A Layout Method Using Fuzzy Constraint Satisfaction for Art Design

Takuto Yanagida*1

Abstract – Art design is made of a set of rules (constraints). In a design of posters, for example, many constraints are known: A title should be more appealing than other elements; and the alignment of and the distance between a text and a figure should be arranged based on their reference relation. The author intends to automate layout in art design, which consists of many constraints like above. In this paper, as a first step, the author proposes a system that solves positional relation among layout elements using the framework of fuzzy constraint satisfaction.

Keywords : fuzzy constraint satisfaction problem (FCSP), art design, layout, and positional relation

1. Introduction

In this paper, we use *art design* to mean both processes and results of designing *two-dimensional* embodiments of information. There are a lot of examples around us such as, especially, posters, flyers, and digital information panels (including data broadcasting on TVs). Art designers design these things based on their aesthetic feelings, some designing rules, and of course, requirements of their clients. We think that what they actually do could be summarized as activities of deciding what kind of information should be selected, where it should be put, and which way should be taken to describe it.

For ordinary people (non-designers), regardless of whether they have artistic sense or not, the art design is difficult because there are many rules (*constraints*) should be followed[1]. For example, a title should be more appealing than other elements; and the alignment of and the distance between a text and a figure should be arranged based on their reference relation. One font is more suitable for title or headings than other fonts; and the brightness of contents and their background should vary in a certain level for readers. Nowadays, we can easily use a lot of professional tools and application software and have a lot of chance to design. However, it does not necessarily lead to well-designed materials.

We need to handle constraints (rules) posed on design elements (items) on a poster when designing. In the field of artificial intelligent, there is a framework of *constraint satisfaction problems* (CSPs) and its variations such as MAX CSPs, weighted CSPs, and fuzzy CSPs[2]. In these frameworks, problems are formulated using variables with each domain and constrains between the variables. So far, a lot of solvers (algorithms for solving CSPs) have been proposed, and they can be used for obtaining solutions of problems. Using these frameworks, we can formulate problems in the real world.

What we are interested in and intend to realize finally is a system that automates art design and supports normal people when designing. We expect that the frameworks of CSPs can be a useful tool for developing such intelligent systems. Actually, we have already developed a related system as previous work[3],[4], which handles automatic layout of widget, parts of graphical user interfaces. Basically, the idea that CSPs are used for layout is not unique to us, and many textbooks and papers mention this idea. However, there are not so many studies using directly the frameworks, and we have not found yet any tools for that purpose in the market.

In this paper, we propose, as a first step, a formulation of positional relations among layout items and a system solving layout problems using the framework of fuzzy constraint satisfaction.

2. Related Work

The automation of generating layouts is one of the most important challenges[5] in the both fields of human-computer interaction and artificial intelligence. Schrier et al. proposed a grid-based document design system for generating layouts of aggregated contents on the Internet[6],[7]. Their work aims at realizing adaptive newspaper-like layouts, and it might be useful for electronic publishing. Lok et al. proposed WeightMaps as a method for evaluating layouts based on the concept of visual weight and visual balance[8]. Evaluating what is good layouts and how good they are is a problem when developing automatic layout systems.

Layout problems arise when graphical user interfaces (GUIs) are automatically generated because widgets (GUI elements) should be placed appropriately on a

*1: Research Institute of Electronics, Shizuoka University, takty@rie.shizuoka.ac.jp

window. Gajos et al. proposed SUPPLE system, which generates GUIs using a combination search algorithm[9]. As we mention in the previous section, we also proposed a system for automatic GUI generation, which utilizes the framework of FCSPs[4],[10]. In addition, for evaluating results of the generation, layout appropriateness[11] is proposed as a metric for evaluating widget layouts based on both sequences of widget-level actions which users perform and how frequently each sequence occurs.

3. Fuzzy Constraint Satisfaction

A CSP is a generic term for search problems to find combinations of values that satisfy the given constraints. It consists of the following components: a set of *variables* $X = \{x_1, \ldots, x_n\}$, a set of finite *domains* $D = \{D_1, \ldots, D_n\}$, and a set of *constraints* $C = \{c_1, \ldots, c_r\}$. Each variable x_i is supposed to take a value from the domain D_i. A constraint c_k denotes a relation R_k on a subset S_k of X, i.e.,

$$R_k \subseteq D_{k_1} \times \cdots \times D_{k_w} \text{ for } S_k = \{x_{k_1}, \ldots, x_{k_w}\}. \quad (1)$$

It represents permissible combinations of values to each variable of S_k. S_k is called the *scope* of R_k. If $w = 1$, 2, or 3, the constraint is called a unary, binary, or ternary constraint respectively. An *assignment* v to a scope S_k is denoted by $v[S_k] \in D_{k_1} \times \cdots \times D_{k_w}$. If $v[S_k] \in R_k$, then v *satisfies* the constraint c_k; otherwise, it *violates* c_k. To find a solution to a CSP involves the search for an assignment $v[X]$ that satisfies all the constraints of C.

