Modular architecture shape design recursive algorithm

Raphaël Sasportas[1,*], Feriel Mustapha[1,2]
[1]Laboratory of Research on Eco-Innovation of Industry and Energetic, ECAM-EPMI, 13 Boulevard de l'Hautil, 95092 Cergy-Pontoise (France)
[2]Laboratoire of Mechanic and Materials of Civil Engineering, University of Cergy-Pontoise (France)

Abstract. The architecture and architectural shape of a building plays an important role in the interior housing design, many concepts have been developed to optimize this and to have a housing that reflects both visual and architectural comfort or even thermal one. The evolution of these houses however is not controlled, modular architecture has appeared because among its basic principles is building from modules that can be assembled, then separately modified or even disassembled, the advantage of this typology of architecture is that it is fast in execution, light in installation and above all has very little impact on the environment in which it is part. These buildings are characterized by more dynamic geometries, which results in a very rich architectural language and an interesting sky line, however, to be able to project all the possibilities is a challenge in itself. In this article the discrete geometry is treated through the development of an algorithm that illustrates the process that allows to expose all possible cell combinations, the latter represents the basic module of modular architecture, the most attracting choices will be used to illustrate the formal and architectural optimization of a house. In this study the emphasis will be placed on the mathematical part in order to highlight the relationship between geometry and its influence on the final architectural rendering

1 INTRODUCTION

Discrete geometry is a subgraph of square meshes composed of an infinity of several unit cells paired edge to edge, with adjacent cells forming the edges of the graph. These studies have a long history in the scientific literature, but they were first presented by Solomon Golomb, then by Martin Gardner LQ KLV $PHULFDQ FROXPQV 6FLHQWLILF 0DWKHPDWLFDO DPHV [1], and by several research articles by David Klarner. They are now one of the most popular subjects in the field of mathematical recreation and have attracted the interest of mathematicians, physicists, biologists and computer scientists [2].

Work on the analogy between mathematics and the rest of the disciplines is now omnipresent, particularly between architecture and geometry, which are undeniably linked at the morphological and structural levels. Discrete geometry is applied in modular architecture, the latter is characterized, in the design phase, by the use of identical combined cells, by juxtaposition and/or superposition [3]. Among the adheres of modular architecture existing today, the architecture of containers, indeed this metal box is a standard product so considered as a basic cell, mainly used for logistics [4]. This archetype is inclined to be studied from a purely analytical point of view, and therefore the overview on the number of combination possibilities is difficult to visualize it without a numerical processing. These combinatorial morphologies have a relative impact, these impacts are mainly of an aesthetic nature, but also structural.

However, from a practical point of view these constructions face physical constraints that limit the choices in terms of purely morphological possibilities, because, for structural reasons, for example, the superposition of 15 containers cannot be achieved. From an architectural point of view, this type of construction reflects a strong ecological approach [5], insofar as the reuse of used containers is possible, therefore, during the design phase the architect decides on the number of cells in this case containers required for the projects, thus meeting the specifications from the surface point of view, and thus satisfy the criteria required by the project, the use of discrete geometry is first of all a temporal and morphological optimization, having the choice would considerably facilitate the design, while being in conformity with reality.

In this article, the main objective is to demonstrate the possibility of using discrete geometry in real conditions to serve modular architecture and architects to explore all possibilities. In this paper we showed that the concept of recursion and its application contributes greatly to the success of architectural projects of this type, more specifically we describe in the following sections graphically and analytically an algorithmic approach to illustrate the possible combinatorial forms.

* Corresponding author: r.sasportas@ecam-epmi.com

2 ALGORITHM DESCRIPTION

As already said, this section describes the basic recursive algorithms that generate all possible shapes with a predefined number of cells. The cells must be adjacent and every cell in the shape must be located above another cell or on the ground (see figure below)

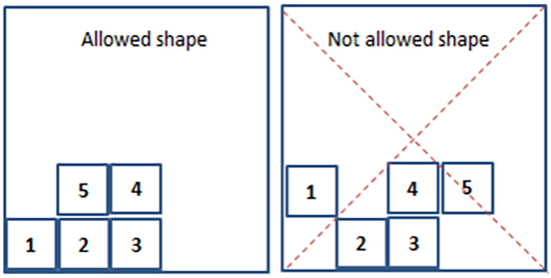


Figure 1: connexity constraints (examples of allowed/ not allowed shapes)

We first describe the basic recursive algorithm, outline its drawbacks and then we present a second algorithm that corrects the imperfections of the first.

