B. Dave, A. I. Li, N. Gu, H.-J. Park (eds.), *New Frontiers: Proceedings of the 15th International Conference on Computer-Aided Architectural Design Research in Asia CAADRIA 2010*, 175–183.

FLOATING BUBBLES

An agent-based system for layout planning

HAO HUA and TING-LI JIA
ETH, Zurich, Switzerland
Southeast University, Nanjing, P. R. China
hhua@student.ethz.ch

Abstract. This program converts bubble diagram into an agent-based system for architectural design. The program suggests a model for layout planning based on bubble diagram which explicitly describes the adjacency requirements in architecture. Generally there is a basic set of rules for every agent dealing with adjacency topology and also an alternative set for other objectives. Then this basic program is developed into several generative tools for different design tasks. They imply that the agent-based system is efficient for elementary spatial arrangement and it could generate a wide range of complex solutions.

Keywords. Agent-based modeling; layout planning; bubble diagram.

1. Introduction

"Floating bubble" program comes from the bubble diagram, however, its mechanism differs from the conventional ways in which architects use the diagram in design. The main feature of bubble diagram is that it extracts adjacency requirements out of numerous objectives and constraints in architectures and it illustrates the specifications by 2D graphs which could be easily understood. The floating bubble program does follow the two features of the bubble diagram, while the primary purpose of the program is to produce complex forms with restrictions.

The program actually converts the bubble diagram into an agent-based system through programming. It is not only a research on methods but also oriented to architectural design practices which demand efficiency and flexibility. The program reproduced master plans for a research on Chinese tra-

ditional courtyards and also yielded a series of solutions with corresponding 3D-models in *Clouds Collective* project (figure 1). These projects imply the agent-based system can evolve into a basic model for layout planning and it offers unpredictable diversity in solutions.

Figure 1. Clouds Collective project.

2. Backgrounds

By means of computation there are a wide range of methods to generate layouts according to the adjacent relationships between rooms. One method is optimisation on the basis of state space search, for instance, the approach to "optimum layout of single-storey building" by Peter Manning and the programs by Whitehead and Eldars. They usually work on a data structure representing a range of solutions on regular grids and search for a best state employing special strategies. William Mitchell termed these methods with "automated spatial synthesis" in his *Computer-aided architectural design* (1977) which offers a list of strategies: generate and test, heuristic search procedures, analytical procedures, etc. Actually many of them were not always competent except for using advanced algorithms such as genetic algorithm. While the IMAGE system (Weinzapfel and Handel, 1975) discussed in the book (figure 2) may connect to many contemporary concepts such as agent-based modeling, evolution strategy and being self-adaptive.

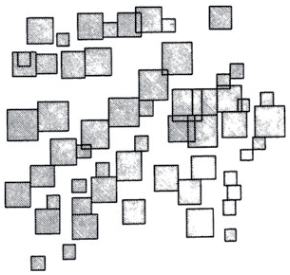

Figure 2. IMAGE system by Weinzapfel and Handel, 1975.

In fact this alternative method is similar to an active bubble diagram. Significantly, bubble diagram has indicated that layout planning task can be divided into the arrangements of relationships between one bubble (room) and another, a "divide and conquer" strategy. Thus a direct method for layout

planning is to construct a dynamic system in which every bubble floats for its own adjacency configurations. Here comes the method of agent-based modeling. It's a little surprising that the primary advantage of the method is not the efficiency on solving the adjacency problem but is the potential of producing complex forms since it extracts order from chaotic initialisation.

3. Basic model of floating bubble

As an agent-based concept the system is made up of autonomous agents, each of them (in a bubble shape) presents a room (or a space extent). Furthermore, every bubble carries its own adjacency configurations as well as its specified area. A square matrix is employed to hold all adjacency information. Every element a_{ij} in the matrix can hold a Boolean value which denotes "whether bubble i needs to be adjacent to bubble j." Whereas a_{ij} can also hold a real number indicating different levels of adjacency.

The system exposes adjacency relationships explicitly just as bubble diagram: a line connecting a pair of bubbles means the two bubbles should be next to each other; otherwise the system doesn't concern the relationship between them. These connections could also be weighted then the bubble diagram becomes a weighted graph.

Boolean values are used to describe the connections with bubbles (value of "true" denotes "two bubbles need to be adjacent" and there will be a line connecting them, figure 3) in the basic floating bubble program. This data structure helps connection lines to examine the current adjacent situation (see figure 3, wide black connection lines suggest two bubbles have not been adjacent, gray ones already adjacent) and to witness the history of adjacent situations. (The history is essential for adding perturbation to the system when it's stuck).

