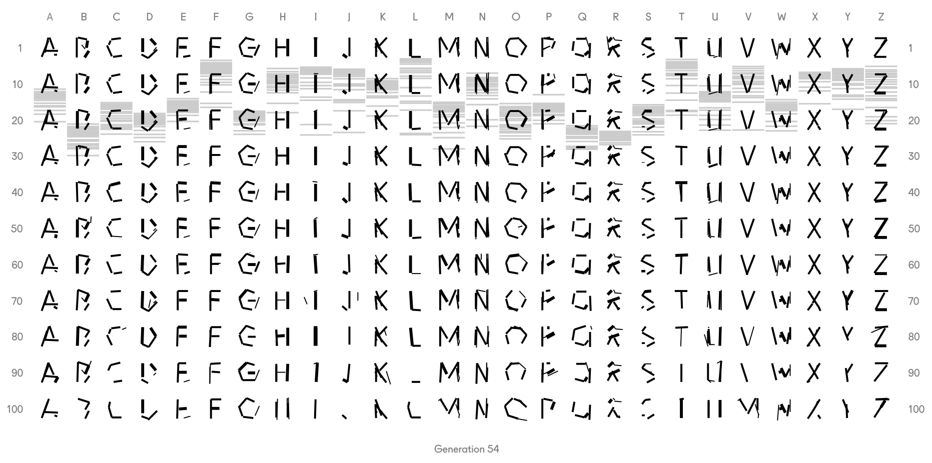


Fig. 6. Screenshot of the graphic user interface of Evotype. A demo video can be seen on `http://cdv.dei.uc.pt/2015/evotype.mov`.

Fitness Guided – When migration occurs, each island evaluates all its individuals according to the environment of the other islands. A copy of the individual that attains the highest fitness is added to the queue of that destination island. When this process is completed for all islands, each island checks its immigration queue, which may be empty, and selects the fittest individual. This individual is added to the population of the island, replacing the individual with worst fitness.

It is important to remember that in Evotype fitness is local. Therefore, the fitness of an individual depends on the environment (i.e. the character it is trying to represent graphically) and each island corresponds to a different character.

3.7 Visualisation

Evotype is able to fluidly show the evolutionary process through a simple graphic user interface, conceived to allow the user to visualise the different evolving glyphs of all islands (see figure 6). Islands (each corresponding to a character) are arranged horizontally in different columns, in alphabetical order from left to right. The individuals (glyphs) of the current generation of each island are depicted vertically, in descending order of fitness. At any moment of the evolutionary process, the user can export the evolved glyphs as vectors files to make further design refinements.

The glyphs' fitness is visualised through a simple graphic approach. The fitness value of each individual is represented through a horizontal line that overlays the corresponding column, and is vertically positioned according to the mapped value of the fitness value to the height of the interface (higher fitness values on top). We consider this fitness visualisation technique functional for the

Table 1. Evolutionary System

Parameter	Setting
Number of runs	30
Number of generations	100
Population size	100
Crossover rate	0.8
Delete gene mutation rate	0.075
Insert gene mutation rate	0.075
Change gene mutation rate	0.3
Selection method	Tournament
Tournament size	3
Elite size	1
Glyph grid size	40 × 40

Table 2. Classifier

Parameter	Setting
Input image size	48 × 48
Quantized colours	5
Threshold (θ)	200
Promotion (α)	1.005
Demotion (β)	0.995
Initial weights values	2
Training iterations	1000
Examples per island	78

purposes of this work, in the sense that we are not particularly interested in seeing the specific fitness values. Instead, we wish to visualise the distribution and convergence of fitness over time, and the comparison of fitness values among islands.

4 Experimental Setup

In this work, we evolve glyphs for all the uppercase letters of the Roman alphabet, so a total of 26 islands are considered. We conduct experiments to assess the adequacy of the engine for the evolution of glyphs. Furthermore, we study the impact of the migration policy (see section 3.6) in evolution.

The experimental parameters used in the course of the experiments described in this paper are summarised in tables 1 and 2. In the following subsections we detail the fitness assignment scheme for the experiments.

4.1 Fitness

In this work, we have 26 islands, each one evolving towards a different objective, so we need to provide a proper fitness function to guide evolution for each island. We consider that interactive evolution of all these islands' populations would be an hard and tedious task. For that reason, we sought to use an automatic fitness assignment scheme.

We are dealing with the evolution of character glyphs. Certainly, one of the preconditions of a visual character representation is its recognisability. As such, we evolve images that are recognised automatically as specific characters. We choose an OCR ML approach to automatically assign fitness. Although the use of ML to assign fitness is not novel (refer to, e.g., [7,8,9]).