It is sometimes hard to use classical CSPs for formulating real-world problems. Thus, an FCSP introduces a soft constraint called a *fuzzy constraint*, which is not necessarily satisfied completely, but instead, its *degree of satisfaction* (satisfaction degree) should be considered as follows. In an FCSP, a constraint c_k denotes a fuzzy relation with its membership function defined by

$$\mu R_k : \prod_{x_i \in S_k} D_i \to [0, 1]. \quad (2)$$

In other words, the membership value is defined by an assignment $v[S_k]$ to the scope S_k. This value is called the *satisfaction degree* of the fuzzy constraint.

Since an FCSP requires the satisfaction of the fuzzy conjunction of all fuzzy constraints, the satisfaction degree of the whole FCSP is defined as the smallest satisfaction degree of all the constraints as follows:

$$C_{\min}(v) = \min_{1 \leq k \leq r} \Big(\mu R_k \big(v[S_k] \big) \Big). \quad (3)$$

If $C_{\min}(v) > 0$, then an assignment v is called a *solution* of the FCSP, and a solution that maximizes $C_{\min}(v)$ is called an *optimal* solution. Therefore, an FCSP is regarded as an optimization problem that requires finding the assignment that maximizes the smallest satisfaction degree of the constraints.

We introduce the framework of FCSP, formulate layout problems as FCSPs, and represent the rigidness of rules with satisfaction degrees of fuzzy constraints. We use binary fuzzy constraints for expressing positional relations between arbitrary two layout items, or rectangles. Fuzzy constraints enable us to represent naturally the gradual layout rules. After the formulation, we apply one of the solvers for FCSPs, *fuzzy forward checking*, which is an extension of the forward checking for crisp FCSPs.

4. Formulation

In this paper, we handle the positional relations between layout items, as the first phase of the system for art design. Then, we consider the problem in which the arbitrary numbers of rectangles with specific sizes are put on a grid satisfying constraints between the rectangles. When the number of the rectangles is n, the size of the grid is $n \times n$, and then, the maximum size, width and height of each rectangle is n and n. You can assign constraints of positional relations such as *put this one in the left of another one*, *put this one above another one*, and *align these ones horizontally*.

Each variable x_i corresponds to each rectangle, and its domain D_i contains the following values (tuples) t:

$$t = \langle left, top, width, height \rangle, \quad (4)$$

where $left$ and top denote a coordinate of a upper left corner, and $width$ and $height$ denote literally a size. Each rectangle dominates some cells on a grid, and then, the sizes of the domains are not large comparing with when it is put freely on a plane.

Each constraint c has multiple sets of conditions of relating variables, and a condition set has its strength (Table 1). We provide five conditions, which are boolean

Table 1 Strength of condition sets

Strength	Satisfaction Degree
WILL	0.0
MUST	0.2
NEED_TO	0.4
HAD_BETTER	0.6
SHOULD	0.8

Table 2 Positional condtions[1]

Name	Condition
LEFT	$right_2 < left_1$
RIGHT	$right_1 < left_2$
TOP	$bottom_2 < top_1$
BOTTOM	$bottom_1 < top_2$
HORIZONTAL	$(top_1 \leqq top_2$ and $bottom_1 \geqq bottom_2)$ or $(top_2 \leqq top_1$ and $bottom_2 \geqq bottom_1)$
VERTICAL	$(left_1 \leqq left_2$ and $right_1 \geqq right_2)$ or $(left_2 \leqq left_1$ and $right_2 \geqq right_1)$
ADJACENT	$((right_1 + 1 = left_2$ or $right_2 + 1 = left_1)$ and $((top_1 \leqq top_2$ and $bottom_1 \geqq top_2)$ or $(top_2 \leqq top_1$ and $bottom_2 \geqq top_1)))$ or $((bottom_1 + 1 = top_2$ or $bottom_2 + 1 = top_1)$ and $((left_1 \leqq left_2$ and $right_1 \geqq left_2)$ or $(left_2 \leqq left_1$ and $right_2 \geqq left_1)))$
NOWRAP	$right_1 < left_2$ or $right_2 < left_1$ or $bottom_1 < top_2$ or $bottom_2 < top_1$