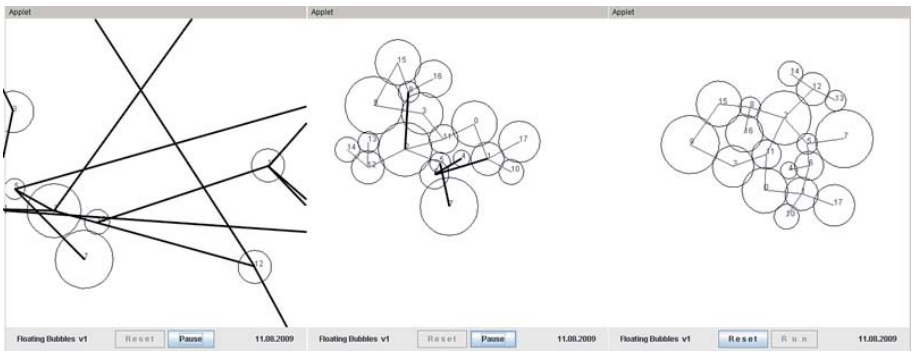

Figure 3. Three stages in the basic floating bubble system.

In the dynamic system showed above, every bubble behaves based on its own rules in a changing environment composed by other bubbles. The system is initialised randomly (first picture in figure 3) and will not stop evolving until all connection lines turn to gray (which means all of the adjacency requirements are satisfied, see the third picture in figure 3). This programming approach makes the bubble diagram live and subsequently make the program be easier to be understood compared with many other complicated spatial synthesis solutions. While it's must be emphasised that this program doesn't only aims to solve spatial problem but also to exploit the diversity of the forms resulting from the adjacent configurations.

Most generally, all the rules for agents lie in two main categories: a basic set of rules dealing with adjacency topology and an alternative set for other specified objectives.

The basic set is the essence of the system. The principle is to generate a structure emerging from "unfixed but stable" balance between several forces. Thus there must be at least two types of "forces": one drives a bubble moving towards other bubbles that it must be adjacent to; the other keeps an agent not overlapped with all other ones. The two forces are against each other, however, they are the very source building a system contributing to an unpredictable structure (layout) when it reaches equilibrium.

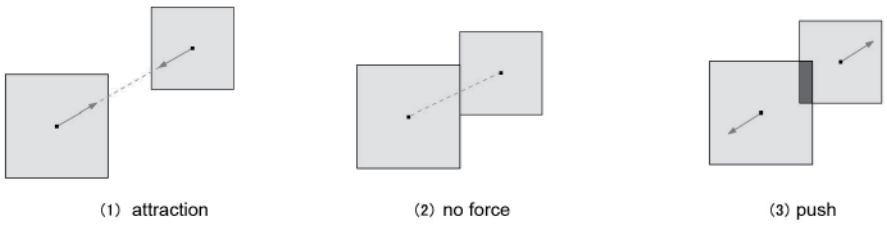

Figure 4. Forces betweens agents.

Two types of forces (attraction and push) are illustrated in Figure 4 (instead of circles, rectangles are used to present the extents of the agents). As is shown in the figure, attraction force drives two agents to be closer, more rigorously, a vector corresponding to an agent moves in this way:

$$\partial V_i / \partial t = k a_{ij}[(s-1)V_i + (1-s)V_j] \quad (1)$$

V_i and V_j are the vectors describing the positions of agent i and agent j respectively, a_{ij} denotes the weight of the connection between the two agent ($a_{ij} = 0$ means that the two agents needn't to be adjacent then there will be no attraction between them), k is a positive constant. Notice that if $s = 0$ then equation (1) becomes:

$$\partial V_i / \partial t = ka_{ij}(V_j - V_i) \tag{2}$$

In this case the attraction force will not vanish until the two vectors of a pair of agents are equal. We can also make the forces disappear once the two agents touch each other, just assign a special value to s:

$$s = (r_i + r_j) / ||V_i - V_j|| \tag{3}$$

r_i and r_j denote the corresponding radiuses of the two agents.

The equations above just take account of one of the adjacency requirements of agent i, the overall effect on the vector is:

$$\partial V_i / \partial t = \sum^n_{j=1} ka_{ij}[(s_{ij} - 1)V_i + (1 - s_{ij})V_j] \tag{4}$$

s_{ij} is the same item as the s in (1), while it is unique to every pair of agents.

On the other hand the push effect on the agent i can be described as:

$$\partial V_i / \partial t = lS\,(V_i - V_j) / ||V_j - V_i|| \tag{5}$$

S equals to the area in which a pair of agents overlap and l is a positive constant. We can see that there are both attraction and push forces for the bubbles in the same time; this is a key characteristic making the agent system evolve towards balance.

