Modeling virtual cities based on interaction between cells

Nobuko Kato
Department of Information
Science and Electronics
Tsukuba College of Technology
Tsukuba, 305-0005

Tomoe Okuno and Hirosi Kanoh
Institute of Information Sciences
and Electronics
University of Tsukuba
Tsukuba, 305-8573

Abstract
In this paper, we propose a novel method that enables automatic modeling of time varying virtual cities. We apply cellular automata to lay out many build-<br/>ings, and a genetic algorithm to produce time series change of virtual cities. We produce virtual cities by giving cellular automata several states, vacant, a variety of buildings, two kinds of roads. The sequence of applied rules is determined by using a genetic algorithm, so as to generate the virtual city required by users.

Key words: Cyber Space, Cellular Automata, Genetic Algorithm

1 Introduction
A variety of virtual cities are widely used for World Wide Web sites and 3-D games. Modeling of these virtual cities requires a great deal of expense and effort. A method for modeling a virtual city based on a photograph of an actual city has been proposed [1], but new cities cannot be modeled this way. Using artificial life techniques has recently attracted a lot of interest for producing various novel patterns [2][3][4]. We have developed a method that uses artificial life techniques to model original virtual cities that have the characteristics of actual cities[5][6].

In this paper, we propose a novel method for modeling virtual cities that uses cellular automata to lay out many buildings, and genetic algorithm(GA) to produce time series change of virtual cities. In the following sections, we describe the details of each process. In section 2, we show the outline of city landscape and our basic strategies. Section 3 describes the coding method and the fitness function. Examples of virtual cities are presented in Section 4.

2 Virtual City

2.1 City Landscape
An actual city is composed of several types of areas: residential, business, high-rise, etc. Thus each area has its own characteristics, and the way of distribution and shape of these areas gives the landscape of the city. Each typical area has a sort of thematic uniformity of characteristics with its own, which include floor space, building height, building use, roofline and pitch of a roof. On the other hand, a city is complex system as a result of large number of simple local interactions.

We applied a genetic algorithm for determining the feature values of these typical areas[6]. Since this method is not able to express the urban development, we propose the new method of modeling time varying virtual cities in order to study the self-organization process.

2.2 Cellular Automata
Cellular automata are mathematical models for complex natural systems containing large numbers of simple identical components with local interactions [8]. Definition of cellular automata is as follows.

- Cellular automata consist of a lattice of sites.
- Each site takes on k possible values.
- The values of the sites evolve synchronously in discrete time steps according to identical rules.
- The value of a particular site is determined by the previous values of a neighbourhood of sites around it.

αt(x, y) is taken to denote the value of site (x, y) in a two-dimensional cellular automaton at time step t. Each site value is specified as an integer in the range 0
through k-1. The site value evolve by iteration of the mapping

\[ a_{t+1}(x,y) = F(a_t(x-r, y-r), a_t(x+r, y+r), \ldots, a_t(x-r, y+r)) \]  

(1)

F is an arbitrary function which specifies the cellular automata rule.

Figure 1 shows an example of a cellular automata which have k=2 and r=1. F is calculated using the sum of the Moore neighborhoods value V. We call V as field value.

\[ V = a_t(x-1, y-1) + a_t(x-1, y) + a_t(x-1, y+1) + a_t(x, y-1) + a_t(x, y+1) + a_t(x+1, y-1) + a_t(x+1, y) + a_t(x+1, y+1) \]  

(2)

\[ a_{t+1} = \begin{cases} 0 & (a_t = 1, V < 1) \\ 0 & (a_t = 1, V \geq 4) \\ 1 & (a_t = 0, V = 3) \end{cases} \]  

(3)

2.3 Our Strategies

We apply cellular automata to lay out buildings and GA to produce time series of changing virtual cities. Our method for modeling a virtual city has four key components.

- We regard a virtual city as a grid of cells, and each cell is in any one of seven states: vacant, four kinds of buildings and two kinds of roads.
- An interaction among cells is described as a field. We take into account the effect of interaction among buildings and roads by using the strength of the fields.
- The state of the cell is updated according to a building layout rules and road generation rules. The sequence of applied rules represents time series of changing virtual city.

3 Our Method

3.1 Format of cells and fields

A virtual city is represented as grid of cells, each with an integer \((s=0: \text{vacant}, 1: \text{office}, 2: \text{shop}, 3: \text{apartment}, 4: \text{house}, 5: \text{arterial road}, 6: \text{access road})\) as shown in Figure 2. A cell distribution in each state is described as a field \(V_s\). \(V_s\) is calculated based on distance between neighboring cells at each change of cell states. There can thus be several kinds of fields \(V_s\) depending on cell states \((s=0, 1, \ldots, 6)\) as shown in Figure 3.

