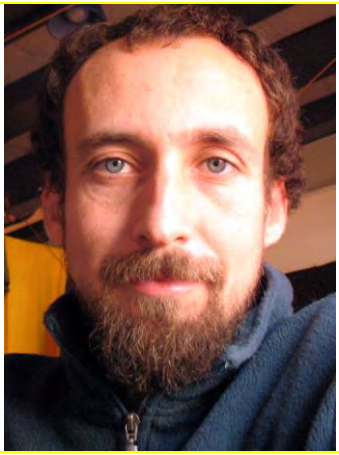
N. ONUR SÖNMEZ

Paper: **AUTOMATED EVALUATION AND GENERATION OF GRAPHIC ARRANGEMENTS THROUGH ADAPTIVE EVOLUTION**

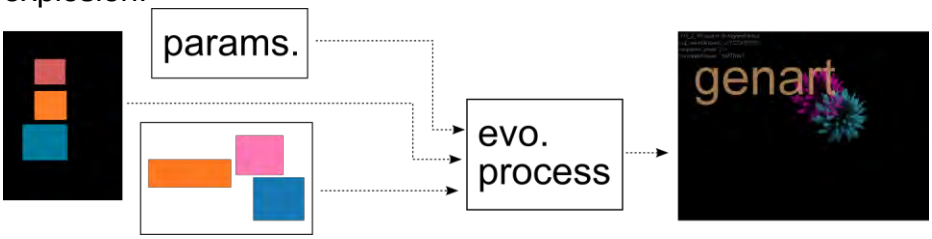


Topic: Graphic design, illustration

Authors:
N.Onur Sonmez
Istanbul Technical University, Faculty of Architecture
Delft University of Technology, Faculty of Architecture
I. Sevil Sariyildiz
Delft University of Technology, Faculty of Architecture
Arzu Erdem
Istanbul Technical University, Faculty of Architecture

References:
[1] A. Smeulders, et.al, *"Content-Based Image Retrieval at the End of the Early Years",* IEEE Trans. on P.A.M. I., 2000
[2] J. R. Smith and S. Chang, *"Integrated spatial and feature image query",* Multimedia Systems 7, 1999

Abstract:
This study explores methods to automatically generate graphical arrangements through adaptive evolutionary algorithms. A first experimentation with this idea, where all the parameters for evaluations had been manually pre-determined, had resulted in a proliferation of parameters each of which were quite sensitive, hence shown impracticality of the manual approach. Therefore a new approach is adopted, mainly inspired by 2D shape retrieval studies [1][2]. In the current implementation, although manual parameters can also be given, the main interface between the user and the system is a series of vector and bitmap images (newly created graphics, existing images, sketches, groups of images etc.)
Through these images features like desired colour distributions, textures and spatial graphical arrangements for a series of 2D design units can be determined separately. This separation enables us to create novel images using different targets for each feature type. Three usages for these features are proposed: 1. As distribution maps and color palettes for the initiation steps of the evolutionary processes. 2. As 1D-2D histograms, to measure similarity to the desired page layout or color distribution. 3. As character strings, for grid-based layout evaluation, which reorganizes and compares the layouts as rows, columns, cells and neighbor lists.
A highly adaptive evolutionary process is implemented, which is suggested to mimic the human generate-and-test procedure. Both constraints (i.e. overlap), and the proposed evaluation approaches, together with corresponding mutation operations are utilized at the same processes consecutively, according to the course of the processes.
Expected benefits of this study are: 1. Extracting styles from given images may supply us with a huge style-base implicit in existing graphics. 2. For formal parameters, images/graphics constitute a more intuitive and fast interface compared to a fully parametric approach. 3. Adaptive evolution alleviates the burden of technical parameter setting procedure. 4. Consecutive fitness functions approach enables us to apply a multitude of possibly conflicting fitness criteria at the same process while avoiding combinatorial explosion.



Process Scheme

Contact:
onursonmezn@yahoo.com

Keywords:
design automation, adaptive evolutionary algorithm, automated graphic evaluation and generation

Automated Evaluation and Generation of Graphic Arrangements through Adaptive Evolution

N.O. Sonmez, Bsc, Msc
Faculty of Architecture, Istanbul Technical University, Turkey
Faculty of Architecture, Delft University of Technology, The Netherlands
onursonmezn@yahoo.com

Prof. A. Erdem, Bsc, Msc, PhD
Faculty of Architecture, Istanbul Technical University, Turkey

Prof. I. S. Sariyildiz, Bsc, Msc, PhD
Faculty of Architecture, Delft University of Technology, The Netherlands

Abstract

This study explores methods to partially automate a graphic design task through evolutionary computation. Automating visual evaluation is not a simple task, as visual characteristics, style and taste are often hard to define, or to parameterize. Regarding this problem a series of target images are used to define desired formal characteristics. Through these images, target features (describing color distributions and spatial arrangements) for a series of 2D arrangements are determined separately. This separation enables us to create novel images using different targets for each feature type. In this study, three usages for the extracted features are proposed: 1. For the initiation steps of the evolutionary processes, as distribution maps. 2. As 1D histograms for measuring color similarity between the target distribution and candidate arrangements. 3. For grid-based layout evaluation, in order to compare the layouts in the form of rows, columns and cells.