2.1 Basic recursive algorithm

The recursive algorithm used to generate all possible shapes composed of $N+1$ cells is shown below (figure 2). The algorithm has two input arguments. The first argument is a binary 2D matrix whose elements have a value 0 except for a single element whose value is 1 and which represents the initial cell. The second argument is the number of cells to be added to the initial one. For each cell the algorithm checks its closest neighbours located in East, North and West directions. In the North direction (line 17 of the algorithm) whenever the location value is 0 the value is set to 1. East direction, whenever the location value is 0 the value is set to 1 if and only if there is the ground or a cell in the South-East direction (constraint 1 line 11). In West direction, whenever the location value is 0 the value is set to 1 if and only if there is the ground or a cell in the South-West direction (constraint 2 line 23) (see figure 2). These two constraints allow to construct a subclass of 2D polyminos [6] that correspond to physical construction: there is no suspended cell, each cell in the generated shape has a support cell or the ground. When a location value is set to 1, the same algorithm is executed with input arguments corresponding to the updated 2D matrix and number of cells minus one (lines 13, 19 and 25). The algorithm execution is stopped when the number of cells to be added is equal to 0. The output of the algorithm is a 3D array (TabConfig) in which each slice is a binary 2D matrix corresponding to a shape.

Algorithm 1.

```
1   function y=ShapeGenV1(M,N)
2   global nbcfg;
3   global TabConfig;
4   [nl,nc]=size(M);
5   if N~0
6       for i=1:nl
7           for j=1:nc
8               if M(i,j)==1
9                   %check East neighbor
10                  if M(i,j+1)==0
11                      if i-1 == 0 | M(j+1,i-1) == 1
12                          A=M;A(i,j+1)=1;
13                          ShapeGenV1(A,N-1);
14                      end
15                  end
16                  %check North neighbor
17                  if M(i+1,j)==0
18                          A=M;A(i+1,j)=1;
19                          ShapeGenV1(A,N-1);
20                  end
21                  %check West neighbor
22                  if M(i,j-1)==0
23                      if i-1 == 0 | M(i-1,j-1) == 1
24                          A=M;A(i,j-1)=1;
25                          ShapeGenV1(A,N-1);
26                      end
27                  end
28              end
29          end
30      end
31  else
32  nbcfg=nbcfg+1;
33  TabConfig(1:nl,1:nc,nbcfg)=M;
34  end
35  end
```

Figure 2. basic recursive proposed algorithm

2.2 Improved recursive algorithm

The major drawback of the basic recursive algorithm is that it generates, among all possible shapes, subsets of identical shapes. From an algorithmic point of view, due to its recursive processing, the identical shapes are in fact different shapes because the algorithm takes into account these two following points

a) for a given spatial occupation the cells are differentiated as we can see in configuration B. (see Figure 3)
b) spatial occupation of the shape as we can see in configuration D. and C. below. (see Figure 3).

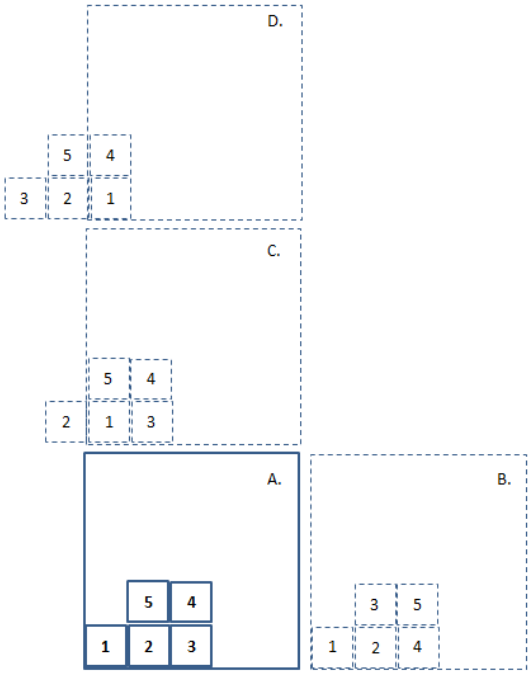


Figure 3. algorithm translation of the same shape

After the launch of the algorithm, the generation of shapes is efficient but we face the problem of-D-let, which is the phenomenon of having duplicate, triplate, quadruplicate..., indeed we are witnessing the graphic generation of the same shapes several times.

In order to reduce the number of identical shapes resulting from the remark in a. we added a condition in the algorithm (line 23) (see Figure 4) which avoids to generate shapes that are only translations of shapes already generated.
To eliminate the identical shapes resulting from the remark in b. we added lines of code starting at line 37 to line 45. This additional processing allows to verify that each generated shape is unique by comparing it with previously stored shapes. If there exists among the stored shapes a shape identical to that which has been generated then it is not memorized in the TabConfig array.

