An Auto Graphics Layout Design System using Genetic Programming

Aranya Walairacht*, Isara Burintramart**, Olarn Wongwirat**, and Shigeyuki Ohara*

* Department of Electronics, School of Engineering, Tokai University  
1117, Kitakaname, Hirasuka-shi, Kanagawa-ken, 259-1292, Japan  
Phone: 0463-58-1211 Ext. 4107, Fax: 0463-50-2031  
E-mail: 7jeed001@keyaki.cc.u-tokai.ac.jp, ohara@keyaki.cc.u-tokai.ac.jp

** Faculty of Information Technology and  
Research Center for Communications and Information Technology,  
King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand  
Phone: (+66 2) 326-8020 Ext. 527, Fax: (+66 2) 326-9074  
E-mail: isara@it.kmitl.ac.th, olarn@it.kmitl.ac.th

Abstract

This paper introduces a graphics layout design tool that has a capability to learn user preferences and to recognize designing consistencies. The introduced tool has ability to generate layout templates for selection according to user preferences learned from previous interactions. Additional objects suggestion to preserve consistencies can be made by the system. User preferences are in form of spatial relations and transitive relations. The system employs genetic programming to generate the layouts for selection and to be a learning mechanism. The system architecture, system learning, and experimental results are also included.

Keywords: Graphics Layout Design Tool, Learning Mechanism, Spatial Relations, Transitive Relations, and Genetic Programming

1. Introduction

In recent computer technologies, presentation works become increasingly effective. Presentation design tools are powerful and have the capability of helping users to design a wide range of presentations. These tools allow users to design new presentations and keep their preferable layouts in form of templates to be used to redesign a new presentation with same previous pattern. The area of our research work involves the implementation of layout design tool as part of the graphic presentation.

In this paper, we introduce a graphic design system that has ability to learn layout patterns that favorable to the user. The system uses genetic programming to generate and create the design layouts and to be a learning process. The system creates a list of layouts for users to choose after objects are selected. Users may identify only major objects, and then the system suggests additional objects and appropriate layouts, which are recognized from the previous selection learning. After the adjustments have made, the layout selection will be kept in a knowledge base for further references. The system can provide an automated layout suggestion and consistency verification benefits for both rapid layout designers and novice users.

The paper is organized into five sections. In section 2, we discuss our previous work and other related issues. An overview of the system and system modules are provided in section 3. Knowledge base, system learning and genetic programming are discussed in detail in section 4. Section 5 and 6 present the results of experimental from the application program and conclusions respectively.

2. Previous Work

Our work has been started from the study of object property and definition for 3D visual graphics [1]. We have defined spatial relations among the property of objects [2], e.g., fish must be in the water. In [3], additional positional relations (e.g., left-right-on-under) and object rules have been introduced and included for the definitions. We carry on our works to implement the tool for graphic layout design by using genetic programming. There are several issues relating the graphic presentation approach and our work.

Clay and Wilhelms [4] introduced an object-placement system called Put, which uses linguistic input and interactive manipulation to identify spatial relationships between objects. Their work focuses on cognitive linguistics in the usage of natural language instead of typical manual tools in
controlling graphical applications. Another system related with spatial relations is NALIG (Natural Language driven Image Generator) by Giunchiglia et al. [5], which is a CAD tool initially intended for interior design. In a work of Igarashi et al. [6], Link Model parsing strategy is introduced using genetic algorithm in its parameter tuning. Link Model targets on the ambiguous layout interpretation; on the other hand, our system targets on the suggestions made based on prior knowledge of user preferences. GALAPAGOS (Genetic Algorithm And Presentation-Assisted Graphic Object layout System), an interactive layout system proposed by Masui [7], uses genetic algorithms in user preferences justifications to solve directed graph layout problems. In addition, Graf [8], [9] introduced a constraint-based multimedia layout manager, LayLab, using constraint-processing techniques, i.e., semantic-pragmatic constraints, spatial constraints, dynamic constraints, and temporal constraints. Together with LayLab, a graphical editor called InLay [10] is used to automatically beautify visual programming displays.