An adaptive multi-objective evolutionary process is implemented through a consecutive micro-processes approach, which is suggested to mimic the human generate-and-test procedure. Four objective functions with corresponding mutation operators are utilized at the same run consecutively, according to the course of the process. Several test series are carried out and the approach is found applicable: 1. Each of the four objective functions is tested separately, without initiation maps, for adaptive and non-adaptive versions. While all of them successfully converged to desired levels, non-adaptive versions were slightly better. 2. For 12 scenarios, four objective functions are used together in a multi-objective process setting, with initiation maps. All scenarios succeeded in maintaining and/or obtaining desired fitness levels. Yet, when compared, non-adaptive versions are found more successful and reliable for our parameter and problem settings.

1. Introduction, problem statement and definitions

We want to introduce a series of experiments aimed at partially-automated design generation in the context of graphical arrangements. For our aims the task is defined as to generate arrangements using a canvas, basic design units (DUs, i.e. pattern stamps and text strings), and color sources (gradient and block colors). The goal is to obtain pleasing solutions, whose color and layout distributions resemble different target images, supplied by a user. The problem is chosen as an exemplary design task which involves a series of sub-tasks, a multitude of phases, several formal operations, and most important, visual evaluation.[1]

A convincing automated visual evaluation approach, which is able to take users' desires into consideration, would be indispensable for an intelligent, automated or semi-automated design system. While in artistic illustration studies attempts have been made for automated visual evaluation [1], especially in design automation studies visual evaluation, if not completely omitted or hidden beneath functional performance, is still mostly relegated to the human judges; who are summoned either during or at the end of the processes. Interactive evolutionary algorithms exemplify one case, and parametric approaches another. In the latter case, parameters are adjusted before the process. After evaluation, the user is supposed to re-adjust the parameters and restart the process until desired results are obtained. Real-time versions can also be envisioned, however, in any case defining visual characteristics by numeric parameters might become a very complex and tedious task, while interactive sessions tend to become tiresome for the human judges.

In this study, evolutionary computation is utilized to mimic the generate-and-test (i.e. evaluate) procedure of the human designers, where, in a series of successive stages, a multitude of alternatives are proposed, evaluated and selected according to a number of objective functions. Instead of an interactive process, a simple method is proposed to automatically evaluate the candidates. Desired visual characteristics are defined and fed to the system by means of target images. Images create a fast, intuitive, yet controlled interface for the human designer, thus the need for numeric parameter setting is eliminated. An image-based user interface for knowledge input for an architectural design context is exemplified in [2]. In a broader perspective, style extraction from given images may supply us with a huge style-base implicit in existing graphics. For the extraction task, content based image retrieval (CBIR) studies supply us with a wealth of fast and well-studied techniques[2] [3, 4, 5]. This study incorporates simplified variants of techniques described in [6]. Through these techniques, evaluation procedures are implemented to evaluate the candidate images according to their similarity with given target images.

Unfortunately, there still remains a large amount of parameters to be adjusted for the evolutionary process itself to perform well. These parameters are not independent

[1] In this study the phrase, 'visual evaluation' is preferred over 'aesthetic judgement'.

[2] CBIR is any technology that in principle helps to organize digital picture archives by their visual content and mainly deals with two problems: how to mathematically describe an image, and how to assess the similarity between a pair of images based on their abstracted descriptions. [4]

from each other and searching for the best values for all different combinations is practically impossible. Even if parameters are optimized one by one regardless of their interactions, the process is still time consuming. Moreover best value combinations tend to vary for each different design scenario, and even worse, these tend to change during the evolutionary process [7]. Regarding parameter adjustment problem, [7] makes a distinction between 'parameter tuning' and 'parameter control'. The former amounts to finding good values for the parameters before the running of the algorithm, which remain fixed during the run. The latter offers an alternative, as it amounts to starting a run with initial parameter values that are changed during the run, with the hope that the process finds best values itself during the process. To address this problem we have tested and compared both alternatives.

Finally, our approach utilizes a multitude of objective functions, corresponding to different aspects of the graphic arrangement. One easy solution for using multiple objective functions together is to combine the multiple objectives into a single objective function as a weighted sum [8], which, in our case, is not a straightforward task. Another approach is 'pareto optimization', which we believe is too strict for our aims. Thus, a rather tolerant approach is experimented with: Micro evolutionary processes are interleaved into a macro process according to the course of the evolution of the same population. In this way a multitude of possibly conflicting objective functions are incorporated consecutively at the same run. This is weakly reminiscent of a human designer's attitude.

2. Review

Our problem constitutes a sub-region of a broader layout generation problem. Layout generation is a long-studied field of computational design, which has many variants including artistic illustration, graphic design, user interface design and architectural/engineering layout problems.