Each individual is processed as a bit map image by an OCR system, and the intermediary values from OCR process are used to assign fitness. Based on the work of Burry et al. [10] we use Sparse Network of Winnow (SNoW) [11], a

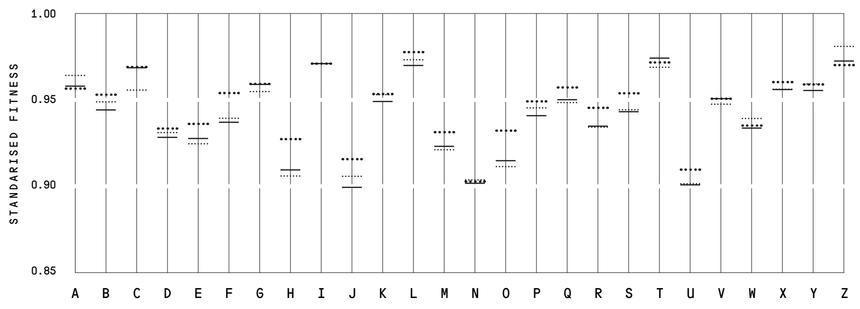


Fig. 7. Standardised fitness of the best individual of the last generation of each island. With a solid line, the *no migration*; dashed line, the *random* migration; and dotted line, the *fitness guided* migration. The visualised results are averages of 30 runs.

sparse network of linear units, as the classifier system for the OCR approach. The SNoW classifier is characterised as having two layers, the input layer and n target nodes, which are linked through weighted edges. To perform OCR, we train, offline and per character, 26 different classifiers. The input examples consist of bitmap images corresponding to the characters of 78 different typefaces. These are pre-processed to extract features, which are used in training/classification. The process and decisions made for the feature extraction are based on the work of Burry et al. [10]. Table 2 summarises the overall OCR system parameters.

To deal with the OCR multi-class classification problem we use a *one versus all* strategy for training and classification. We train each classifier by treating all the instances of the character as the positive class and all instances of all other characters as the negative class. Thus, each classifier has two target nodes, one for the positive class (i.e. the character in question) one for the negative class (i.e. all the remaining characters). The activations of the nodes indicates the degree of membership of the input image to the respective class.

To assign fitness each input image is mapped to a $2D$ space where the x coordinate corresponds to the activation value of the positive class node of the classifier, and the y coordinate corresponds to the activation value of the negative class node. Thus, the ideal scenario would be maximising x while minimising y. Unfortunately, this is typically not possible since the letters share characteristics among them. For instance, the input pattern that maximizes the x coordinate for Q, will, necessarily, yield a high y value due to the presence of examples of the letter O in the negative class. Therefore, as is often the case multi-objective optimization problems, a compromise between x and y is necessary to obtain good results.

As such, we adopted the following procedure, establish a *target* activation point empirically. We begin by calculating the input pattern that minimises the y coordinate, the activation value of the output node corresponding to the negative class for this pattern becomes y_t. Then, also analytically, we determine the input pattern for which the absolute difference between the activation values between x

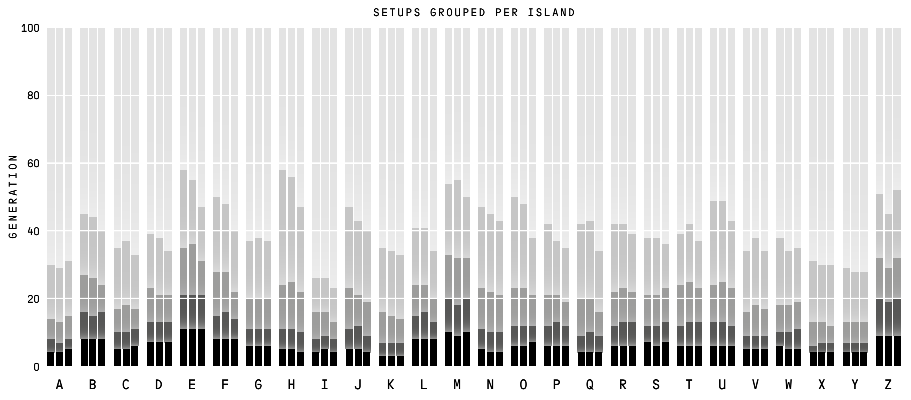


Fig. 8. Progression of the fitness of the fittest individual of each island over 100 generations. The results are grouped per island from left to right in the following order: *no migration*, *random* migration and *fitness guided* migration. Each graphic bar represents the variation of the standardise fitness of the best individual from 0 to 1 across the generations, divided in 5 different intervals of equal size (0.2). These intervals are represented by different shades of grey which change from darker to lighter according to its fitness value (from lower to higher). The visualised results are averages of 30 runs.

