F-006

Case-Based Interior Coordination

Satoshi Ono† Tatsuki Izumi†
Christopher J. Ashley† Shigeru Nakayama†

1 Introduction

Interior coordination is a task involving the careful selection and placement of materials, fittings, fixtures and furnishings. Coordinating interior is interesting activity, but it requires a great deal of time to single out the most desirable furniture items from an overwhelming pallet of selection made by numerous manufactures. It also requires highly qualified knowledge, techniques, and sense to build a functional and comfortable room with consideration to inhabitants' generation, life style, and preferences.

In this paper we focus on furniture selection and placement in interior coordination. There are methods for instrument placement and floor planning based on artificial intelligence techniques such as constraint satisfaction problem (CSP), genetic algorithm (GA) and so on [1–4]. The methods formulate the instrument placement problem involving a complicated structure, and find a solution by combinatorial search algorithm. But they do not consider instrument selection, and combinatorial search algorithms can hardly deal with the sensitivity of user preference and other implicit knowledge which are difficult to be represented as explicit numerical function. In addition, it is difficult for the methods to reuse past problem-solving results, i.e. existing interior coordination. Therefore they must solve problem instances similar to previously-solved problem from scratch.

We propose a method that presents coordinated furniture sets and their placement which adapt users' room conditions, budget, preferences by using case-based reasoning (CBR) [5–8] and CSP [9–12]. Interior coordination involves both an ill-defined and an ill-structured problems: The former is furniture selection, in which implicit domain knowledge must be dealt with, and the latter is furniture placement, in which it is difficult to find a value set that specifies all conditions for a solution. The method uses existing model rooms, which are mainly made by furniture manufacturers, distributors, stores and so on, as cases which are past problem-solving results. It enables the method to get appropriate furniture combinations, although appropriateness is hard to be represented as a numerical function. The method uses CSP to place the selected furniture set, in order to decide appropriate positions of furniture with consideration to various conditions.

This paper is organized as follows: Section 2 gives outlines of CBR and CSP. Section 3 discusses the proposed method. Section 4 demonstrates output examples of the proposed method.

†Department of Information and Computer Science, Faculty of Engineering, Kagoshima University

2 Proposed method

2.1 Principles

We propose a method for interior coordination based on the following two principles:

Principle 1: Selecting furniture items by CBR. Furniture selection must be performed with considering user's ambiguous preferences and conditions, and causes combinatorial explosion. The proposed method therefore selects the same furniture item set to the nearest case in a case base. It enables the method to produce appropriate furniture combinations without implementing implicit knowledge into the method, and to solve the problem faster than starting from scratch. The proposed method uses model rooms made by furniture manufacturers, distributors, stores, designers, and so on, as cases.

Problem-solving results obtained by using the proposed method can also be added to case base. It enables the method to acquire knowledge incrementally.

Principle 2: Placing furniture items by CSP. There are gaps between conditions and preferences of user's room and those of the model room retrieved by CBR, although CBR proposes the model room nearest to the problem. The proposed method therefore uses selected furniture items and their positions in the model room as an initial solution candidate and finds a feasible solution by repair-based combinatorial search.

The proposed method represents knowledge required to place furnitures as constraints. This enables a solution to be found, that is a feasible furniture placement, by the employment of a general combinatorial search algorithm like hill-climbing, simulated annealing, GA, and so on.

In general CBR requires a case adaptation process depending on a target problem. Formulating furniture placement, which corresponds to case adaptation, as CSP gets rid of developing the case adaptation module. This is a modification of an idea for combining CBR and GA proposed by Louis et al. [13] in order to combine CSP and CBR [14,15].