In the field of evolutionary art, a major line of production has been expression-based image/texture generation initiated by Karl Sims [1]. Mathematical expressions are used as genotypes which can be represented as tree graph structures. When the expression represented by the tree is evaluated at a pixel coordinate, the resulting value can be used to determine the color of that pixel [1]. Candidate images are generally evaluated and ranked by a user during interactive sessions. A series of studies explored a neural net approach to learn the user's preferences from the images evolved during these sessions, and to use this knowledge to automate the evaluation stage [9, 10, 11]. Efforts have been made by [12] to use complexity estimates for images to produce automated image 'critics' with the aim of using this knowledge for automated and semi-automated image generation sessions [13]. [14] attempts to automatically evolve procedural texture formulas by matching the candidates with artistic imagery. A multi-objective pareto optimization process is employed, where candidate textures are evaluated according to several objectives including bell curve distributions of color gradients and color histogram scoring by matching a candidate texture's color composition with the color histogram of a target image [14].

In design fields layout problems generally involve distinct design units to be arranged on a 2D space/canvas according to a multitude of requirements. Constraint-based

approaches are common in visual interface layout generation, while template and grid based layout organizers are also widely utilized [15]. [16] proposes automated layout generation based on automated evaluation of visual balance, while [17] describes a flexible system for automatic page layout that makes use of genetic algorithms for albuming applications. Layout fitness is based on graphic design preferences supplied by the user according to following criteria: balance, spacing, chronology, emphasis, unity. Another line of research, which is closely related to our study formulated the architectural layout problem as an evolutionary search. [18-20] proposed to derive an objective function implicitly from a set of target spatial distributions using axial maps, and made experimentations on the representation of style.

3. Method

3.1 Target images

In our study, two types of target images are utilized: for color distribution, an arbitrary bitmap image is supplied by the user. For design unit (DU) layout, a vector graphic is generated by the user according to a layer scheme,[3] and then this vector definition is parsed with a script so that coordinates of all the DUs are acquired according to DU type. From this information, an initiation map is produced as follows: each DU center's coordinates are divided by a cell width (100 pixels is used for all tests) and the resulting cell (row nr, column nr) is placed in a dictionary holding possible initiation positions for each DU type. Preparation of target features is described below.

3.2 Genotype, initiation, selection and crossover

The genotype comprises all data related to a single candidate image, including phenotype mapping information. Both evolved and non-evolved values are stored in an individual's genotype.[4] Each DU has a static location on a fixed length chromosome. For each DU, evolved values are center coordinates and RGB values for gradient stops. In the adaptive version, for each objective function, separate lists of standard deviations for nudge and color mutations (one SD for each DU), and swap rate values (one for each individual) are also stored and evolved.

An individual is initiated by assigning center coordinates, color values and optional mutation parameters to all of its DUs. Center coordinates are initiated either from amongst possible values of the initiation map or randomly.

Three selection operators are implemented: fitness proportional, tournament and uniform. Test runs showed that each objective function required different selection combinations for the three selection thresholds (selections for crossover parents, mutation candidates and new generation)

[3] For graphic related tasks Inkscape is used, which is an open source vector graphic application.

[4] All programming and computation is done in Python language. In addition to Python's standard library, several external modules are utilized: numpy, scipy, matplotlib, PIL, pycairo, polygon, shapely and psyco.

An ‘n point crossover’ operator is preferred. Because the DU positions are static on the genotype, it is rather straightforward to exchange randomly selected DU’s between genotypes. Adaptive sigma rates and swap ratios are not exchanged, so that crossover acts as a mutate-crossover operator. Crossover/mutation rate was kept around 0.25.

3.3 Mutation operators and parameter control (adaptivity)

There are three mutation operators: Nudge, swap and color mutations. Nudge mutation is a type of ‘creep mutation’ [7]. First a number is drawn from a uniform distribution between 0 and 1. If this number is above a specified threshold (for our case 1), the unit isn’t mutated, otherwise, another number is drawn for the nudge step, this time from a Gaussian distribution with mean 0, and standard deviation (SD) s. If this step is below a threshold it is ignored, else, it is summed with the present value.

Color mutation is implemented similarly. For each of the H, S and V values, the value is mutated as above, then if the resulting value is below 0 or above 1, it is rescaled with a remainder operator so as to stay circular (-0.2 = 0.8).

In the adaptive version, for each of the above operators and for each fitness type, separate SD values for each DU are stored in an individual's genotype. These values are mutated with a similar creep mutation operator before the nudge mutations are done.

For swap mutation an individual is chosen, and then a number is drawn, if above a certain swap rate, according to another ratio, required numbers of DU couples are selected amongst the individual’s chromosome. Then center coordinates of these DU couples are exchanged. In the adaptive version, for each individual and for each fitness type separate swap rates are stored. Swap rate values are bounded at 0 and 1 and mutated using a creep mutation before the swap mutations are applied.

3.4 Objective functions

In total five objective functions (fitness types) are implemented, two of which are combined together. Hence, four different types of evolutionary processes are defined each with different mutation and selection operator combinations and parameters.