and y is minimal. Such input, typically, has the features necessary for the image to be classified as a member of the positive class, but still possesses features that are common with other characters. The activation value of the output node corresponding to the positive class for this pattern becomes x_t. Finally the fitness of a new input image is based on the following formula: $dist(x) = \|(x_t, y_t) - (x_i, y_i)\|$, where x_i, y_i are the activation values of the output nodes associated with the input image, and results in the euclidean distance between these two points. Since we wish to minimise the distance to this point, fitness becomes: $fitness(x) = 1/(1 + dist(x))$.

5 Experimental Results

We begin the analysis of the experimental results by focusing on the fitness values obtained by the 3 migration strategies. Figure 7 summarizes these results by presenting the average fitness of the best individual of the last generation. For the purposes of readability the fitness values have been normalized to $[0, 1]$, by dividing the raw fitness scores by the maximum fitness score obtained for each character in the course of the 90 runs (30 per migration strategy). A brief perusal of the results indicates that fitness based migration outperforms the other two strategies, attaining higher fitness values for 20 out of the 26 islands. There are two “ties” among strategies, namely for islands I and N. *Random* migration outperforms the other methods for three islands (A, W, and Z), while *no migration* obtains the best results for island T.

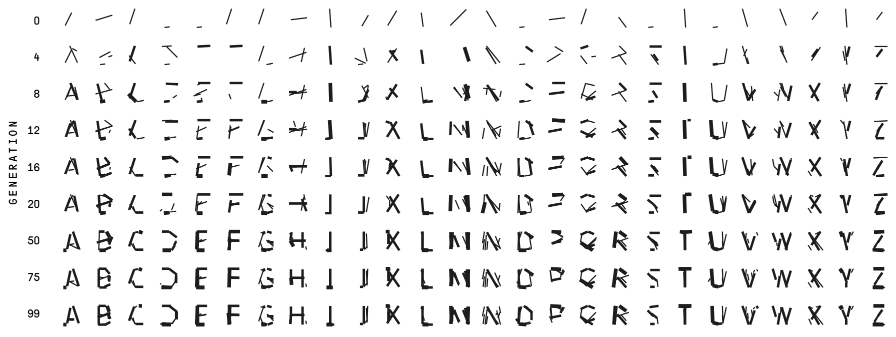


Fig. 9. Best individuals of each island for different generations from a typical run using *fitness guided* migration.

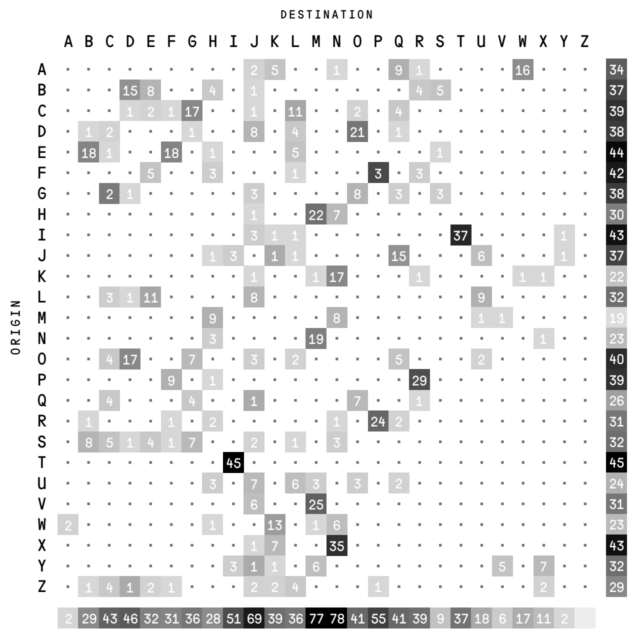
To better understand the dynamics of the evolutionary process we summarise in figure 8 the evolution of fitness over time. The results suggest that the use of migration (*fitness guided* or *random*) promotes a faster convergence for maximum fitness values. To illustrate how the evolution of fitness is reflected in the graphic appearance of the glyphs, figure 9 depicts the glyphs evolved through time in a typical run using fitness based migration.

Regarding the speed of convergence to maximum fitness values, *fitness guided* is the first approach to attain high fitness values in 22 of 26 islands, with the exceptions being the letters A, V, W, and Z. These exceptions can be justified by the analysis of the migration paths along the evolutionary process. Figure 10 depicts the average number of migrations among islands for *fitness guided* migration.