Let denote a state of a cell at position \((x, y)\) of time \(t\). Number of cells within a distance of \(r\) cells from \((x, y)\) is calculated using

\[ A_s(x, y, r) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} \delta(s, a_t(x + i, y + j)) \]  

(4)

, where \(\delta\) is Kronecker delta. Number of cells in state \(s\) at a distance of \(r\) is calculated using

\[ \Delta A_s(x, y, r) = A_s(x, y, r) - A_s(x, y, r-1) \]  

(5)

Field strength at position \((x, y)\) is defined using \(\Delta A_s(x, y, r)\) and weighting factor \(W(r)\) as follows.

\[ V_s(x, y) = \sum_{r=1}^{6} W(r) \cdot \Delta A_s(x, y, r) \]  

(6)

- The genetic algorithm search of the sequence of applied rules generates the virtual city required by users.
3.2 Transition Rules

A cell state is updated in discrete time steps according to a stochastically applying local interaction rules. A format of rules is defined as follows:

<table>
<thead>
<tr>
<th>rule id</th>
<th>probability</th>
<th>if</th>
<th>a current cell state, field strength</th>
<th>then</th>
<th>a next cell state</th>
</tr>
</thead>
</table>

Example of rules shown in Table 2 can be denote by a following equation:

\[
\alpha_{t+1}(x,y) = \begin{cases} 
1 & (\alpha_t(x,y) = 0, V_6(x,y) \geq 30) \\
1 & (\alpha_t(x,y) = 0, V_6(x,y) \geq 30) \\
0 & (\alpha_t(x,y) = 1, V_6(x,y) \geq 30) 
\end{cases}
\]

### Table 1: Weighting factor \(W(r)\)

<table>
<thead>
<tr>
<th>(r) (number of cells)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>(W'(r))</td>
<td>12</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1 shows the weighting factor \(W(r)\).

### 3.3 Application of Genetic Algorithm

#### 3.3.1 Coding

We apply GA search to generate a time series pattern of changing virtual cities. A sequence of applied rules determine time series pattern and a GA searches a proper sequence of rules. A chromosome represents a city. The coding method is shown in Figure 4. Each locus stores a rule id.

#### 3.3.2 Fitness Function

The fitness values of each individual is calculated using

\[
\text{Fitness} = F_1 + F_2
\]

\[
F_1 = \sum_{i=1}^{4} \omega_1(i) \cdot f_1(t_i)
\]

\[
F_2 = \sum_{i=1}^{4} \omega_2(i) \cdot \sum_{j=1}^{N_i} f_2(s_{ij}) = \frac{f_2(s(N_i+1))}{N_i}
\]

where

\[
\omega_1(i) : \text{ weight} \\
\omega_2(i) : \text{ weight} \\
i : \text{ building number (1: house, 2: apartment, 3: shop, 4: office)} \\
R_i : \text{ building ratio = (number of buildings i) / (total number of buildings)} \\
j : \text{ area size ranking} \\
N_i : \text{ area size for building i} \\
s_{ij} : \text{ area size = } \frac{j\text{-th largest area size}}{\text{total number of cells}}
\]

\(F_1\) is the fitness of the building ratio, and is calculated using \(f_1\) shown in Figure 5.

\(F_2\) is the fitness of the area, and is calculated as follows:

1. An area for \(i\) is defined as a cluster of cells having value greater than threshold value. An area size \(s_{ij}(j = 0, 1, \ldots)\) is the number of cells in a cluster.
2. $f_2$ shown in Figure 6 is calculated using area size $s_{ij}$ of largest $N_i$ areas for $j (j = 1, \ldots, N_i)$ . $F_2$ is sum of the $f_2(s_{ij})$

3. $F_2$ is deducted using $N_{i+1}$-th area size as a penalty for excess areas.

4 Experiments

For our evaluation experiments we used 100x100 cells. Each chromosome has 100 rules, and 100 individuals generated randomly. Population size is 100 and upper limit of generation is 200.
4.1 Sequence of rules

Figure 7 shows four types of virtual cities using sequence of rules generated manually: a) uniform city, b) a random city, c) an ordering city, and d) a city that has several areas. This results show that different sequence of rules can generate different types of virtual cities.

4.2 GA search

Table 3 shows users requests which we use. Figure 8 shows an example virtual city generated using the request (a) before and after searching. Another example virtual city generated for the request (b) is shown in Figure 9. Figure 10 shows time series of changing virtual city generated for the request (c). These results indicate that the genetic algorithm search of a sequence of rules works well to produce time series of changing virtual cities required by users.

Figure 11 represents an example virtual city with tree type road networks. This means that genetic algorithm works well to produce realistic virtual cities.

Figure 7: Four types of virtual cities

Table 3: User demands

<table>
<thead>
<tr>
<th>Request</th>
<th>Ratio</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td></td>
<td>middle shop area: 1</td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td>middle office area: 1</td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td>small house area: 3</td>
</tr>
<tr>
<td>(d)</td>
<td>0.5:0.2:0.2:0.1</td>
<td>large house area: 1</td>
</tr>
</tbody>
</table>

*Ratio: house/apartment/shop/office

Area: area size: number of area

Figure 8: Example virtual city (middle office area: 1, middle shop area: 1)
5 CONCLUSION

We proposed a novel method for modeling virtual cities that uses cellular automata to lay out many buildings, and a genetic algorithm to produce time series change of virtual cities. Examples of virtual cities verify followings. We can successfully generate four types of virtual cities: a uniform city, a random city, an ordering city, and a city that has several areas as actual cities. The GA search of a sequence of rules works well to produce various types of virtual cities required by users and time series of changing virtual cities.

References