1. Out + Overlap fitness: In this project, a flexible spatial representation is used, so that the DUs can overlap each other or cross the outer boundary. This combined objective function is implemented to control outward migration and overlapping of the DUs. Each DU has a rectangular bounding box. Fitness is calculated as follows: for each candidate layout, first, all DUs are checked for outer boundary violation. If a DU is partially violating the boundary, a penalty is given according to the ratio of the violating part. If a DU is totally out of the boundaries penalty is given according to the distance from the center of the image. Secondly, all DUs are checked for overlapping other DUs. If overlap occurs overlapping area is divided by the smaller DU’s area, and a penalty is given accordingly [Fig.1]. Finally total outer boundary penalties and DU overlap penalties are combined into a single fitness value by a weighted sum. A nudge mutation is used for this fitness.

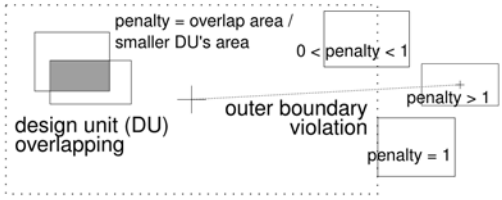


Figure 1. Penalty calculations for out + overlap fitness.

2. Cell fitness: Target feature is prepared as follows: coordinate values of each DU's center are divided by a cell width (100 pixels) and a symbol representing the DU's type is placed in a grid cells dictionary, this dictionary is stored as the fitness target [Fig.2]. For evaluation, same is applied to each candidate layout. Then, the two cell dictionaries are compared. For each cell a penalty is given for the mismatches. If a DU type is present in a specific cell of the target and not in the corresponding cell of the candidate, a penalty is applied. This fitness is meant to be tolerant for abundance cases, where candidate images contain larger numbers of DUs than the targets do.

3. Sequence fitness: Again a grid is prepared for a specified width (100 pixels), then for each row and column, symbol strings are produced according to DU types' sequences. Dots ('.') are placed for empty cells. This procedure is applied to both targets and candidates [Fig.2]. For fitness evaluation, each of a candidate image's strings is compared with the corresponding string of the target. This comparison is done by finding the "Levenshtein distance", which gives the minimum number of edits needed to transform one string into the other, with only insertion, deletion, or substitution operations.[5] This fitness has the advantage of taking relative positioning of the DUs into account, but penalizes abundance cases. Swap and nudge mutations are used for these fitnesses.

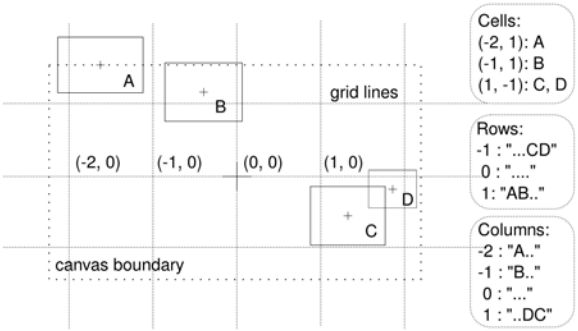


Figure 2. Cell, row and column preparation for cell and sequence fitnesses.

4. Histogram fitness: For target preparation, first the RGB values of the target image are converted into HSV mode, which is relatively more intuitive for human perception. Then for each series of H, S and V values, separate histograms are prepared and normalized so that the total area of the histogram equals 1. This step is necessary to compare images with different pixel quantities.[6] Several bin widths are tested and no considerable difference is observed; in the end, histograms with 70 bins are used. After the same procedure is applied to prepare candidate

[5] Python implementation of the Levenshtein distance is taken from the Wikibook, "Algorithm_implementation" October, 2010: http://en.wikibooks.org/wiki/Algorithm_implementation/Strings/Levenshtein_distance#Python

[6] Numpy's histogram procedure is used.

histograms, Manhattan distance is calculated between target and candidate histograms.[7] Then the three distance values corresponding to H, S and V histograms are combined with a weighted sum (equal weights are used). Color mutation is used together with an optional nudge mutation for this fitness.

3.5 Process

Before each consecutive run, an initial priority list is made amongst the fitness types. And for each fitness type, desired fitness thresholds (beginning and end), selection and mutation operators, and initiation values are determined. Care should be taken for these adjustments, because success of the run is closely connected with proper settings.

First fitness type on the priority list is set as the current fitness, and then the population is initiated either according to a map or randomly (10 individuals in our scenarios). Number of individuals is fixed. All the individuals are evaluated according to all of the fitness types, but selections are conducted according to the current fitness. After mutation and crossover (20-30 and 8-12 candidates are produced respectively), candidates are evaluated and a new population is selected from amongst the old population plus candidates. Best n individuals can be directly preserved. At this stage for all fitness types, mean fitness values are calculated. If all the fitness values are above their current threshold, all the thresholds are raised one step. Then the priority list is traversed to find the first fitness type where mean fitness is below its current threshold. This way, when the current fitness surpasses its threshold, the next fitness type is activated, and when because of conflicts current fitness decreases the mean fitness value of a prior fitness, prior one is re-activated to readjust the population. Though there are other details concerning adaptive process parameters, stability checks and runtime priority re-adjustments, basically this is the consecutive approach that is adopted.