The results in these islands can be justified as follows. For the letter Z, the migration paths indicate that the island does not receive immigrants from other islands, as such the performance of the *no migration* and *fitness guided* migration should be, and is, comparable. The same justification explains the results obtained for island A and V. The explanation for the behaviour observed in island W is harder to explain, in this *fitness guided* case migration appears to be detrimental to the evolutionary process, the migration of individuals from island A to island W, although helpful in the beginning of the run, leads to premature convergence to sub-optimal solutions.

Figure 10 also shows that islands corresponding to visually similar characters migrate more individuals among them. Some examples of these migration paths are the following: A$\rightarrow$W; B$\rightarrow$D; E$\rightarrow$B,F; F$\rightarrow$P; K,X$\rightarrow$N; L$\rightarrow$E; H,N,V$\rightarrow$M; C$\rightleftarrows$G; D$\rightleftarrows$O; I$\rightleftarrows$T; and P$\rightleftarrows$R.

Although the analysis of the evolutionary process is valuable, from an evolutionary design point of view, the analysis of the visual output of the system is also important. The visual results presented in figure 11 highlight the diversity of the glyphs evolved in different runs. Additionally, and although this is difficult to measure objectively, migration appears promotes the visual coherence among the characters of different islands, which is important from a typeface design point



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	·	·	·	·	·	·	·	·	·	2	5	·	·	1	·	·	9	1	·	·	·	·	16	·	·	·	34
B	·	·	·	15	8	·	·	4	·	1	·	·	·	·	·	·	·	4	5	·	·	·	·	·	·	·	37
C	·	·	·	1	2	1	17	·	·	1	·	11	·	·	2	·	4	·	·	·	·	·	·	·	·	·	39
D	·	1	2	·	·	·	1	·	·	8	·	4	·	·	21	·	1	·	·	·	·	·	·	·	·	·	38
E	·	18	1	·	·	18	·	1	·	·	·	5	·	·	·	·	·	·	1	·	·	·	·	·	·	·	44
F	·	·	·	·	5	·	·	3	·	·	·	1	·	·	·	3	·	3	·	·	·	·	·	·	·	·	42
G	·	·	2	1	·	·	·	·	·	3	·	·	·	·	8	·	3	·	3	·	·	·	·	·	·	·	38
H	·	·	·	·	·	·	·	·	·	1	·	·	22	7	·	·	·	·	·	·	·	·	·	·	·	·	30
I	·	·	·	·	·	·	·	·	·	3	1	1	·	·	·	·	·	·	·	37	·	·	·	·	1	·	43
J	·	·	·	·	·	·	·	1	3	·	1	1	·	·	·	·	15	·	·	·	6	·	·	·	1	·	37
K	·	·	·	·	·	·	·	·	·	1	·	·	1	17	·	·	·	1	·	·	·	·	1	1	·	·	22
L	·	·	3	1	11	·	·	·	·	8	·	·	·	·	·	·	·	·	·	·	9	·	·	·	·	·	32
M	·	·	·	·	·	·	·	9	·	·	·	·	·	8	·	·	·	·	·	·	1	1	·	·	·	·	19
N	·	·	·	·	·	·	·	3	·	·	·	·	19	·	·	·	·	·	·	·	·	·	·	1	·	·	23
O	·	·	4	17	·	·	7	·	·	3	·	2	·	·	·	·	5	·	·	·	2	·	·	·	·	·	40
P	·	·	·	·	·	9	·	1	·	·	·	·	·	·	·	·	·	29	·	·	·	·	·	·	·	·	39
Q	·	·	4	·	·	·	4	·	·	1	·	·	·	·	7	·	·	1	·	·	·	·	·	·	·	·	26
R	·	1	·	·	·	1	·	2	·	·	·	·	·	1	·	24	2	·	·	·	·	·	·	·	·	·	31
S	·	8	5	1	4	1	7	·	·	2	·	1	·	3	·	·	·	·	·	·	·	·	·	·	·	·	32
T	·	·	·	·	·	·	·	·	45	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	45
U	·	·	·	·	·	·	·	3	·	7	·	6	3	·	3	·	2	·	·	·	·	·	·	·	·	·	24
V	·	·	·	·	·	·	·	·	·	6	·	·	25	·	·	·	·	·	·	·	·	·	·	·	·	·	31
W	2	·	·	·	·	·	·	1	·	·	13	·	1	6	·	·	·	·	·	·	·	·	·	·	·	·	23
X	·	·	·	·	·	·	·	·	·	1	7	·	·	35	·	·	·	·	·	·	·	·	·	·	·	·	43
Y	·	·	·	·	·	·	·	·	3	1	1	·	6	·	·	·	·	·	·	·	·	5	·	7	·	·	32
Z	·	1	4	1	2	1	·	·	·	2	2	4	·	·	·	1	·	·	·	·	·	·	·	2	·	·	29
	2	29	43	46	32	31	36	28	51	69	39	36	77	78	41	55	41	39	9	37	18	6	17	11	2		

