Go to contents 17

Positioning of Buildings in a Land

BARRIONUEVO, Luis F.
Laboratorio de Sistemas de Diseño
Facultad de Arquitectura y Urbanismo
Universidad Nacional de Tucumán
Av.Roca 1800, S. M. de Tucumán, 4000, Argentina
http://www.labsist.net

Configurational studies are useful tools to architectural designing, since they help to understand the grouping of objects in the space (two-dimensional and/or three-dimensional). The designer, after a classification of objects that satisfies the needs set to a group of objects, impose some restrictions to the objects that will govern the composition. These restrictions are those that will define the result through operations carried out by the designer. Among these operations the location of buildings in a determined area and the particular environmental qualities condition the final result.
This work presents the results obtained by means of the implementation of a computing program of the type Evolution Program (EP) implemented in language C. The implementation of the program is explained in the first part of the paper. In the second part the successive steps are described.
The numerical results obtained with the mentioned program are shown graphically. Examples of different complexity level illustrate the discussion of the theoretical matters.
***Keywords:** Positioning Buildings, Configurational Studies, Evolutionary Design, Evolutionary Algorithms, Evolutionary Programming.*

Introduction

Design and planning are activities needing a large range of substantially different data. In computer programming terms, the bulk of constraints to be included in the design process can be broken into small particular sets provided that the involved operations could be integrated into a major set once the results are obtained. *“Analysis tools that can simulate and measure the performance of designs are also becoming more common, with much of engineering design relying on software analysis to test designs before prototypes are built”. Bentley (1999).*

The graphical presentations of the program output ease the evaluation of alternative locations. Barrionuevo (1999).

This paper describes a method to locate objects (Buildings) on a place (Land) considering its context (environment), using the principles of the Evolutionary Design, Bentley (1999b). It aims to the following objectives:

- To avoid the overlapping of objects.
- To obtain a distribution of objects on the site, according to the orientations the designer wants to preserve.
- To make sure that some determined sights from the buildings are preserved.
- To obtain a set of alternative object groupings aiming to improve land use.

These objectives can be satisfied through the implementation of an Evolutionary Algorithm, Michalewicz, Z. et al (1996). This kind of algorithms

Go to contents 17

are useful to fast analysis. When the amount of data and restrictions to be examined is of considerable bulk an exhaustive analysis become unfeasible. In these cases “Evolutionary Programming” is needed (Fogel J., 1963; Fogel D., 1992). Based on that kind of technique a method of generation of alternative groupings is presented in the forthcoming part of this paper. The numerical results are shown graphically.

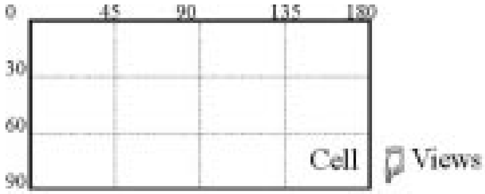


Figure 1: Shape, dimensions (in meters) and better views of the land

Purposes

Architectural objects can be located in the space following configurational rules. This rules are selected according to aims or restrictions such as orientation, desired and undesired views, distances between buildings often imposed by urban regulations. Combes and Barrionuevo (1999).

Figure 2: The geometry building plans

Data Representation

For the sake of simplicity a rectangular area will be used to illustrate step by step the implementation of the algorithm. Figure 1 shows the dimensions and orientation of a rectangular land taken as an example.

The land will be divided in an array of 3 rows by 4 columns. It will help the insertion of an object representing a building on each cell.

The geometry of the architectural objects will be defined in the following way. See Figure2:

They have three possible height each one (Figure 3):

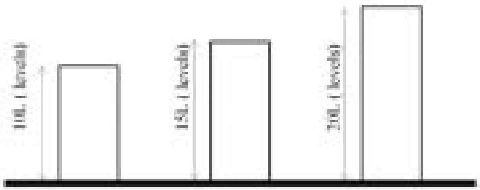


Figure 3: Possible building heights

Each object will be able to rotate around a point located on one of its vertexes. The Table 1 shows the different rotations:

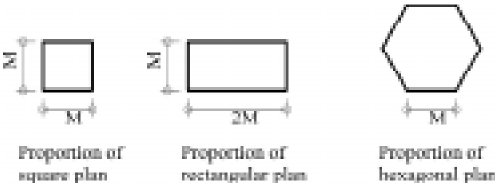
Table 1: Possible rotations of the buildings in plan.