4. Improved model

Unfortunately the original system seldom works for it always gets "stuck": all the bubbles stop moving but some adjacency requirements are still not satisfied. In other words, the system always becomes "mature" before it develops its form adequately. Figure 5 shows this situation according to eight independent executions of the system: horizontal axis is the time axis and vertical value denotes the current number of unsatisfied adjacency requirements in the system.

This disadvantage can be considered as an innate consequence of the mechanisms of the agent system. Notice that all agents are moving for themselves (local optima) then it's reasonable that global target is difficult to be hit. It is a fatal problem since the system never satisfies all the requirements during dozens of executions.

One method to solve this problem is to introduce mutation, or perturbation into the system, which is guided by the "memory" of wide black lines (presents unsatisfied adjacency requirements, see figure 3). If one adjacency requirement stays unsatisfied for a long time then it drives one connected agent jumping towards another connected agent. This action breaks the current balance and usually brings chaos to the system, however, it creates

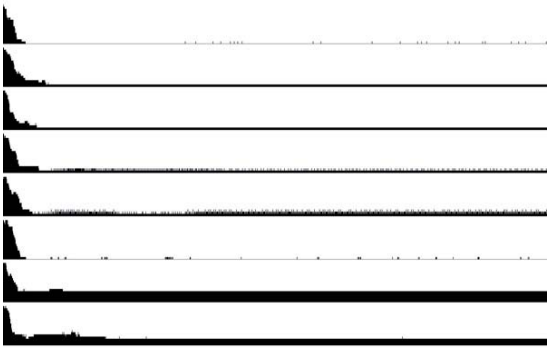

Figure 5. Records of eight executions of the original program. Horizontal axis is the time axis and vertical value denotes the current number of unsatisfied adjacency requirements in the system.

vital opportunities for the whole system to get out of stuck situation. This mechanism also contributes to the complex forms of the final solutions.

As is shown in figure 6 there are many disturbances in the number of unsatisfied requirements compared with the flat patterns in figure 5. The new pattern indicates that the system is struggling between local and global objectives, is breaking balances and making gradual achievements. The result is that the improved system is always able to reach equilibrium though the processes take more time.

5. Alternative systems

The system demands an alternative set of rules dealing with a wide range of particular requirements besides adjacency issue, for instance, some principles in plans of Chinese traditional courtyards.

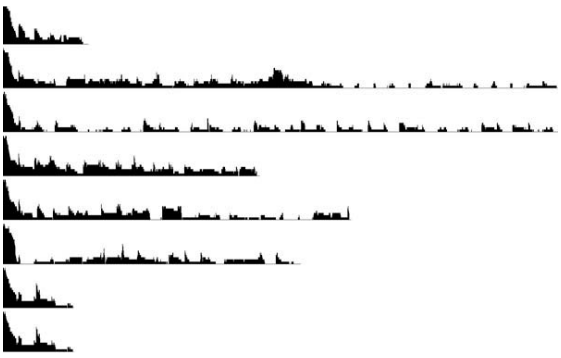

Figure 6. Records of the improved program.

Figure 7. Courtyards in Dali, China (http://blog.sina.com.cn/s/reader_4c8dfded01000e9b.html, 2008-09-18).

In spite of a great number of different types of courtyards in China, there are several common rules in planning, for example, primary courtyards are usually located around the center of a cluster of courtyards, which is consistent with traditional ethic culture. So new rules are added to the alternative set to make "important" courtyards much easier to get a central location than other ones. Usually the new rules are not opposite to basic rules, while it's necessary to negotiate between the two sets of rules to improve the performance of the system.

In another project, Clouds Collective, the system arranges spatial structure in a two-storey museum design. The rules in the alternative set deal with relationships between the rooms in different layers. For example, the two main exhibition halls in two layers are supposed to be overlapped (in top view) to get a vast space.

In addition, a BIM solution (Autodesk Revit) is connected to this system via Revit API for an integrated design flow: First, models of elements are created and modified in Revit manually (figure 10), then these models are imported into the bubble system; after the system gets a solution, the plan and 3D model of the solution are produced automatically (figures 11, 12).

Employing this method every agent in the system holds full 3D information. This project suggests a design mode that designers can both work with

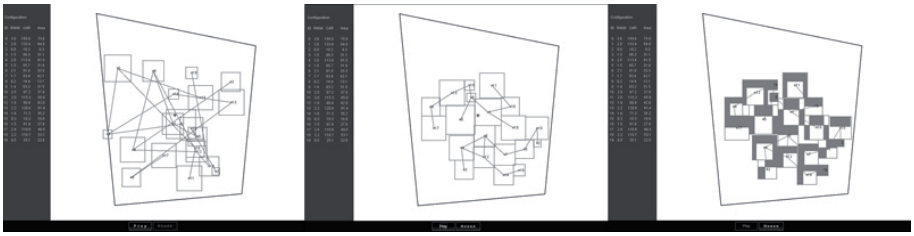

Figure 8. Three stages of the generative process.

their modeling software and benefit from automated spatial arrangement of generative system.