4. Tests and results

First a series of tests are conducted to find the best process parameters; as if these were independent. Then with these parameter settings, tests are carried out to assess the performance of the objective functions separately. To better asses the relative performance of the adaptive versions, three series of tests are carried out with different population and mutation count combinations of (10-20), (40-40) and (80-80). For each fitness type: 10 x 2 (adaptive vs. non-adaptive, with the same initiation values) x 3 (population sizes) = 60 tests are carried out. Figure 3 shows target images, example results and average fitness graphics for each objective function for the first case (10-20). The “out + overlap” fitness did not perform well on adaptive versions. For the other fitnesses, in the first two cases (10-20 and 40-40), non-adaptive versions worked (mostly) slightly better, and in the last case (80-80) performances of adaptive and non-adaptive versions became roughly equal. Unfortunately, this increase in population, though lengthened the evolutionary process, did not result in considerably better fitness levels.

[7] Scipy spatial package is used.

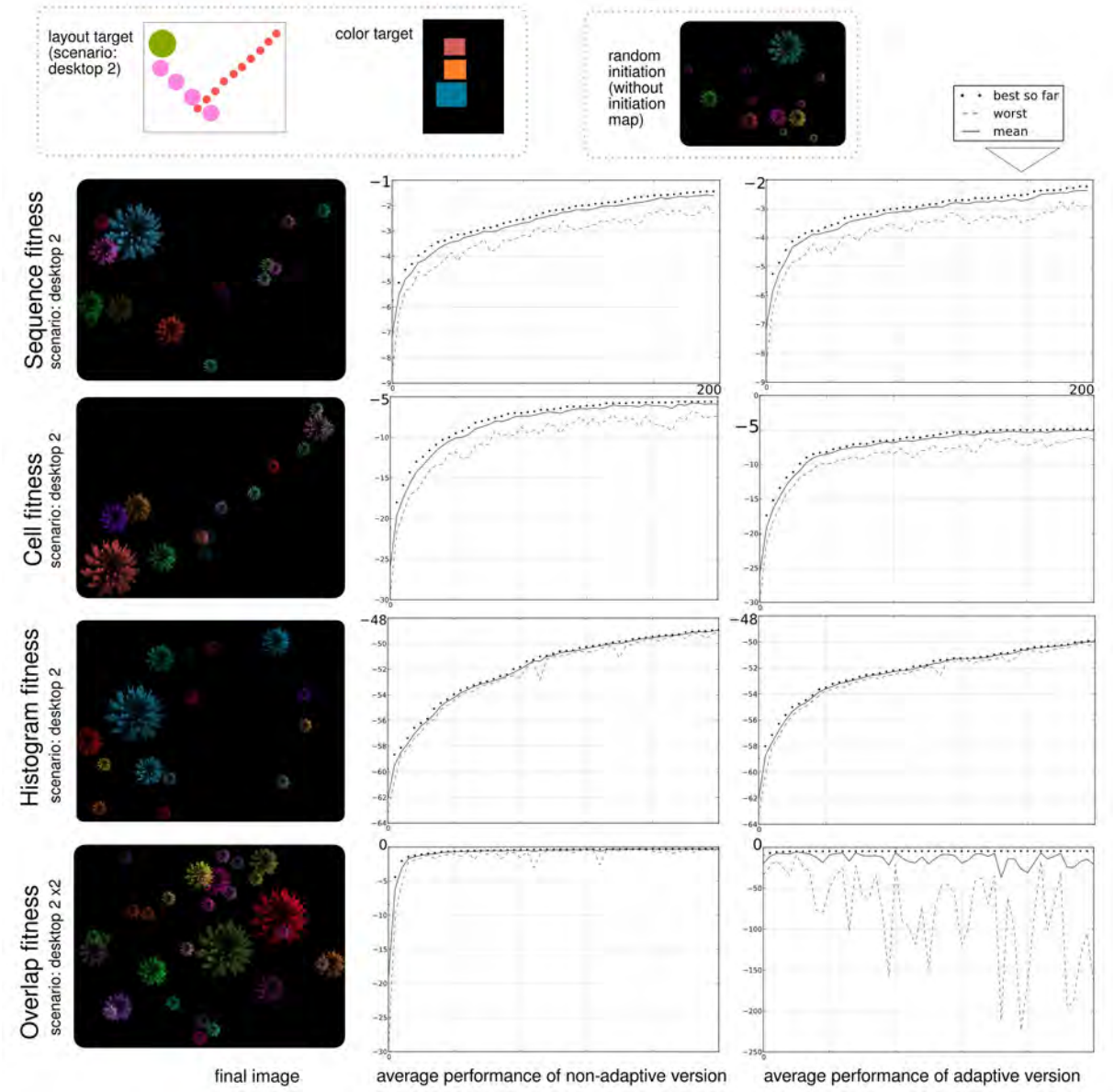


Figure 3. Test series for measuring the performance of objective functions in separate runs. (First case: 10 individuals 20 mutations)