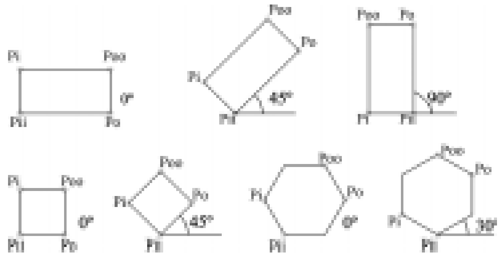
Rotation Objects	0° (Degree)	30° (Degree)	45° (Degree)	90° (Degree)
Square				
Rectangle				
Hexagon				

Go to contents 17

Figure 4 (left): Buildings plan proportions

Figure 5 (right): Control Points





On the horizontal plane the object proportions are defined as shown in Figure 4, where M = 15 units:

Implementation of the Evolution Program (EP)

The EP is determined by the following structure (Michalewicz Z., 1995, 1996):

```
begin
  initialize()     Variables initializes
  initpop()        Population initializes
  evaluate()       Population evaluates
  the_best()       Obtains the best chromosome
  while(visual_area<visual) do
  begin
    Fitness()      Function to obtain the fitness
    select()       Selection function
    cross()        Function of Crossover
    mutate()       Mutation Function
    evaluate()     Evaluation function
    the_best()     Obtains the best chromosome
    AddBest()      Adds the best chromosome to the
    evaluated population
    elitist()      Elitist function
  end
  report()         It generated a report
end
```

For the implementation of the Evolution Program they were considered four aspects: structure of the chromosome, evaluation function, selection function, the transformation operators.

Design of the Structure of the Chromosome

Owed to its geometric characteristic the designed chromosome starts from a structure whose basic entity is a geometric point in 2D.

```
struct SPoint {
double x;        Coordinate in x
double y;}       Coordinate in y
```

Where x and y are the coordinates in the plane that determine the position of the building in the land.

To define the object representing to the building to insert in the land, the following structure of data is defined:

```
struct SObject {
int Type;            Type of object
int Height;          Height of the object
int Rotation;        Rotation of the object
struct SPoint Pi;    Insert point and overlapping
    control
struct SPoint Pii;   Overlapping control point
struct SPoint Po;    Overlapping control point
struct SPoint Poo;}  Overlapping control point
```

Where Type stands for the type of architectural object to insert on the land. Height is the height of the building to insert. Rotation indicates the rotation type to carry out on the building. Pi indicates the insert point of the building on the land. Pii, Po and Poo are control points that will be consulted to ensure that buildings do not overlap (fig 5).

Go to contents 17

Each cell of the array that conforms the land is modeled respecting its possibility to contain (or not) an architectural object. The following data structure is proposed:

struct SCell *{*
int Occupied*;* *determines if a building occupies this cell*
struct SObject Object*;}* *building data*

Occupied is a variable that represents the existence (different of 0) or not (0) of a building on a certain cell. Object is a variable that records the qualities defining the building to insert in the cell. These are building type, height of the building, type of rotation to consider, and the points that define the position of the building on the land.

Lastly, each instance of a land will be considered a chromosome to be evaluated by the evolution programs. The following data structure is defined:

struct SChromo_floor *{*
struct SCell Land[3][4]*;* *Array representing the land (cells)*
int Quantity_Obj*;* *Quantity of Objects inserted*
float Fitness, RFitness, CFitness*;}* *Genotype´s relative and cumulative fitness*

The array Land[3][4] records the information corresponding to the cells that define the land with the buildings positioned on each one of them.

"Evaluation Function" Implementation

The fitness of every chromosome constituting each population is assessed by the evaluation function. For the function implementation the following variables are considered to evaluate: i) views from the buildings, ii) percentage of use of the land, iii) location of the highest buildings, and iv) overlapping among buildings. The evaluation function is defined for,

Fitness=Fc3*view(P)+Fc2*land_use(P)+Fc4*highest_tower(P)-Fc1*overlapping(P)

A function is implemented to each variable to evaluate. Each function calculates the values that correspond to each population's chromosome. Also, exist factors that determine the execution priority of each variable in the evaluation function (Fc1, Fc2, Fc3 and Fc4). The numeric result obtained when applying the function of evaluation is recorded inside each one of the chromosomes. Once evaluated the population of chromosomes for a certain generation, the best adapted chromosomes are selected. This is carried out by means of a "most capable" mechanism of selection.