2.2 Data and Process Flow

Figure 1 shows data and process flows in which constraint satisfaction paradigm and CBR solves the problem respectively. Constraint satisfaction paradigm deals with the knowlegde necessary to place furniture items well, but has trouble with furniture selection. On the other hand, CBR deals with furniture

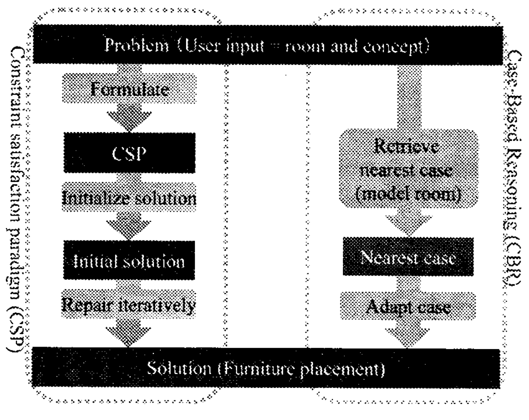

Constraint satisfaction paradigm (CSP)
Problem (User input – room and concept)
Formulate
CSP
Initialize solution
Initial solution
Repair iteratively
Case-Based Reasoning (CBR)
Retrieve nearest case (model room)
Nearest case
Adapt case
Solution (Furniture placement)

Figure 1: Process flow of CSP and CBR respectively.

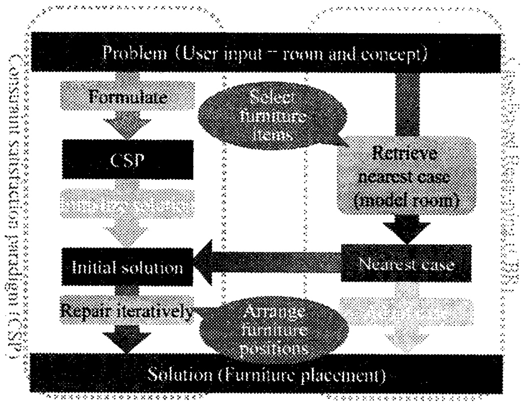
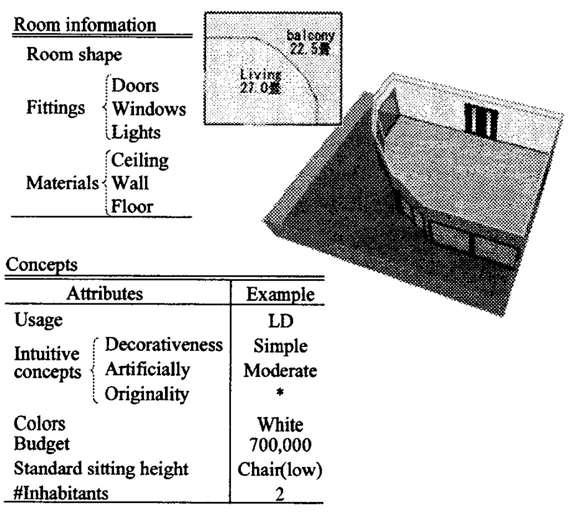

Constraint satisfaction paradigm (CSP)
Problem (User input – room and concept)
Formulate
Select furniture items
CSP
Retrieve nearest case (model room)
Initialize solution
Initial solution
Nearest case
Repair iteratively
Arrange furniture positions
Adapt case
Case-Based Reasoning (CBR)
Solution (Furniture placement)

Figure 2: Process flow of the proposed method.



Room information

Room shape

Fittings {Doors, Windows, Lights}

Materials {Ceiling, Wall, Floor}

Concepts

Attributes		Example
Usage		LD
Intuitive concepts	Decorativeness	Simple
	Artificially	Moderate
	Originality	*
Colors		White
Budget		700,000
Standard sitting height		Chair(low)
#Inhabitants		2

Figure 3: User input data.

selection well, but finds it hard to deal with furniture placement.

The proposed method therefore solves the problem as shown in Figure 2. The method selects furniture items by retrieving a case and places furniture positions by iteratively revising the case in order to support mutually between constraint satisfaction paradigm and CBR. From the viewepoint of CBR, the proposed method performs case adaptation by using CSP. From the viewepoint of constraint satisfaction, the proposed method makes an initial solution by case-based reasoning.