Main test series are conducted for assessing the viability of the consecutive approach. Twelve real-world scenarios are developed, that are: producing decorative arrangements for coffee-cups, t-shirts and computer desktop wallpapers. For each scenario, target images are produced for color and layout. Six of the scenarios use exactly the same amount of DU's with their layout targets. The remainder are initiated with double this number, in order to test the tolerance for abundance cases. For each scenario ten tests are carried out with the same initial parameters. In order to save time, individuals are initiated with maps acquired from the layout targets. In this case while the aim of the histogram fitness is reaching from a random color distribution state to a higher fitness, layout fitnesses mostly have to preserve the initial levels. The "out + overlap" fitness has an additional task to stir up the arrangements especially in the beginning of the run.

Adaptive versions were initially preferred, which performed sufficiently well on regular scenarios [Fig.4]. But especially for the 'abundant' problems, success rates decreased down to around %50. When compared with the fixed parameter versions, adaptive versions yielded worse results in most of the scenarios tested so far. The main reason appears to be the poor adaptive performance of the "out + overlap" fitness. The second reason is, in our consecutive approach the main problem is not to supply variety but the contrary. In order to restrict variety, population size is kept small (i.e. 10), and this doesn't seem to provide a search space wide enough for the proliferation of parameters required by the adaptive process.

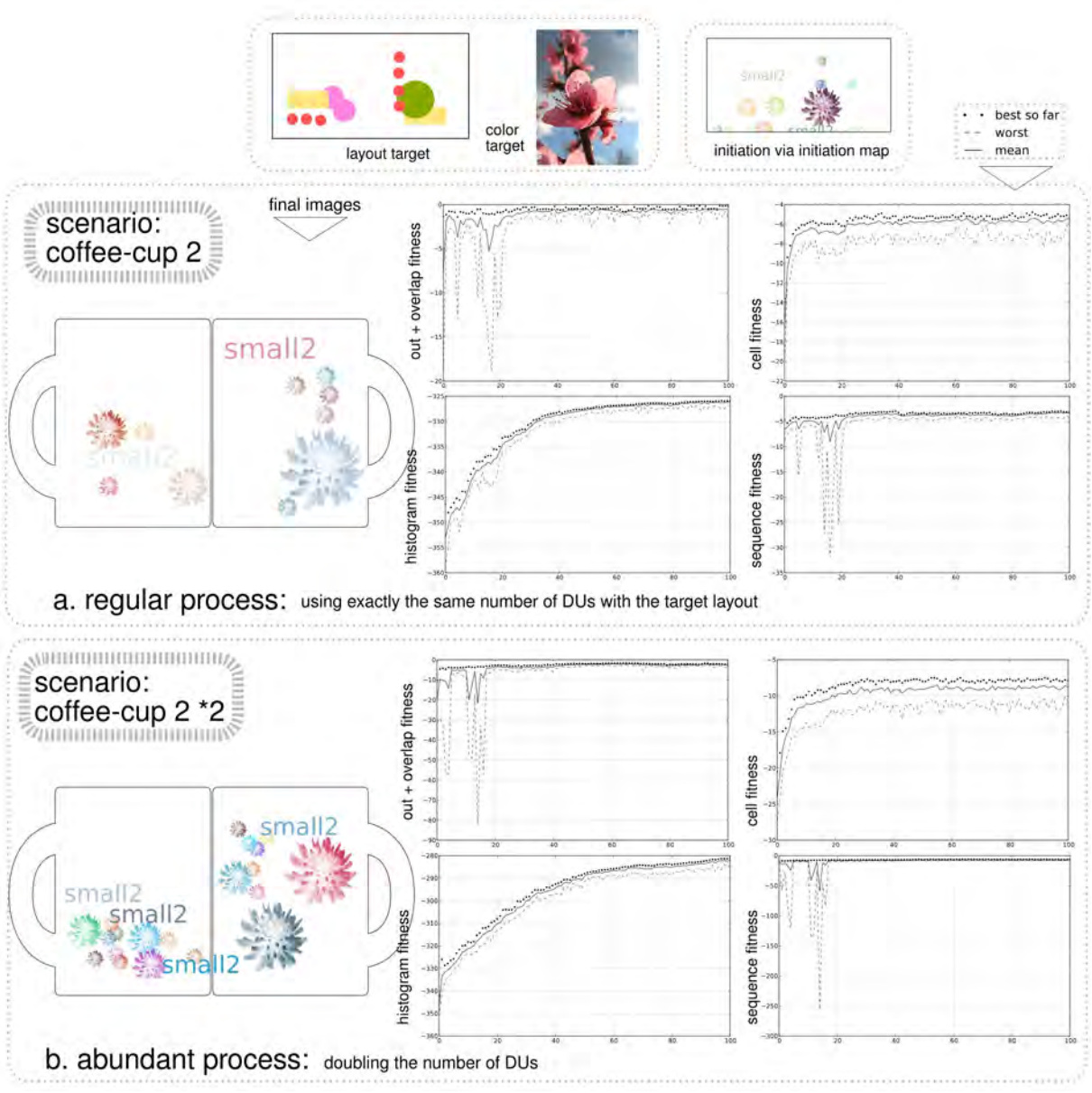


Figure 4. Comparison of regular and 'abundant' scenarios. In the abundant case number of DUs is doubled in the candidate image, but the targets are the same.