[1] Conditions between item 1: $t_1 = \langle left_1, top_1, width_1, height_1 \rangle$ and item 2: $t_2 = \langle left_2, top_2, width_2, height_2 \rangle$ ($right = left + width - 1$, and $bottom = top + height - 1$).

functions taking two values from the two variables as arguments (Table 2). Each condition set contains multiple conditions and the satisfaction degree will be 1, the maximum, if all the conditions are satisfied, otherwise, the degree will decrease to the value defined by the strength. If one condition in a condition set does not satisfied, then the satisfaction degree of the set becomes the value of the strength of the set. The strength `WILL` is the strongest because if a condition set assigned `WILL` violates, its satisfaction degree reduces to 0, the minimum.

5. Implementation

We have implemented a sample system as an editor of layout items and positional relations among them (Fig. 1). For utilizing FCSP framework, we adopt a library[12] developed for our other studies. The system consists of two tabs: 'Structure Editor' tab (Fig. 1) and 'Automatic Layout' tab (Fig. 2). In the first tab, you can add layout items on a pane and make positional relations between the items with drag-and-drop operations. When a relation is double-clicked, the relation property dialog box (Fig. 3) pops up. In this dialog box, you can configure both condition sets and the strength of the sets.

When you click on the 'Do Layout' button on the layout tab, the following process occurs in the background:

1. Variables that correspond to each rectangle on the pane are generated.
2. Domains for the variables are generated:
   a. Minimum sizes of the items are calculated based on the number of connected relations.
   b. Clipped areas, where items can be put, are calculated by the relations.
   c. Maximum sizes of the items (one by one or the size of the whole grid) are roughly calculated.
3. Automatically `NOWRAP` relations are assigned each pair of the rectangles, and constraints corresponding to relations are generated.
4. Fuzzy forward checking solver is applied to the problem made in the above steps.

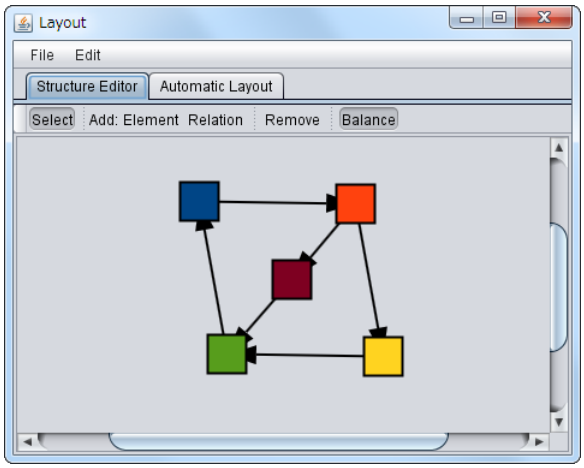

Fig. 1 Screenshot of 'Structure Editor' tab. Users can add and remove elements (rectangles) and relations (arrows), and improve their appearance by 'Balance' button.

In Fig. 1, relations of a set of `adjacent` and one of `left`, `right`, `top`, and `bottom` conditions are assigned between outside rectangles, and relations of a set of `adjacent` and `bottom` conditions are assigned between the center rectangle and its upside and downside rectangles. Then, the result of the layout process is Fig. 2. In this case, the relations make a reference cycle, but we do not need to handle it on our formulation.

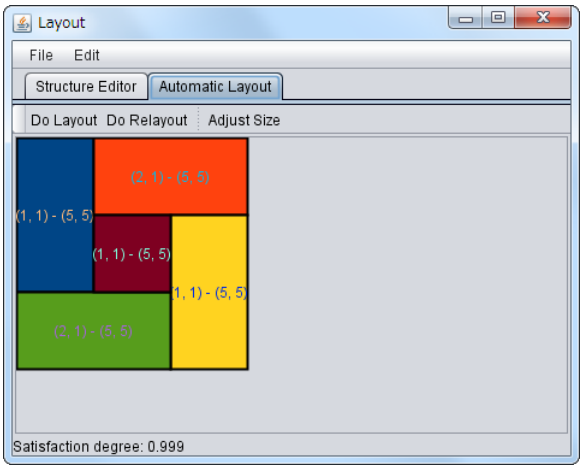

Fig. 2 Screenshot of 'Automatic Layout' tab and a result of the automatic layout process. The numbers on a rectangle mean its minimum and maximum size.