3. System Architecture

3.1. System Overview

In graphic layout design for presentation, one of the most important tasks in computer is the ability to generate several possible combinations of the graphic layout for user selection. To generate the combinations, most computers use numerical search and mapping algorithm for computation in general. Instead of other numerical search algorithms, we use genetic programming [11] as a learning algorithm for the following reasons:

- Genetic programming produces generalized results envisioned by human.
- In our case, as object positions are randomized, the number of result is in explosive combination. Genetic programming helps reducing the number by eliminating uncompetitive ones.
- The benefit over genetic algorithms is that genetic programming allows the length of chromosome to be flexible.

In the proposed system, we consider layout selection of users as a feedback for learning process. Selected layouts are converted into chromosomes, assigned fitness values, and stored in the knowledge base. These layout chromosomes are then processed using genetic operators to make new generations, which generate better and more consistent suggestions. The more generations of chromosome, the better suggestions the system makes.

3.2. System Modules

As shown in Fig. 1, the proposed system composes of three major components: user interface, knowledge base, and genetic programming. User interface composes of two main modules, which are object selection and layout selection. The object selection functions as a tool for users to identify objects (images) to be included in the designing layout. The layout selection functions as a feedback for machine learning mechanism. After the layout is selected, the information relating the objects will be stored in knowledge base for arrangement and comparison. The system uses genetic programming to generate a new set of selections in case that the user does not satisfy with the selection. These processes are done repeatedly until the user makes selection. Finally, the selected layout is stored back to the knowledge base for further references.

![Fig. 1 System Modules](image)

4. Computation Mechanism

As mentioned in the previous section, the selected data will be stored in the knowledge base and will then be used for learning mechanism by genetic programming. This section explores the functions of knowledge base and genetic programming algorithm.

4.1. Knowledge Base

Knowledge base in the system includes two types of data, which are basic data elements (e.g., images) and information extracted from user-interactions. The information supplies knowledge used in learning process and is changed adaptively without direct interventions from users. Layout selections made by users are merely a trigger to the learning process. Image database stores image objects used in user-interfaces. These images are mapped to selected objects for visualization.
purpose only. The system recognizes some object properties, e.g., size and position, from mapped object; neither from image, nor from objects alone. Since images are categorized into classes and subclasses, and environment properties are defined in class level, therefore, environment properties of selected objects refer to class properties to that mapped images belong.

4.2. System Learning

As discussed in the previous sections, genetic programming is used for learning mechanism. The basic element of genetic programming is the chromosome structure. A chromosome is generated randomly by extracting relative positions out of the coordinates from the selected objects in a layout. Thus, there is one chromosome representing one layout template.

A chromosome consists of two types of gene, i.e., terminal gene and function gene. One terminal gene represents one selected object. It contains object properties, including the object class mapped by selected images. A function gene contains a relationship between two objects, or a relationship between one object and a set of related objects. These relationships are relative positions, $\delta x$ and $\delta y$, projected on x- and y-axes (including $\delta z$ on z-axes in case of 3D layouts).

A chromosome can be written in context-free grammar ($G$) as

$$G = ([F], [T], P, F).$$

Where, $F$ is function gene, $T$ is terminal gene, $P \subseteq (F \cup T)^+$, and

$$F \rightarrow FF|FT|TF|TT.$$  

Since chromosomes are only logical metaphors, in programming, they have to be structured in form of binary trees. In that case, a terminal gene is called a terminal node, while a function gene is called a function node. In addition, the root node of the tree (a function node) also contains absolute coordinate for the entire layout. Chromosome structures in Backus-Naur Form (BNF) and in binary tree are shown in Fig. 2 (a) and Fig. 2 (b) respectively.

Figure 2 Chromosome Structures

4.3. Genetic Programming Algorithms

The layout templates and their chromosomes are randomly generated until the population size is reached. This operation is called population process. The crossover will be performed when none of the displayed layout templates is selected to generate another generation. Our crossover takes the entire population to be crossed. This is based on an assumption that users always select only their preferred layouts; therefore, unselected layouts are supposed to be discarded by passing through crossover and mutation processes. To be more specific, unselected layouts should never appear in any set of selection again.

