A Preliminary Approach to Allocate Categories of Buildings into Lands based on Generative Design

Ignacio Pérez-Martínez
Dept. of Electr. and Computer Engineering
University of Córdoba, Spain
email: ep2pemai@uco.es

María Martínez-Rojas
Dept. of Economics
University of Málaga, Spain
email: mmrojas@uma.es

J.M. Soto-Hidalgo, *Member, IEEE*
Dept. Comput. Architecture & Comput. Technology
University of Granada, Spain
email: jmsoto@ugr.es

***Abstract*—In comparison to some other economic sectors, the productivity of workers in the construction has not increased much over the last 20 years. There are many approaches to tackle this problem, and one of them is to rethink the design process and introduce automation. In this paper, we propose a workflow that allocates houses on a plot of land. This workflow starts with GIS data and urban planning requirements and ends with a BIM model on that property. A genetic algorithm tackles the design process of allocation of houses by iterating and optimizing a parametric model. In the end, a designer chooses the final solution between the generated range. The degree of automation achieved by the algorithm will be tested by comparing the results of the algorithm with human-made designs.**

***Index Terms*—Generative design, building architecture, visualization software**

I. Introduction

The McKinsey Global Institute (MGI's) published a study on the construction industry in February 2017 and found that the construction industry has an unsolvable productivity problem. While sectors such as retail and manufacturing have reinvented themselves, the construction industry seems to be in a time loop. Global labour productivity growth in construction has averaged only 1% per year over the last two decades, compared to 2.8% in the global economy as a whole and 3.6% in manufacturing as shown in Fig. 1.

Fig. 1. MGI´s in Global productivity

There are many reasons for the continuation of such poor performance, including strict regulations and dependence on public sector demand, informality and sometimes corruption, industry fragmentation and a mismatch in risk allocation and reward. The construction sector should learn from successes in other industries and disrupt the ongoing process of thinking, working and building. In recent years, the rethinking of processes and an increase of automation level has been the key factor of increased productivity. Recently, digital technologies - from 5D BIM (Building Information Modeling) to advanced analysis techniques - have spread rapidly. However, studies indicate that BIM, in particular, has not brought significant productivity gains, although this technology has introduced improvements in the design process.

The idea presented in this paper is to create a tool that automates the assignment of houses on a property. This tool should be useful for different reasons. On one side, it should be useful for real estate to identify land purchase opportunities with above-market returns, and on the other, for the planner as a generative modelling tool that suggests many optimal solutions on the property of the study.

This paper is structured as follows. Section II presents some preliminary concepts related to generative design, while Section III describes the proposal. A case study illustrating the potential of this proposal is presented in Section IV. Finally, Section V contains some concluding remarks.

II. Preliminary concepts: Generative Design

This section summarises some preliminary concepts. Section II-A introduces Generative Design in Space Planning with special focus on Multi-Objective Optimization NSGA-II. Section II-B presents the concept of Computational Design and its tool for Generative Design in Autodesk, "Refinery".

A. Generative Design

In space planning, generative design refers to a method to iterate our designs into optimal solutions [1], [2], [3], [4]. In nature, a similar design process is based on natural selection principles. These principles, applied to an algorithm, define a genetic algorithm. Due to the complexity of architectural models, for this study, we will use the NSGA-II algorithm, which is an algorithm based on genetic principles for multiobjective optimization.

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

1) NSGA-II algorithm: Only four basic operators drive the algorithm. [5] Generation: The algorithm begins by generating a set of designs which form the initial ´´generation".

Selection: Next the algorithm selects which of the initial designs are going to be used to generate the next generation.

Crossover: This is similar to the idea of "breeding" in natural evolution; it recombines the most suitable designs to create a new population of designs.

Mutation: Mechanism to inject new information randomly into the gene pool. Randomly changes the inputs of a random number of children before they enter the next generation.

Plotting the designs relative to two goals define a line called the Pareto optimal front. Fig. 2 illustrates this plotting. Notice that all designs occurring on the boundary are optimal because it is impossible to make any of them better in one goal without making it worse in another one. Any designs found inside the boundary are feasible but non-optimal.

In practice, this front is not always clear and continuous. This model presents some discontinuity every time the configuration of certain parameter changes [6]. It is important to consider these discontinuities when optimizing a model to allow all possibilities to be explored within a given discontinuity, i.e., by fixing the number of zone subdivisions on the plot of land per optimization.