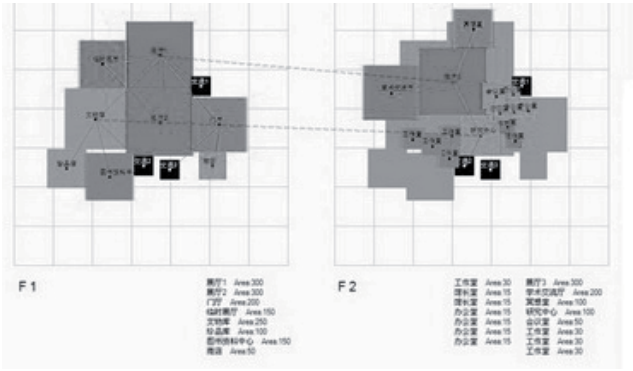

Figure 9. Two-storey layout arrangement, Clouds Collective project.

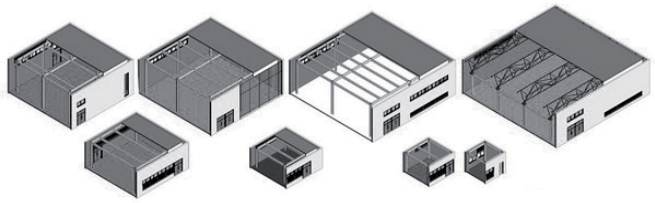

Figure 10. Modes of elements by Revit.

6. Conclusion

These applications stemming from the basic floating bubble system indicate that efficient is agent-based system for layout planning in terms of adjacency topology. Moreover, these systems don't lose the compatibility to combine other objectives such as transportation, shape constraints, space order and so on. As a result, more generative tools can be developed from the basic float-

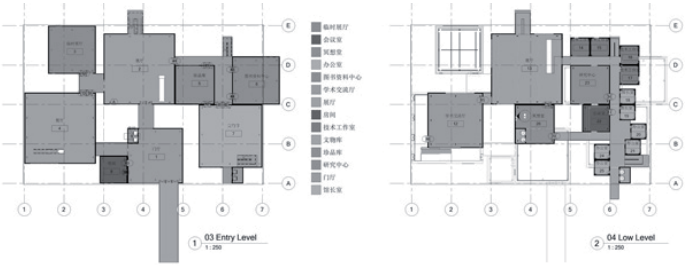

Figure 11. Plans of a solution.

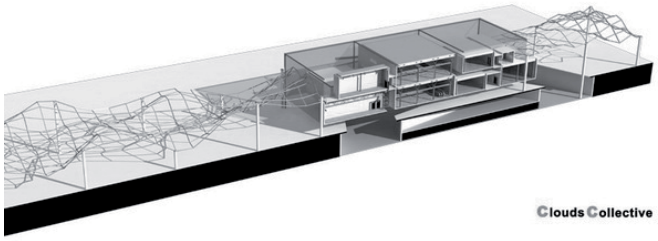

Figure 12. 3-D model of the solution.

ing bubble system. It is also reasonable that other agent-based programs could take the rules from bubble system as an attachment to their own systems, for it is sometimes true that adjacency issue is not the concentration of architects but a basic consideration in designs.

After all, in contrast to classic methods of searching a good state base on grids, floating bubble system leads to diversity rather than deterministic solutions, and towards complexity rather than finite forms.

References

Hensel M. and Menges A.: 2008, *Versatility and vicissitude: performance in morpho-ecological design,* Wiley-Academy, London.

Li, B.: 2007, A generic house design system based on multiagents: expertise of generating architectural plan, in G. Yu, Q. Zhou and W. Dong (edss), *CAADRIA 2007,* Nanjing, 191–198.

Manning, P.: 1964, An approach to the optimum layout of single-storey buildings, *The architect's journal information library,* **17**, 1373–1380.

Mitchell, William John.: 1977, *Computer-aided architectural design,* Van Nostrand Reinhold, New York, 425–474.

Ruch, J.: 1978, Interactive space layout: a graph theoretical approach, *Design automation conference proceedings 15,* IEEE/ACM, SIGDA, 152–157.

Weinzapfel, G. and Handel, S.: 1975, IMAGE: computer assistant for architectural design, *in* C. Eastman (ed.), *Spatial synthesis in computer-aided building design,* Wiley, New York, 61–68.