Prior to the crossover, chromosomes are selected to form crossover-pairs. This is done in the selection process, in which best-believed chromosomes are matched using fitness value as a criterion. Normally, best parents are expected to produce best offspring, even with better characteristics than of the parents. However, since our application relies on user preferences, which cannot be standardized, and the meaning of "fitness" in one situation may be different from another, therefore we do not deliberately match best chromosomes together. In stead, we use roulette wheel sampling [12] in our programming algorithm, in order to allow lesser-fit chromosomes to have a chance to breed with best-fit chromosomes. Note that roulette wheel sampling gives probabilities to chromosomes according to their fitness values as a ratio against overall. Since we expect selection varieties, we give higher priority to the offspring than to their parents if they have equal fitness values.

In crossover process, one node from each paired-chromosome is arbitrary selected, which can be either a leaf node or a node with sub-tree, but not a root node. An example of crossover is shown in Fig. 3.

![Example of crossover](image-url)
Fig. 3 Crossover Operator (a) Before, and (b) After

In order to allow some good characteristics appeared in chromosomes with low fitness value to survive from crossover processes, after repeating a number of cross-over generations, a mutation process is triggered. We, again, use roulette wheel sampling to select a chromosome to be mutated. For this purpose, the roulette gives more chances to the less-fit chromosomes. One node is arbitrary selected from the entire population as a source chromosome. Then, the node is copied to replace one randomized node in the mutating chromosome. There is neither offspring reproduced in the mutation process, nor changes in the number of population.

Fitness value of each chromosome is reassigned in every crossover, mutation, and layout selection processes. This is called an evaluation process. When a chromosome is first generated, it is assigned a randomized fitness value ranged from 0.1 to 0.9. When it is cross-over or mutated, the fitness value is averaged with the corresponding chromosome. In a crossover, both chromosomes have their fitness values changed; while in a mutation, only the mutated chromosome has its fitness value changed. Finally, as the system is selection-driven, when a chromosome is selected by the user, we consider it is the fittest one. As the highest fitness value is 1, therefore, we reassign its fitness value using

\[ f_{(i,t+1)} = \frac{f_{(i,t)} + 1}{2} \]

where, \( f \) is fitness value of chromosome \( i \) at generation \( t \).

5. Experimental Result

We use Java programming language in developing a 2D prototype employing our concepts. We use a crossover rate of 0.9 and a mutation rate of 0.01 at population size 30 as control parameters [12].

Mean fitness values versus number of generations per one learning session are tested on 3, 5, 10, and 20 objects. Marked circle, square, diamond, and triangle respectively, Fig. 4 shows that most of the cases reach the acceptable range of fitness (0.9-1.0) at the early of 10s generations.

[Graph showing Per-Session Learning Test Results]

Fig. 4 Per-Session Learning Test Results

[Graph showing Cumulative Learning Test Results]

Fig. 5 Cumulative Learning Test Results

Fig. 5 shows a comparison between per-learning session (square-marked line) and cumulative learning sessions (round-marked line) tested on five objects. The fluctuating nature of cumulative learning can be explained as an effect from object reselections made by the user. Since users may select objects differently in each session, the evaluation may be interrupted (dropped points on the line). However, the result suggests that although fluctuated, the cumulative learning improves its fitness values gradually shown by dotted trend line.

A screen snapshot of the program is shown in Fig. 6. Left panel of the screen lists available objects, while bottom panel displays the layout templates generated for selection. Users may make
modifications by relocating objects on the screen using mice.

Fig. 6 Prototype of the Automatic Layout Arrangement System

6. Conclusions

We have developed a prototype system that automates and assists users in designing general 2D presentation layouts. The system uses genetic programming for its learning mechanism. User preferences are recognized from user selections by parsing spatial relationships and transitive relationships of objects in the selected layouts. The information is stored in its knowledgebase and use to generate future suggestions. Apart from layout pattern, the system has ability to suggest sizes, relative positioning, and additional objects learned that could consistently appear in the scene. In addition, its graphical user interface allows users to preview and make selection or modification easily.

Our future work will be on performance improvements, optimization of learning process especially in cumulative learning, and additional parameters such as colorization and design categorization.

7. References