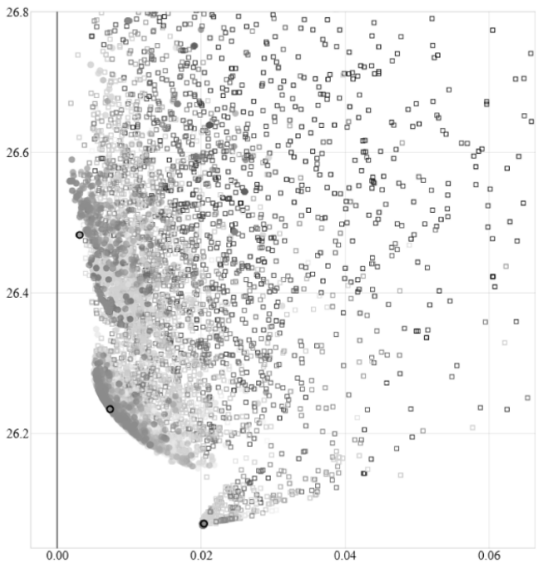

Fig. 2. Discontinuities on the pareto front

B. Computational Design

Visual Programing Language (VPL) is excellent because it allows sequential workflows on a functional basis, and due to the visual approach is more intuitive for designers to develop their workflows. However, VPL provides only limited support in embedding through custom components. It does not provide support for recursion nor the one for object-oriented programming. The solution is to integrate a Python code into Dynamo, so we can use the advantages of the Visual Programming and also avoid the limitations throughout Python.

1) Autodesk Refinery vs Discover: Refinery is an Autodesk beta product for a generative design that allows users to quickly explore and optimize their Dynamo designs. Discover is a similar tool developed by Danil Naggy and runs in Grasshopper from Rhinoceros-McNeel.

After Refinery or Discover go through the generative process, the results are displayed in geometric form and through a series of diagrams or tables. Both are based on the NSGA-II algorithm.

Although Refinery is an easy-to-use product, it allows you to control only a few hyperparameters of the optimization: Design per generation and Number of generations. Mutation = 0.08 can not be modified. Refinery is well suited for the first approach to optimization with NSGA-II in a prototypical way, but not as a final workflow, as it is a "black box" approach.

On the other hand, Discover allows you to control more hyperparameters: Design per generation, Number of generations, Mutation, Elitism and Crossover. Input parameters can be defined as categorical or continuous. Initially Mutation = 0.05 and Elites = 1.

The output of the optimized geometry was to be a BIM model, so we initially used Refinery to remain native in a BIM program, Revit. As the transition from Grasshopper to Revit has become easier since early 2020 with Rhino-Inside, we will continue to develop further in Grasshopper as it provides an open-source environment and an open API. Nevertheless, in this paper, we will present the results achieved in Refinery.

III. Generative design proposal

To develop a workflow that allows the allocation of buildings on plots, we need to consider the following points: GIS data integration, Regulations, Module apartment types, Urban planning, Goals and constraints, and Generative Design workflow.

A. GIS data integration

GIS data download can be automated. OpenStreetMap allows to process *.osm files (free to download from OSM website) with a Python file and generate geometry within Rhino or Revit. So far, the program only looks into the building geometry (if available). However, the OSM data-structure is not made to store 3D data. It is more a sliced representation of the geometry. Therefore, you only can get extrusions by height and construct the roof geometry based on the given type.

B. Regulations

Deep Learning object detection allows reading zoning plan´s information and converts it into constraints of a generative model. Online availability of these regulations allows automation. Europe is nevertheless in a disadvantage against USA or Australia due to its policies on Data Protection. In Europe, unpublished data does not allow certain levels of automation. In contrast, two Startup companies in New York

https://envelope.city/ and Australia https://archistar.ai/ show that this workflow is doable.

C. Modularity

A catalogue of housing units is defined to ensure that generated models meet all conditions and needs necessary for the construction of a real project. Besides, these modules are easy to parameterize and are cost-effective to optimize, which simplifies a preliminary approach of assigning different buildings to plots of land.

In this paper, as example, the TUM München is considered and graphically illustrated in Fig. 3. In this case, we can see how different depths are possible, depending on the maximum ceiling span usability (V1), maximum utilization of the construction (V2) and better proportioned and more flexible furnished rooms, which insure a lot of natural light due to their shallowness (V3 and V4). The sizes of the floor plans are based on the requirement´s catalogue criteria and are coordinated with each other, which allows easy creation of combination and building groups.