Implementation of the "Function of Selection"

It is based on a mechanism of population's sampling. For this case the "Stochastic Sampling" has been chosen: a value of "cumulative fitness" between 0 and 1 is assigned to each chromosome, which represents the probability of being selected among all the chromosomes that compose the current population. In this work the "Universal or for Roulette sampling" type was implemented. A random number is generated and by this means the selection of chromosomes is carried out.

The selection function is defined by means of the following algorithm:

```
foreach Population[a]
{p=rndreal(0.0,1.0)
 if(p<Population[0].CFitness)
  foreach Object
   NewPopulation[0]¨Population[0]
 else
 {foreach Population[b]
{if((p>=Population[b].CFitness)AND(p<Population[b+1].CFitness))
   {foreach Object
     NewPopulation[b]¨Population[b+1]}}}}
foreach Population[c]
 foreach Object
  Population[c]¨NewPopulation[c]
```

The selected chromosomes can come up either by chance or also owed to their bigger probability assigned by means of the function of stochastic

Go to contents 17

sampling. Each selected chromosome will reproduce through the use of transformation operators: in this case crosses and/or mutation. They will conform the next population to be evaluated by the program.

Transformation Operators

Two types of operators have been implemented: crossover and mutation. Both act on columns or rows for each chromosome (land + buildings). To prevent not valid results a validation and correction function should be implemented. The bigger the lapse between the first and the last generation, the bigger the probability of obtaining a best order in the arrangements. This is the result of having implemented the heuristic for correction of the non-valid results obtained after applying the transformation operators.

Crossover

Two alternative operators are implemented as illustrated en Figure 6:

a) Crosses between two rows of two chromosomes, b) Crosses between two columns of two chromosomes.

The selection of the crossover operators that they will act on the chromosomes it is carried out at random.

Mutation

Two alternative operators are also implemented (see Figure 7).

a)._ Row chromosome mutation. b) Column chromosome mutation.

Obtained results

The numeric results obtained with the implemented evolution program have been represented graphically by means of CAD tools.

For the case presented in this work, exists a "Maximum Visual Area" (MVA). It can be calculated. Considering buildings with a maximum height of 20L (levels), having a 180m x 90m land, knowing the views orientation (side of 180m), the possible MVA is:

MVA = 20L x 180m = 3600Lxm, where L is a generic module in metric measures

For example, for L = 3.00 meters one has:

MVA = (20 x 3.0m) x 180m = 10800 m≤

Once obtained the possible MVA, the

Figure 6: Example of a) Crossover by row and b) Crossover by column

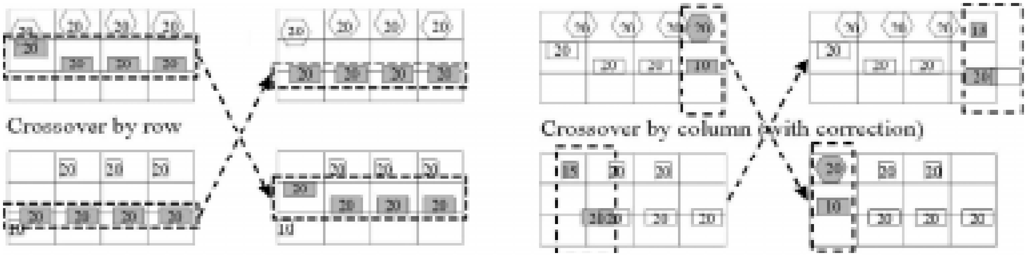
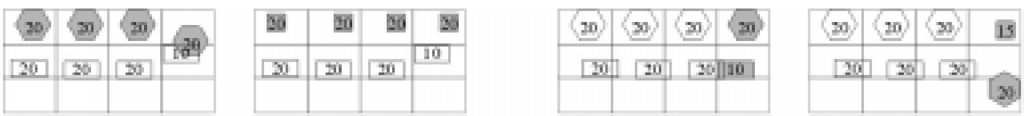


Figure 7: Example of a) Mutation by row and b) Mutation by column



Go to contents 17

implemented program is analyzed. Being a generic and extensive case to specific values of heights of buildings, the value that will be used as MVA will be 3600.

