A tool for analyzing and visualizing constraint satisfaction problems

Takuto Yanagida[†], Masahito Kurihara[‡], and Hidetoshi Nonaka[†]

Graduate School of Information Science and Technology
Hokkaido University, Sapporo 060-0814, Japan
E-mail: [†]{takty, nonaka}@main.ist.hokudai.ac.jp, [‡]kurihara@ist.hokudai.ac.jp

Abstract Constraint satisfaction problems (CSPs) and one of its extensions fuzzy CSPs (FCSPs) are simple models for formulating problems that exist in the real world. There are a lot of general-purpose solvers for FCSPs, and once we formulate a problem as an FCSP and then can have one of the solvers solve it. This formulation is however another problem because it is a process that entails highly abstracting the problem, it is not often determined uniquely, and it governs the efficiency of solving the problem. The authors consider that it improves this situation to increasing the visibility of solvers' behavior in order to adjusting models developed once. In this paper, the authors show their ongoing work for developing a tool for analyzing and visualizing FCSPs, which provides two- and three-dimensional views of FCSPs and shows the behavior of solvers with animations.

1. INTRODUCTION

Constraint satisfaction problems (CSPs) are a simple model for formulating problems that exist in the real world, and they have been studied in the context of the artificial intelligence. However, there are still a lot of problems that classical, rigid (crisp) CSPs are hard to handle. Therefore, *fuzzy* CSPs (FCSPs) are proposed as an extension of the crisp CSP, where fuzzy relations represent constraints [1]. It accommodates suboptimal solutions for providing useful information for handling the problems.

Since there are a lot of general-purpose solvers for FCSPs, once we formulate a problem as an FCSP, and then we can have one of the solvers solve it. This formulation is however another problem because it is a process that entails highly abstracting the problem, it is not often determined uniquely, and it governs the efficiency of solving the problem. That is, the formulation is a problem left to humans, and can be one of the research topics including ours [2].

We consider that it improves the situation of formulations to increasing the visibility of solvers' behavior in order to adjusting models developed once. When we utilized the framework of FCSPs as a tool for solving a problem, we were often confused because we were not able to comprehend behavior of solvers. Why does not a solver output a solution? Why do not they stop? What is the bottleneck of the modeled problem?

We are now investigating related work for improving this situation. Sadaoui et al. have proposed a tool [3], which enables us to generate and solving CSPs. JCLEditor is another graphical tool for developing and solving CSPs and soft CSPs including FCSPs [4]. However, both of them do not provide any functions for helping comprehend the behavior of solvers.

In this paper, we show our ongoing work for developing a tool for analyzing and visualizing FCSPs (Fig. 1). This tool provides the two- and three-dimensional views of constraint graphs (an representation of FCSPs), and it enables users to understand the behavior of solvers with animation intuitively, detect the looping behavior of solvers, and debug in similar way of integrated development environments (IDEs) for programming languages.

2. CONSTRAINT SATISFACTION PROBLEMS

An FCSP consists of the following components: a finite set of variables $X = \{x_i\}_{i=1}^{m}$, domains $D = \{D_i\}_{i=1}^{m}$ associated with the each variable, and constraints $C = \{c_k\}_{k=1}^{r}$ among the variables. Constraint c_k denotes a fuzzy relation μR_k on a subset of the variables, which is called the *scope* of μR_k. A constraint has its *membership function*, the membership value of which is computed by an assignment to the variables in the scope of the constraint. This value is called *the degree of satisfaction*, and it expresses how the assignments satisfy the constraint. The degree of satisfaction of the whole FCSP is defined as the minimum of the degrees. To solve an FCSP is equivalent to find the assignment accompanied by the best satisfaction degree of the problem.

In general, the structure of FCSPs can be represented by a constraint graph, where nodes and edges correspond to variables and constraints. Another form of the graph is a dual constraint graph, which represents constraints as nodes and nonempty sets of

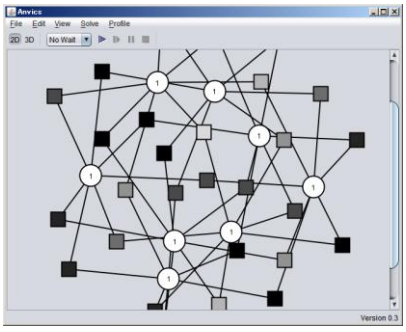

Fig. 1 A screenshot of the main window of the tool.

variables as edges. In this paper instead, we represent FCSPs with variable nodes (rounds), constraint nodes (squares), and edges of their connections (Fig. 1, 2). It can express states of variables and constraints at once by changing the appearance of the nodes.

3. A TOOL FOR FCSPS

We implemented a tool in Java based on our CSP library Stlics (Fig. 1). The tool offers the following 10 solvers implemented in the library: Breakout [5], SRS 3 revised for crisp problems, the forward checking [6], GENET, and the local changes (for crisp problems), and the flexible local changes [7], the fuzzy Breakout, the fuzzy forward checking, fuzzy GENET [8], and SRS 3 [9] (for fuzzy problems). Users can load arbitrary problems that the Stlics library outputs, apply above solvers, and obtain a solution.