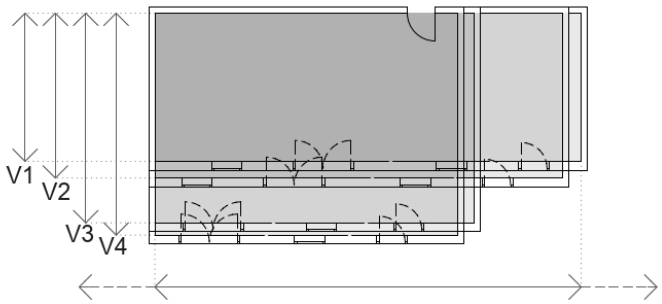

Fig. 3. Housing Units

Common typologies such as block type, external corridor type, central corridor thin type and central corridor thick type are possible in different compositions by packing the housing modules within these four typologies. Fig. 4 shows different building typologies.

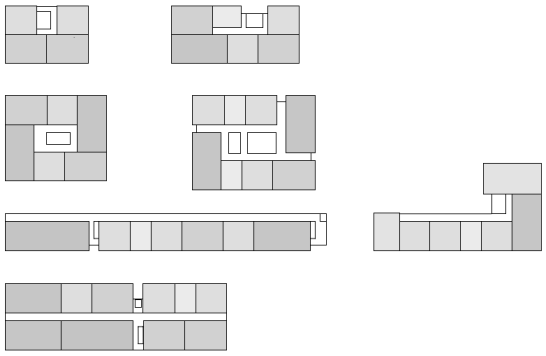

Fig. 4. Building typologies

D. Urban planning

Decoding Spaces is a plugin for Grasshopper from Weimar´s University. They synthesize spatial configurations for street networks, parcels and building volumes. Principles of this plugin [7] could be used to develop a similar workflow. Nevertheless, the state of the workflow relays on the human as a decision-maker for urban planning decisions. The designer needs to define manually which streets, parks, squares, etc. should be prolonged or complemented. Those manual inputs will define constraints on the generative model.

Once the main urban decisions have been made, NSGA-II will optimize further plot subdivision and typology allocation. The generative model will be built using recursive subdivision [6] to divide the plot into many zones. The allocation of the typologies is random and is optimized by maximizing or minimizing the objectives and fulfilling constraints.

E. Goals and constraints

Constraints dictate whether a design is a feasible option at all. Plenty of design problems are defined by several different goals, which may be related to each other in complex, non-intuitive ways. Deciding which design is better than the other is not always clear.

The genetic algorithm will achieve the best performance results by setting some optimization goals. For example: maximizing the built area and compactness, minimizing north-facing apartments and setting a maximum building height as a condition. To judge genetic algorithm design´s relative performance compared to other is usually a balance between visual aspects and best performance rate.

F. Generative Design Workflow

The Generative Design (GD) workflow is displayed in table 1. This table is divided into three sections: Pre-GD, GD and Post-GD, and three columns: Topic, Task and Technique. In the Technique column, the value ”Manual” means Architect, Generative Design Specialist or Designer.

IV. Case study

A. Urban planning competition

The aim of the competition is to create an attractive urban residential quarter [8] (approx. 250 residential units) with a subordinate share of commercial use and high quality of open space. In addition, intelligent, sustainable and flexible housing offers are to be created that respond to different market needs.

B. GIS data

GIS data is downloaded from cadmapper.com and exported to Revit.

C. Parametric Model

1) Urban Planing Concepts: In the book “A Pattern Language”, Christopher Alexander describes patterns that create cities worth living in. Even though he explained several of them, in our code, we will use the most important ones and bring them into our parametric model. They are: 1-Green areas within five minutes walk from each house. 2-Uniform distribution of streets, footpaths, sports fields and common activities in the neighbourhood. 3-Central square or the green area enclosed by buildings typologies while offering views to outside.