Next, an example is shown using the following data: necessary value of MVA = 3600, Seed = 33. The graphic result obtained to different complexity level in 2D and 3D is shown in Figure 8:

With Fitness = 11066, with a MVA = 3520, and in the Generation = 12792.

In the Table 2 some configurations results are shown. The minimal distance among buildings varies. The probability of the land occupation varies depending on the allowed minimal distance among buildings. As much as higher is the minimal distance among buildings smaller is the probability of occupation of buildings on the land. Otherwise, the number of needed generations will grow to such an amount that any solution could be envisageable (table 2).

Summarizing the previous discussion it can said that in this work some results have been presented obtained by means of the Evolutionary Algorithm implementation of the type Evolution Program. The modelization of the problem has been based on the data structures implementation that contemplates some variables that intervene in the positioning of a building on a land.

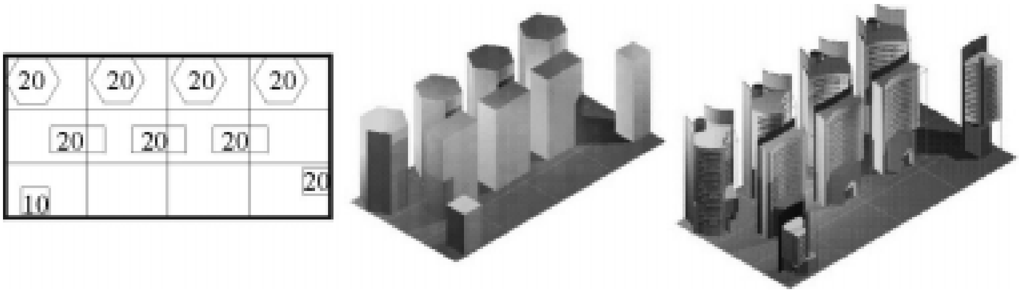
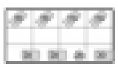
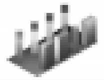
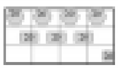
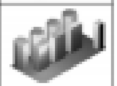
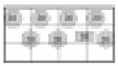
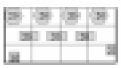
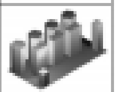


Figure 8: An example to different complexity level in 2D and 3D.

Table 2: Results with distinct dmin, MVA and POccupation in 2D and 3D

	dmin (meter)	MVA (meter x L)	POccupation	Generation	Fitness	Configuration 2D	Configuration 3D
MVA=3500	5	3547	0.80	131	10851.58		
	10	3500	0.60	193	11049.75		
MVA=3600	5	3600	0.80	3739	11179.26		
	10	3600	0.60	12588	11131		

Go to contents 17

References

Barrionuevo, L. F.: 1999, Posicionamiento de volúmenes arquitectónicos mediante algoritmos evolucionistas. In Proceedings of the III Congreso Iberoamericano de Gráfica Digital-SIGRADI, Montevideo, pp. 176-181.

Bentley, P. J.: 1999, An Introduction to Evolutionary Design by Computers, in Peter J. Bentley (editor), Evolutionary Design By Computers, Morgan Kaufmann Publishers Inc., San Francisco, CA.

Bentley, P. J.: 1999b, Aspects of Evolutionary Design by Computers, in Advances in Soft Computing – Engineering Design and Manufacturing, Springer – Verlag, London, pp. 99 – 118.

Combes, L. and Barrionuevo, L. F.: 1999, Distribución espacial de elementos arquitectónicos. In Proc. of the III Congreso Iberoamericano de Gráfica Digital-SIGRADI, Montevideo, pp. 130-133.

Fogel, L. J.: 1963, Biotechnology: Concepts and Applications, Englewood Cliffs, NL, Prentice Hall.

Fogel D., 1992, An Analysis of Evolutionary Programming, Proc. of the 1st Annual Conf. on Evolutionary Programming, San Diego, Evolutionary Programming Soc., San Diego, CA, pp. 43-51.

Michalewicz, Z.: 1995, Evolutionary Computation. In tutorial of the Fifth Scandinavian Conference on AI, Trondheim, Norway.

Michalewicz, Z, et al.: 1996, Evolutionary Algorithms for Constrained Engineering Problems. In Computers & Industrial Engineering Journal, Vol.30, No.2, pp.851-870.

Michalewicz, Z.: 1996, Genetic Algorithms + Data Structures = Evolution Programs, Springer Verlag.