The tool provides two types of view of constraint graphs: the two-dimensional (2D) view and three-dimensional (3D) view (Fig. 2). In order to make appropriate layouts of graphs, we adopted the modified spring model [10] and its extension for the 3D view. The equations for calculating the forces of springs are as follows:

$$f_a(d) = c_a \frac{d^2}{k}\ ,\ f_r(d) = -c_r \frac{k^2}{d},$$

where f_a and f_r means attractive and repulsive forces among nodes, d and k are the actual and the ideal distance between two nodes, and c_a and c_r are constants that decides the strength of those forces respectively. f_a takes effect between connected nodes, variable and constraint nodes, and f_r does among other nodes. In the 3D view, nodes are mapped on the surface of a virtual sphere. Users can customize the constants of the forces and the radius of the sphere, and can rotate and scale the views.

Showing the behavior of solvers is another important function provided by our tool, which flashes each variable node when a solver assign values to the corresponding variable, and it can change the color of variable nodes and constraint nodes in response to the change of assignment counts and the degree of satisfaction respectively. To analyze FCSPs, users can have the tool detect loops of assignments, which is a phenomenon that a solver iterates trying to assign the same values to the same variables. In addition, users can put break points on any variables to pause solvers when they assign values to the variables, and can execute solvers gradually, which is called *step over* in IDEs.

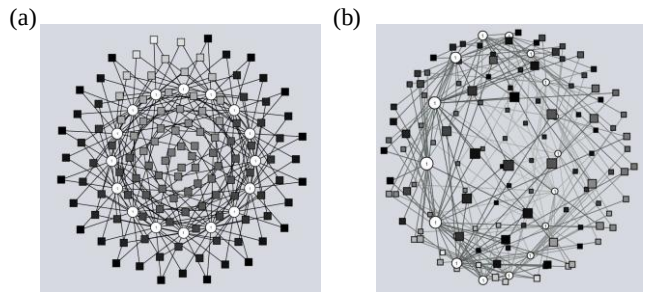

Fig. 2 the two-dimensional (2D) view (a) and the three-dimensional (3D) view (b) of a constraint graph.

4. CONCLUSION

We showed our ongoing work to develop a tool for analyzing and visualizing FCSPs, which enables users who utilize FCSPs as a tool to comprehend the structure of FCSPs and the behavior of solvers intuitively. The features of our tool are the 3D view of constraint graphs and the loop detection function. In addition, users can debug FCSP models with operations like ones given some IDEs for programming languages.

We need to implement functions to generate FCSP models from scratch and to edit them. So far, instead, users have to write a program of a model in Java, to output the model as a text file, and to have the tool load it. We are planning to add the functionality of editing to the tool and to make it comprehensive development tool for FCSPs. In addition, we are going to show the effectiveness of the tool by using it for improving our previous study [2].

REFERENCES

[1] Z. Ruttkay, "Fuzzy constraint satisfaction," in *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, vol. 2. Orlando, FL, USA: IEEE, 1994, pp. 1263–1268.

[2] T. Yanagida and H. Nonaka, "Flexible widget layout formulated as fuzzy constraint satisfaction problem," in *Proceedings of the 1st KES International Symposium on Intelligent Decision Technologies (KES IDT 2009)*. Himeji, Japan: Springer, April 2009, pp. 73–83.

[3] S. Sadaoui, M. Mouhoub, and X. Li, "An OCL-based CSP specification and solving tool," *New Challenges in Applied Intelligence Technologies*, vol. 134, pp. 235–244, 2008.

[4] L. Grangier, "JCLEditor—A graphical CSP solver based on JCL," 2005. Available at http://liawww.epfl.ch/JCL/.

[5] P. Morris, "The breakout method for escaping from local minima," in *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*. Washington, DC, USA: AAAI Press/The MIT Press, 1993, pp. 40–45.

[6] R. M. Haralick and G. L. Elliott, "Increasing tree search efficiency for constraint satisfaction problems," *Artificial Intelligence*, vol. 14, no. 3, pp. 263–313, 1980.

[7] I. Miguel and Q. Shen, "Extending FCSP to Support Dynamically Changing Problems," in *Proceedings of the 8th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '99)*. Seoul, Korea: IEEE, 1999, pp. 1615–1620.

[8] J. H. Y. Wong and H. Leung, "Extending GENET to solve fuzzy constraint satisfaction problems," in *Proceedings of the 15th national/10th conference on Artificial intelligence/Innovative applications of artificial intelligence (AAAI '98/IAAI '98)*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1998, pp. 380–385.

[9] Y. Sudo and M. Kurihara, "Spread-repair-shrink: a hybrid algorithm for solving fuzzy constraint satisfaction problems," in *Proceedings of the 2006 IEEE World Congress on Computational Intelligence/the 2006 IEEE International Conference on Fuzzy Systems (WCCI 2006/FUZZ-IEEE 2006)*. Vancouver, BC, Canada: IEEE, 2006, pp. 2127–2133.

[10] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software—practice and experience*, vol. 21, no. 11, pp. 1129–1164, 1991.