Algorithm 2.

```
1   function y=ShapeGenV2(M,N)
2   global nbcfg;
3   global TabConfig;
4   [nl,nc]=size(M);
5   if N~0
6       for i=1:nl
7           for j=1:nc
8               if M(i,j)==1
9                   %check East neighbor
10                  if M(i,j+1)==0
11                      if i-1 == 0 | M(j+1,i-1) == 1
12                          A=M;A(i,j+1)=1;
13                          ShapeGenV2(A,N-1);
14                      end
15                  end
16                  %check North neighbor
17                  if M(i+1,j)==0
18                          A=M;A(i+1,j)=1;
19                          ShapeGenV2(A,N-1);
20                  end
21                  %check West neighbor
22                  if M(i,j-1)==0
23                      if j-1 >= 10
24                      if i-1 == 0 | M(i-1,j-1) == 1
25                          A=M;A(i,j-1)=1;
26                          ShapeGenV2(A,N-1);
27                      end
28                      end
29                  end
30              end
31          end
32      end
33  else
34  if  nbcfg==0
35      nbcfg=nbcfg+1;
36      TabConfig(1:nl,1:nc,nbcfg)=M;
37  else
38      k=1;
39      same=0;
40      while(k < nbcfg+1)
41          if(TabConfig(1:nl,1:nc,k)==M)
42              same=1;
43          end
44          k=k+1;
45      end
46      if same==0
47         nbcfg=nbcfg+1;
48         TabConfig(1:nl,1:nc,nbcfg)=M;
49     end
50  end
51  end
```

Figure 4. improved recursive proposed algorithm

3 SIMULATION RESULTS

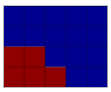
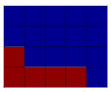
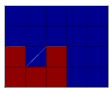
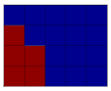
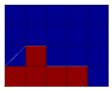
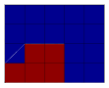
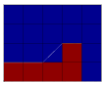
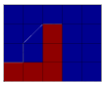
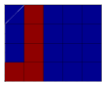
In this section we present results obtained by executing the second algorithms (see Figure 4) written in MATLAB. We have set the number of cells to ($N = 4$) and in order to execute properly the algorithm we run the following script:

```
1  clear global;
2  global nbcfg;
3  global TabConfig;
4  nbcfg=0;
5  M=zeros(20,20);
6  M(1,10)=1;
7  tic;ShapeGenV2(M,4);toc;
```

Figure 5. script used to run the improved recursive algorithm

The binary slices of TabConfig, converted in binary images are shown in table 1.

Table 1. graphic results with 5 cells

 $k = 3$	 $k = 4$	 $k = 3$	 $k = 3$
 $k = 2$	 $k = 2$	 $k = 2$	 $k = 4$
 $k = 3$	 $k = 3$	 $k = 4$	 $k = 5$
 $k = 4$	 $k = 3$	 $k = 2$	 $k = 1$

In the table below we have reported the results of the algorithm with different number of N. The execution time is measured on a CPU running at 1 GHz frequency (see Table 2).

Table 2. Graphics results with 5 cells

N+1 (number of cells)	Number of shapes	Execution time (s)
3	4	0.015
4	8	0.016
5	16	0.078
6	32	0.141
7	64	0.936
8	128	8.174

3.1 Quantitative analysis

We can see from the table above that the number of shapes with $N + 1$ cells are equal to 2^N . To prove this result let us note by k the number of adjacent cells on the ground (see Table 1). For each value of k, k varying between 1 to $N + 1$, the number of shapes resulting from the positioning of $N + 1 - k$ cells on k cells are showed by the equation (1):

$$C_{(N+1-k)+k-1}^{k-1} = C_N^{k-1} = \frac{N!}{(N-k+1)!(k-1)!} \quad (1)$$

The total number of shapes is then:

$$\sum_{k=1}^{N+1} C_N^{k-1} = \sum_{k=0}^{N} C_N^k 1^{N-k} 1^k = (1+1)^N = 2^N \quad (2)$$

4 CONCLUSIONS

Developing an algorithmic script based on recursive geometries, and allowing the generation of all morphologies is a considerable time saving, for architects and all other professionals working in the construction sector.

Indeed, thanks to this contribution, we can claim to have a global vision of all the possibilities from a purely geometric point of view, then the architect decides on the choice best suited to his specifications, both aesthetically, by choosing the volumetric composition that corresponds perfectly to urban language, and spatially by optimizing the interior layout that will form the spatial composition of the inhabitants. in addition thanks to this analytical work we are now also able to predict, from the beginning, the number of exact possibilities we can have, and those of course according to the conditions we have used. That said, this optimization can indeed satisfy both constraints at the same time, and in this perspective, future work will focus on combining several criteria in an optimization, more precisely energy optimization, which today represents a major challenge and an international challenge.

References

[1] J. O. a. C. D. T. J.E. Goodman, Dis and Com Geo Rat: Pres, (2017).

[2] G. B. a. M.Moffie, «On the com of Jen alg for cox pol» Jou of Disc Alg, 5. 348-355, (2007).

[3] W. Y. T. D. A. P.Mendis, «New adv cha and opp of mod bui ±A sta-of-the-art rev»Eng Str, 183. 883-893 (2019).

[4] «International Organization for Standardization,» ligne]. Available: https://www.iso.org/about-us.html.

[5] YASHADA, «Nat Sym on Nem Man: A Cha to Ind A in the Cha»Nem Soc of Ind and Ind Cou of Agr I (2015).

[6] A. J. G. a. I. Jensen, Pol, Pol and Pol, Spr, Dordr (2009).