After adaptivity is disabled even the abundant scenarios reached desired levels on all fitnesses reliably. [Fig.5]. An important result is that, the stochastic nature of the evolutionary algorithms, when coupled with our tolerant objective functions, exhibits a flexible design approach: it is possible to produce a variety of distinct solutions,

which comprise different types and numbers of DUs, while still resembling the same targets.

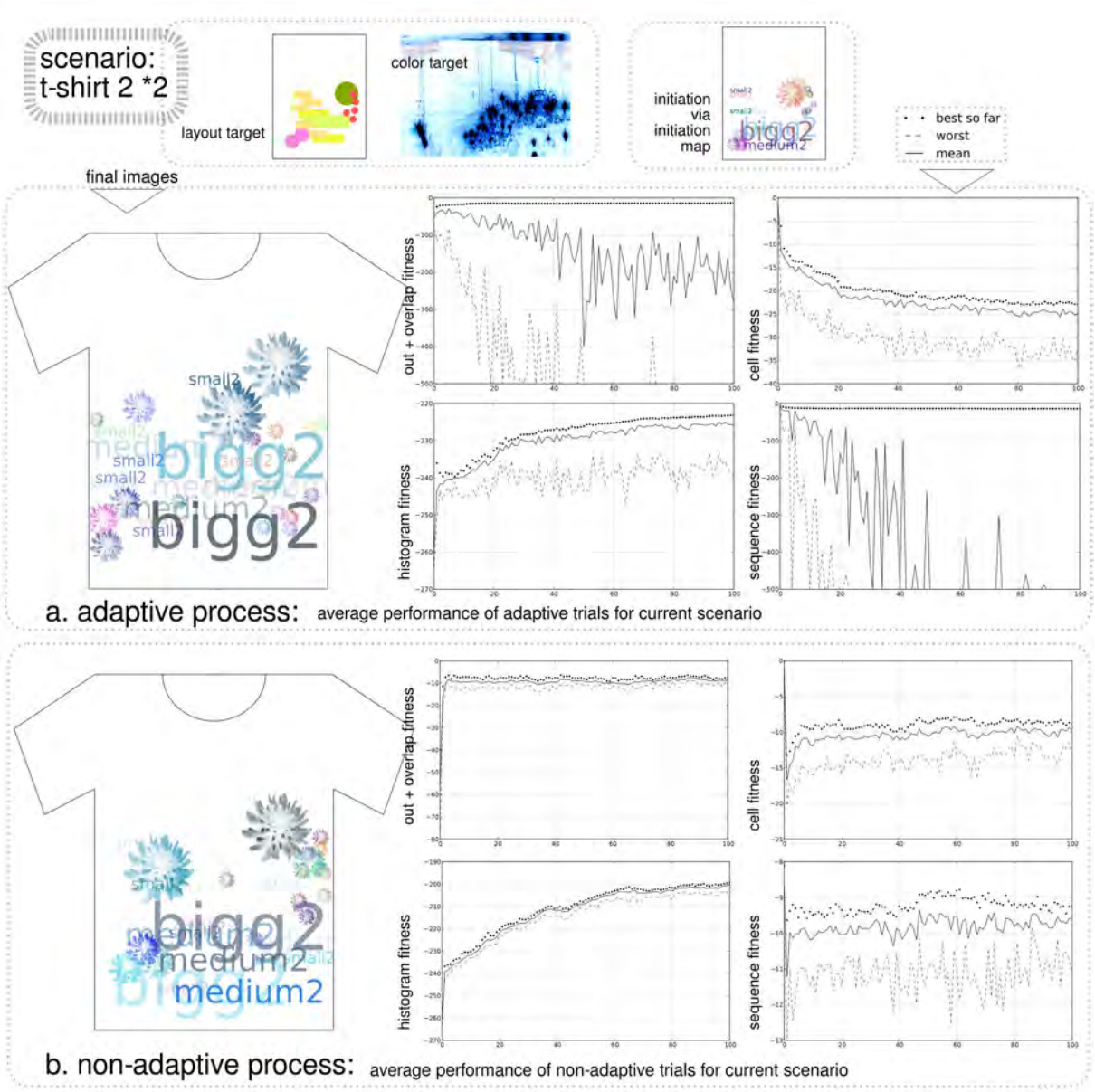


Figure 5. Comparison of average courses of adaptive and non-adaptive multi-objective runs. Non-adaptive version is clearly more reliable.