TABLE I
GENERATIVE DESIGN WORKFLOW

Pre-GD		
Input/data		
Topic	**Task**	**Technique**
GIS data	GIS into CAD software	Automation
Zoning Plan	Images or PDFs into Metadata	Manual / Deep Learning
Design Decisions		
Update housing´s types	Encode requirements	Manual
Urban planning (parks, squares, typologies...)	Design decision into adjacency list	Manual / Machine Learning

GD		
Run Optimization		
Hyperparameters	Set goals and weights and NSGA-II hyperparameters	Manual
Optimization	Run	Generative Design
Choose option		
Re-run optimization	Readjust hyperparameters and run the optimization again until we do not get better results, then manually select the best option	Manual

Post-GD		
Refine		
To BIM	Optimized option to a more detailed BIM model	Automation
Further refinements	Adapt chosen design to further requirements	Manual

2) Subdivision Algorithms: The plot is divided into sub-plots by recursion. The structure of the algorithm is a binary tree and to control the subdivision we have 4 main variables: `var_n` (number of sub-plots), `var_directions` (to determine whether the subdivision is parallel or perpendicular to the longest side of the sub-plot), `var_splits` (to define which sub-plot is further subdivided) and `var_areas` (target area of each sub-plot).

```
var_n = n
var_directions[n-1][0,1]
var_splits[n-2][0,1]
var_areas[perm(n)]
```

3) Typology placement: The know-how of the city planner flows into the algorithm via the adjacency list. This list contains restrictions and requirements per building typology. During the generative design loop, a particular typology is placed in a specific location with its surroundings (existing buildings, streets, paths, green spaces, topography, orientation, etc.) in a particular way that best meets the requirements specified in the adjacency list.

D. Fitness objectives

Six fitness objectives have been set, each with a design intention:

1. Living space factor; the ratio of living space to the built area -maximize. This parameter reduces the building costs to the same living space.

2. The ratio of total built area to land area -maximize. Real estate usually wants to maximize the built area on a property up to a certain value.

3. Compactness (ratio of facade area to building volume) minimize. The facade is usually an expensive part; reducing its area reduces the costs.

4. Sub-plot area (the difference between target and sub-plot area) -minimize.

5. Sub-plot proportions (deviation of the side zone length from the average side length of the same zone) -Minimize. Similarities in the sub-plots proportion are proven to generate a more cohesive neighbourhood.

6. Variability (number of different typologies on the plot) -maximize. To stimulate a more diverse and livable environment.

E. Evolve

We run the optimization in Refinery three times:
Design per generation = 32
Number of generations = 10
Number of zones = 3,4,5
The time required by Refinery is about 15 minutes per model. Fig. 5 shows this case study where 3, 4 and 5 zones are considered.

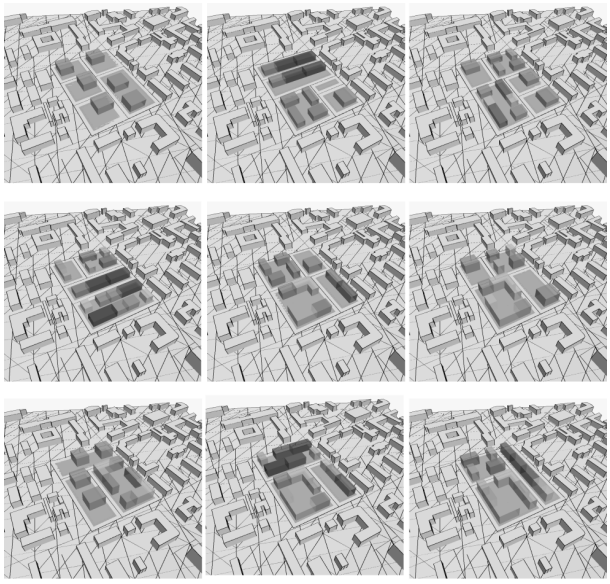

Fig. 5. Case Study: 5, 4 and 3 zones

F. Termination Criteria

Refinery provides a dashboard interface with the hundreds or thousands of generated options and their parameters. By manually filtering the values of the fitness criteria in the dashboard, the user gets a small set of customized solutions. Finally, the user chooses the most appropriate solution by selecting from this set a solution that looks more attractive and easier to build. In this way, the user's subjectivity and expertise are incorporated into the process.

G. Generative Design vs competition´s results

Competition´s awards always show an urban planning concept that rounds off the whole urban solution. In comparison, the results of the GD algorithm look a bit banal, because they may be right concerning the optimized parameters. Still, they do not follow any intention and do not react accordingly to their surroundings. This lack of conceptual understanding of the GD algorithm makes complete automation of the urban design process unrealistic [9]. We can see this development as a design assistant for the designer. An algorithm that explores the plot by proposing many suitable solutions to the designer.