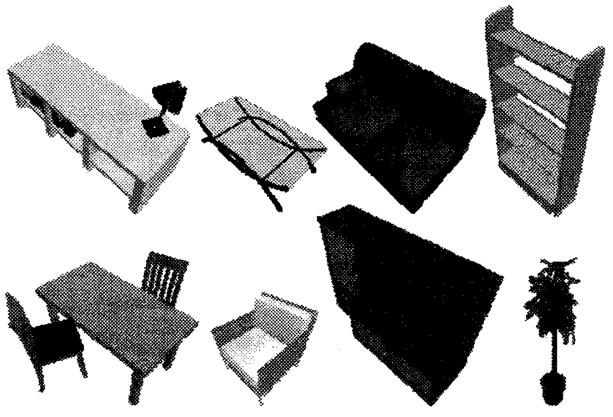


Figure 4: Furniture item examples.

Table 1: A solution example.

	furniture ID	position	direction
F_1	Sofa_KMA001	(2000, 0, 1200)	North
F_2	Table_TB15	(200, 0, 2200)	East
⋮	⋮	⋮	⋮

2.3 User Input Data

A user inputs two kind of information: room conditions and concepts. Figure 3 shows a definition and a example of user input. Room conditions consist of room size, shape, use, fittings such as doors, windows, and lights, and interior materials of wall, floor, and ceiling.

The user inputs the room's shape including fittings and interior materials by using 3D My Home Designer, a software for designing houses and apartments. Room conditions are derived from the room shape.

The user also inputs the concept for the room by using another graphical user interface. The user doesn't need to input all features. A feature of which user omits to input a value has a wild-card value '*' which means "I do not care".

Standard sitting height means the height at which inhabitants sit when they make themselves at home in the room; they sit on a floor or cusion directly placed on the floor in Japanese style room, and they sit on a chair in a dining room. Standard sitting height affects not only furniture selection of sofas, chairs and so on, but positions and heights of television, table, and other furnitures.

2.4 Furniture Database

The proposed method uses furniture items which are on the market in Japan. Each furniture item has shape

Table 2: Examples of constraints.

ID	Constraints	Type
C_1	A furniture item must be in a room.	Unary
C_2	No furniture item must be placed in front of a door.	Unary
C_3	Furniture items must not overlap each other.	Binary
C_4	At least 800 mm width space must be free in front of functional faces of furniture.	Unary, Binary

Table 3: Examples of weak constraints.

ID	Constraints	Type
O_1	Furniture items having the same standard sitting height should be closed.	Binary
O_2	A furniture item whose height exceeds 1,000 mm should be placed with its back against a wall.	Unary

and attribute information. Furniture shape is represented as Java3D code and furniture attributes are represented in XML. Figure 4 shows examles. The attribulte involves 7 features: type, name, size, functional face, color, price, and manufacturer. Type represents kinds of furniture such as sofa, chair, desk, table and so on. Functional face represents the sides which human uses the furniture. Color represents the most characteristic color of the furniture.

2.5 Furniture Selection by CBR

In the proposed method, a case involves input information, which is composed of room conditions and concepts, and a problem solving result, which is composed of selected furnitures and their positions. If there is a model room data involving room condition, concepts, furniture items, and their placements, it is easy to make a case: furniture selection and placement are only required in addition to making user input data as shown in section 2.3, and are done by using 3D My Home Designer.

Distance between user input and a case is calculated as follows:

$$\Delta(I,E) = \sum_a w_a \delta(i_a, e_a)$$

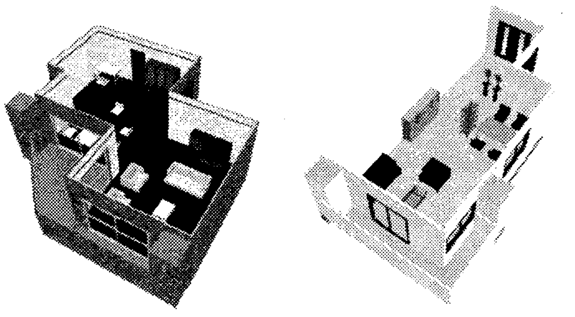
I is user input and E is a case. i_a and e_a are values of feature a in I and E. w_a is an weight of feature a. In the case that feature a has numeric value, $\delta(i_a, e_a)$ is calculated as follows:

$$\delta_n(v_1, v_2) = \left| \frac{v_1 - v_2}{v_{max} - v_{min}} \right|$$

In the case that feature a has symbolic value, $\delta(i_a, e_a)$ is calculated by distance tables which are manually made for each attribute. If i_a = '$*$', then $\delta(i_a, e_a)$ is always 0.