5. Conclusions

In this study a semi-automated approach for generating graphic arrangements is described, an image based interface is exemplified and a kind of multi objective evolutionary algorithm, the “consecutive approach” is successfully demonstrated. Our experiments exhibited the applicability of this approach to our task.

Regarding parameter control and adaptivity, current tests compared the adaptive and non-adaptive versions for only the best initial parameter settings, where non-adaptive versions performed slightly better. But no tests are carried out with other (worse)

parameter settings where adaptivity could have exhibited it's real expediency. More experimentation should be done with a more carefully designed adaptation process, in order to better asses the usefulness of adaptivity. Parameter control is important, because another result of this experiment series for us was that, parameter tuning for fixed values is really a time consuming task and would better be done by the process itself.

Acknowledgements

This study is supported by the International Research Fellowship Program of TÜBİTAK, The Scientific and Technological Research Council of Turkey.

References

[1] Lewis, M., Evolutionary Visual Art and Design. In Romero, J. and Machado, P. (Eds.), *The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music*, Springer-Verlag, 2008.
[2] Dillenburger, B., Braach, M. and Hovestadt, L., Building Design as an Individual Compromise between Qualities and Costs: A General Approach for Automated Building Generation under Permanent Cost and Quality Control. In Tidafi, T. and Dorta T. (Eds). *Joining Languages, Cultures And Visions: Caadfutures 2009*.
[3] Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., and Jain, R., Content-Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 12, December 2000.
[4] Datta, R., Joshi, D., Li, J., and Wang, J. Z. Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM Computing Surveys*, Vol. 40, No. 2, Article 5, April 2008.
[5] Veltkamp, R.C. and Tanase, M., Content-Based Image Retrieval Systems: A Survey. Department of Computing Science, Utrecht University, 2002 (A revised and extended version of technical report UU-CS-2000-34, October 2000).
[6] Smith, J. R. and Chang, S., Integrated Spatial and Feature Image Query. *Multimedia Systems* 7, 129–140, 1999.
[7] Eiben, A.E. and Smith, J.E., Introduction to Evolutionary Computing. Springer, New York, 2003.
[8] de Jong, K.A., Evolutionary Computation A Unified Approach. The MIT Press, Cambridge, Massachusetts London, England, 2006.
[9] Baluja, S., Pomerleau, D. and Jochem, T., Towards Automated Artificial Evolution for Computer-Generated Images. *Connection Science*, Vol. 6, Nos. 2 63 3, 1994.
[10] Wiens, A.L. and Ross, B.J., Gentropy: Evolutionary 2D Texture Generation. *Computers And Graphics*, 26(1): 75–88, 2002.
[11] Hewgill, A. and Ross, B.J., Procedural 3d Texture Synthesis Using Genetic Programming. *Computers & Graphics* 28, 569–584, 2004.
[12] Machado, P., Romero, J., Santos, M.L., Cardoso, A., and Manaris, B., Adaptive Critics for Evolutionary Artists. In Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D. and Squillero, G., (eds.), *Applications of Evolutionary Computing*, Evoworkshops2004, Vol. 3005 Of LNCS. Springer, pp437–446, Coimbra, Portugal, 2004.
[13] Machado, P., Romero, J., Cardoso, A. and Santos, A., Partially Interactive Evolutionary Artists. *New Generation Computing*, 23, pp143-155, Springer, 2005.
[14] Ross, B.J., Ralph, W., and Zong, H., Evolutionary Image Synthesis Using a Model of Aesthetics. In Yen, G.G., Lucas, S.M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.T., Coello, C.A.C. and Runarsson, T.P., (eds.), *Proceedings Of The 2006 IEEE Congress On Evolutionary Computation*, Vancouver, BC, Canada. IEEE Press, pp1087–1094, 2006
[15] Lok, S. and Feiner, S., A Survey of Automated Layout Techniques for Information

Presentations. Smartgraphics ’01 Hawthorne, NY USA, 2001.
[16] Lok, S. and Feiner, S., and Ngai, G., Evaluation of Visual Balance for Automated Layout. *LUI’04*, Madeira, Funchal, Portugal. 2004.
[17] Geigel, J. and Loui, A., Automatic Page Layout Using Genetic Algorithms for Electronic Albuming. *Proceedings of SPIE*, The International Society For Optical Engineering, Vol. 4311, pp. 79-90, 2001.
[18] Hanna, S., Automated Representation of Style by Feature Space Archetypes: Distinguishing Spatial Styles from Generative Rules. *International Journal Of Architectural Computing,* Issue 01, Volume 05, 2005.
[19] Hanna, S., Representing Style by Feature Space Archetypes, Description and Emulation of Spatial Styles in An Architectural Context. J.S. Gero (Ed.), *Design Computing and Cognition ’06*, 3–22, 2006.
[20] Hanna, S., Defining Implicit Objective Functions for Design Problems; *GECCO’07*, July 7–11, 2007.