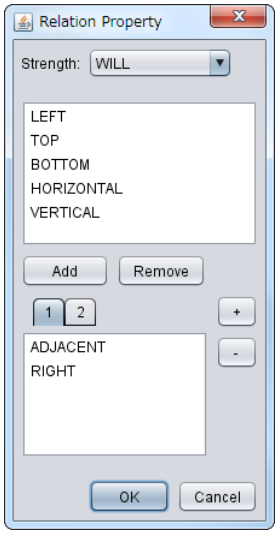

Fig. 3 Screenshot of the relation property dialog box. Users can add and remove condition sets and select their strength.

6. Conclusion and Future Work

In this paper, we mentioned a method for representing layout using FCSPs and showed a primitive implementation with a user interface. As the first step of the trial of developing automatic system for art design, we proposed the positional layout system. The current system is not so novel one, but it would be utilized as a base of future system for our final purpose.

We are going to extend the implementation for our final goal, which would be composed with a few phases, and the first phase is now implemented for determining positional relations among layout items. In the next phase, actual sizes of the items will be decided in accordance with the contents of the elements. Moreover, in the third phase, aesthetic properties of the items will be decided, and final solution, an art design will be obtained. We need to consider in those phases whether we can still use FCSP framework or we should take another method.

Reference

[1] Tondreau, B.: *Layout Essentials: 100 Design Principles for Using Grids*, Rockport Publishers (2009).

[2] Ruttkay, Z.: Fuzzy Constraint Satisfaction, *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, Vol. 2, Orlando, FL, USA, IEEE, pp. 1263–1268 (1994).

[3] Yanagida, T., Sudo, Y. and Nonaka, H.: Flexible Widget Layout Based on Fuzzy Constraint Satisfaction, *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, Vol. 20, No. 6, pp. 840–849 (2008). (in Japanese).

[4] Yanagida, T. and Nonaka, H.: Flexible Widget Layout Formulated as Fuzzy Constraint Satisfaction Problem, *Proceedings of the 1st KES International Symposium on Intelligent Decision Technologies (KES IDT 2009), New Advances in Intelligent Decision Technologies* (Nakamatsu, K., Phillips-Wren, G., Jain, L. C. and Howlett, R. J., eds.), Himeji, Japan, Springer, pp. 73–83 (2009).

[5] Lok, S. and Feiner, S.: A Survey of Automated Layout Techniques for Information Presentations, *Proceedings of the 1st International Symposium on Smart Graphics*, Hawthorne, NY, USA, ACM, pp. 61–68 (2001).

[6] Schrier, E., Dontcheva, M., Jacobs, C., Wade, G. and Salesin, D.: Adaptive layout for dynamically aggregated documents, *Proceedings of the 13th international conference on Intelligent user interfaces (IUI 2008)*, Gran Canaria, Spain, ACM, pp. 99–108 (2008).

[7] Jacobs, C., Li, W., Schrier, E., Bargeron, D. and Salesin, D.: Adaptive Grid-Based Document Layout, *Proceedings of the 30th international conference on computer graphics and interactive techniques (SIGGRAPH 2003)*, San Diego, CA, USA, ACM, pp. 838–847 (2003).

[8] Lok, S., Feiner, S. and Ngai, G.: Evaluation of Visual Balance for Automated Layout, *Proceedings of the 2004 International Conference on Intelligent User Interfaces (IUI 2004)* (2004).

[9] Gajos, K. and Weld, D. S.: SUPPLE: automatically generating user interfaces, *Proceedings of the 9th international conference on Intelligent user interfaces (IUI '04)*, Funchal, Madeira, Portugal, ACM, pp. 93–100 (2004).

[10] Yanagida, T., Nonaka, H. and Kurihara, M.: Personalizing Graphical User Interfaces on Flexible Widget Layout, *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS 2009)*, Pittsburgh, PA, USA, ACM, pp. 255–264 (2009).

[11] Sears, A.: Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout, *IEEE Transactions on Software Engineering*, Vol. 19, No. 7, pp. 707–719 (1993).

[12] Yanagida, T. and Sudo, Y.: Stlics: a Java library for fuzzy constraint satisfaction problems (2009). Available at `http://kussharo.complex.eng.hokudai.ac.jp/~takty/interest/stlics.en.html`.