Fig. 10. Visualisation of the average number of migrations between the different islands when using *fitness guided* migration. Origin islands are positioned horizontally and the destiny islands vertically. High numbers of migrations are highlighted by darker squares. The results are averages of 30 runs.

of view. These outputs may be provided to a type designer as alternative sources of inspiration, thus arguably assisting creation of glyphs during the conceptual phase of the creative process. The analysis of the impact of these suggestions is outside the scope of the paper, and will be left for a future opportunity.

6 Conclusions and Future Work

We have presented Evotype, an evolutionary approach for the automatic generation of character glyphs with an automatic fitness assignment scheme based on an OCR approach. The approach adopts an island model where the glyphs corresponding to each character populate each island. The experimental results shows that migration of individuals among islands is beneficial, provided that the fitness in the destination environment is taken into consideration. More importantly, the experimental results show that Evotype provides an efficient architecture to evolve and explore alternatives for glyph design.

Future work will focus on: (i) the extension of the genetic representation to allow a wider range of graphic primitives; (ii) the exploration of different migration policies and topologies; (iii) the exploration of other fitness assignment

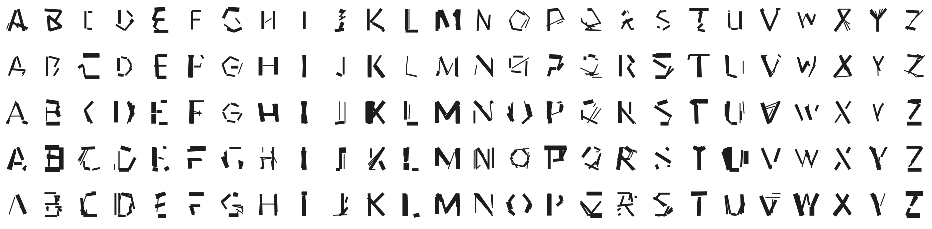


Fig. 11. Some examples of glyphs evolved in different runs.

schemes, which may promote the diversity, aesthetic appeal, and creative potential of the glyphs; and (iv) the further development of the system as tool for supporting the creativity of the designer.

References

1. Butterfield, I., Lewis, M.: Evolving fonts (2000) Consulted in `http://accad.osu.edu/~mlewis/AED/Fonts/` on October 2014.
2. Lund, A.: Evolving the shape of things to come: A comparison of direct manipulation and interactive evolutionary design. In: International Conference on Generative Art. Domus Argenia, Rome, Italy (2000)
3. Unemi, T., Soda, M.: An iec-based support system for font design. In: Proceedings of the IEEE International Conference on Systems, Man & Cybernetics: Washington, D.C., USA, 5–8 October 2003. (2003) 968–973
4. Schmitz, M.: genotyp, an experiment about genetic typography. Presented at Generative Art Conference 2004 (2004)
5. Levin, G., Feinberg, J., Curtis, C.: The alphabet synthesis machine (2006) Consulted in `http://www.alphabetsynthesis.com` on October 2014.
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan (1975)
7. Baluja, S., Pomerlau, D., Todd, J.: Towards automated artificial evolution for computer-generated images. Connection Science **6**(2) (1994) 325–354
8. Machado, P., Romero, J., Manaris, B.: Experiments in computational aesthetics: An iterative approach to stylistic change in evolutionary art. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music. Springer Berlin Heidelberg (2007) 381–415
9. Machado, P., Correia, J., Romero, J.: Expression-based evolution of faces. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design - First International Conference, EvoMUSART 2012, Málaga, Spain, April 11-13, 2012. Proceedings. Volume 7247 of Lecture Notes in Computer Science., Springer (2012) 187–198
10. Artan, Y., Burry, A., Kozitsky, V., Paul, P.: Efficient smqt features for snow-based classification on face detection and character recognition tasks. In: Image Processing Workshop (WNYIPW), 2012 Western New York. (Nov 2012) 45–48
11. Carlson, A., Cumby, C., Rosen, J., Roth, D.: The snow learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department (5 1999)