2.6 Furniture Placement by CSP

We define a solution as a set of variables – furniture identification number (ID), its position and direction as shown in Table 1.


(a) Model room R_{M1} (b) Model room R_{M2}

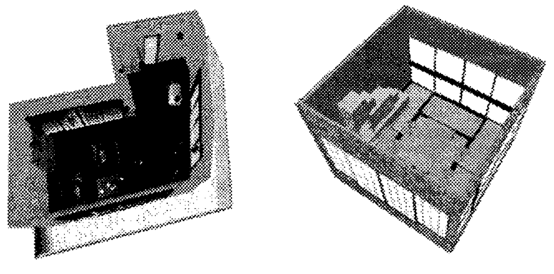
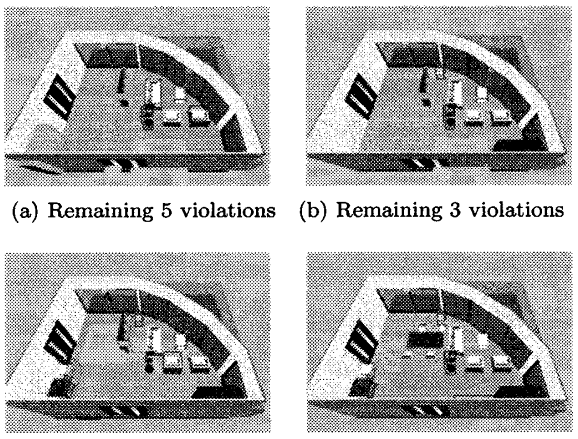

(c) Model room R_{M3} (d) Model room R_{M4}

Figure 5: Model room examples.


(a) Remaining 5 violations (b) Remaining 3 violations

(c) Remaining 2 violations (d) Remaining no violation

Figure 6: Example of furniture placement progress.

Table 2 lists examples of constraints the proposed method uses. "Type" relates to the number of variables that pertain to the constraint, and can be either "Unary" or "Binary" depending upon whether there is one or two variables defining the constraint.

The proposed method utilizes hill-climbing search algorithm to find a solution containing no violation. A solution candidate x, which is represented in Table 1, is initialized to have values of the nearest case. Till x contains no violation, the following procedure is repeated; A violating variable v is chosen randomly and some simple value-allocation operators make x's neighborhoods y_i $(i = 1, \ldots, n)$ by changing a value of v. If there is a neighbor y_k having less violations than x, x is then updated with the neighbor y_k. The operators usually changes variables representing furni-

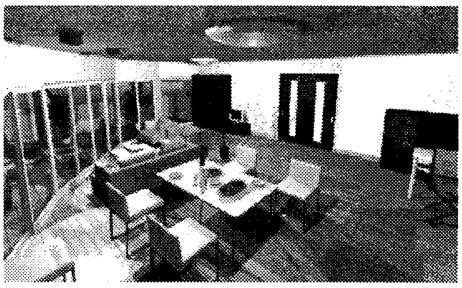


Figure 7: An example solution postprocessed.

ture position or direction, but they changes variables representing furniture ID in case that the number of violations have not decreased for T_s step, a threshold judging local minima.

After finding a solution containing no violations, the proposed method starts fine tuning of furniture positions by using some weak constraints as shown in Table 3 and the same algorithm to that of constraint satisfaction described above.

3 Output Examples

At this stage the system based on the proposed method focuses on living and living/dining room. The furniture database involves about 550 furniture items: 30 chairs, 90 sofas, 90 tables, 60 racks, and 240 storage items such as chest, shelf, cupboard, sideboard and so forth. The case base involves about 40 cases.

Figure 5 and Figure 6 show an output example of the proposed system, assuming that a user inputs room R_U shown in 3. Figure 5 shows model room examples $R_{M1}, R_{M2}, \ldots, R_{M4}$ in our system's case base. At first, the proposed system calculates distances between R_U and all model rooms, and finds R_{M1} as a room nearest to R_U. The method then selects furniture items in R_{M1}, and makes an initial solution candidate (Figure 6 (a)) from R_{M1}, but there are violations because of the difference between their room sizes and shapes. By revising furniture positions iteratively, the method produces a solution containing no violation (Figure 6 (d)).

Figure 7 shows a derived room in which small articles such as a television, foods, and so forth are placed by postprocessing.

4 Conclusion

We have proposed an interior coordination method based on CSP and CBR. The method uses model rooms as cases, searches for a solution by iteratively revising an initial solution that is the case nearest to user input. The method allows users to input ambiguous, incomplete request, and enables them to obtain an ergonomically respectful customary collocation of furniture efficiently and without the use of an interior designer. We plan to modify constraint-based furniture placement in our method to consider appropriateness of relations between furniture items evaluated by cooccurrence frequency.

Acknowledgment

We would like to thank MEGASOFT Inc. and TIS Inc. for supporting this research.

References

[1] M. Gosele and W. Stuerzlinger, "Semantic constraints for scene manipulation," *Proceedings of Spring Conference in Computer Graphics '99*, pp. 140–146, 1999.

[2] M. Korenaga and M. Hagiwara, "An interior layout support system with interactive evolutionary computation," *IPSJ (Information Processing Society of Japan) Journal*, Vol. 41, No. 11, pp. 3152–3160, 2000.

[3] J. Miles, G. Sisk, and C.J.Moore, "The conceptual design of commercial buildings using a genetic algorithm," *Computers & Structures*, Vol. 79, p. 1583.

[4] G. Zhi, S. Lo, and Z. Fang, "A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model."

[5] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*. Hillsdale, New Jersey: Lawrence Erlbaum, 1989.

[6] R. Schank and A. K. C. Riesbeck, *Inside Case-Based Explanation*. Hillsdale, New Jersey: Lawrence Erlbaum, 1994.

[7] B. Bartsch-Spörl, M. Lenz, and A. Hubner, "Case-based reasoning - survey and future directions," *Proceedings XPS-99, Springer Verlag, LNAI*, 1999.

[8] D. B. Leake and D. C. Wilson, "Combining cbr with interactive knowledge acquisition, manipulation and reuse," *Proceedings of the Third International Conference on Case-Based Reasoning*, 1999, pp. 203–217.

[9] R. Haralick and L. Shapiro, "The consistent labeling problem: Part i," *IEEE Tr. PAMI*, Vol. PAMI-1, No. 2, pp. 173–184, 1979.

[10] R. Haralick and L. Shapiro, "The consistent labeling problem: Part ii," *IEEE Tr. PAMI*, Vol. PAMI-2, No. 3, pp. 193–203, 1980.

[11] S. Minton, A. B. P. M. D. Johnston, and P. Laird, "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problem," *Artificial Intelligence*, Vol. 58, pp. 161–205, 1992.

[12] J. Singer, I. P. Gent, and A. Smail, "Backbone fragility and the local search cost peak," *Journal of Artificial Intelligence Research*, Vol. 12, pp. 235–270, 2000.

[13] S. J. Louis and G. Li, "Case injected genetic algorithms for traveling salesman problems," *INFORMATION SCIENCES*, Vol. 122, pp. 201–225, 2000.

[14] S. Ono, Y. Hamada, M. Mizutani, K. Mizuno, and S. Nishihara, "Improving search performance using case-based reasoning for floor layout rearrangement," *Proceedings of The International Conference on Computing*, Vol. 2, p. 365.

[15] S. Ono, Y. Hamada, M. Mizutani, K. Mizuno, and S. Nishihara, "A rearrangement system for floor layouts based on case-based reasoning and constraint satisfaction," *Proceedings of 21st IASTED International Conference on Applied Informatics*, p. 363.