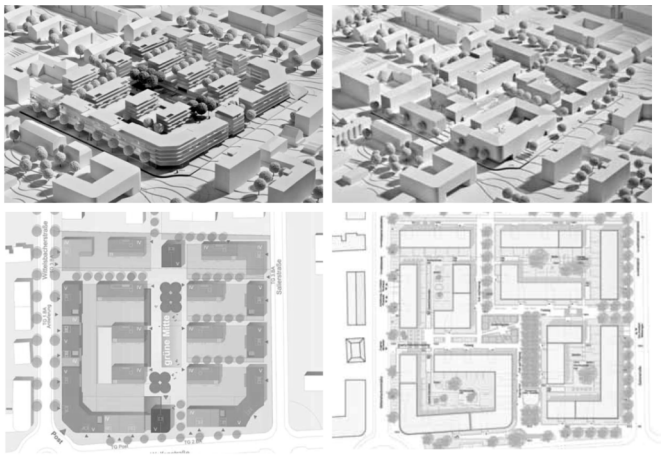

Fig. 6. Competition result

V. Conclusions

The conclusions of both the proposal and the case study show that the lack of conceptual understanding of the generative model optimization does not allow a fully automated process. On this basis, we propose a workflow between humans and algorithms to achieve at least a semi-automated process, and we also reflect on our initial decision to use modules to describe living areas.

A. Singularity vs Modularity

Modular models need fewer parameters to be described, and at the same time, more constraints reduce the number of model possibilities. Therefore, the calculation and optimization of these models are computationally cheaper, and at first, the computational design easily generates modular buildable architectures [10].

However, uniqueness is a significant advantage in architecture: local regulations, GIS data, builders' requirements, budget and other parameters define particular needs that are not always adequately met when the basis is a fully modular design. Besides, modularity has not so far proven to be an advantage on the construction site by reducing costs, and therefore uniqueness is still required by real estates.

Future development in these fields should include graph matching algorithms, [11], [12], machine learning algorithms [13], [14], etc. [15], [16], [17] which provide methods for more flexible and complex models that will eventually generate unique and buildable architectures.

B. Human vs Algorithm

There are three types of parameters to describe a model: Hard parameters: constructed area, heights, compactness, the number of units, etc. These parameters are defined by numbers and are easy to deduct and optimize, either by humans or by algorithms.

Simulation parameters such as energy consumption, sunlight, thermal comfort, paths, etc. must first be simulated and then optimized. They are optimized more efficiently by machines.

Soft parameters: beauty, art concepts, sensations, urban planning decisions or design concepts are parameters that cannot be defined by numbers. Even today, computers can hardly approach these topics and are therefore fields set by humans.

C. Algorithm´s evaluation

To evaluate the algorithm, we will show the results to a group of urban planners and architects. The advantages and disadvantages of the GD workflow should be compared to a common architectural process to ensure that the process adds value.

References

[1] D. Nagy, D. Lau, J. Locke, J. Stoddart, L. Villaggi, R. Wang, D. Zhao, and D. Benjamin, "Project discover: An application of generative design for architectural space planning," in *Proceedings of the Symposium on Simulation for Architecture and Urban Design*. Society for Computer Simulation International, 2017, p. 7.

[2] E. Rodrigues, "Automated floor plan design: generation, simulation, and optimization," Ph.D. dissertation, 2014.

[3] D. Benjamin and D. Nagy, "Generative design for architecture," May 17 2018, uS Patent App. 15/812,885.

[4] V. Singh and N. Gu, "Towards an integrated generative design framework," *Design Studies*, vol. 33, no. 2, pp. 185–207, 2012.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[6] D. Nagy, L. Villaggi, D. Zhao, and D. Benjamin, "Beyond heuristics: a novel design space model for generative space planning in architecture," 2017.

[7] R. Schaffranek and M. Vasku, "Space syntax for generative design: On the application of," in *Proceedings of the ninth international space syntax symposium*, 2013.

[8] C. Alexander, *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[9] D. Lobos and D. Donath, "The problem of space layout in architecture: A survey and reflections," *arquiteturarevista*, vol. 6, no. 2, pp. 136–161, 2010.

[10] M. Mirahmadi and A. Shami, "A novel algorithm for real-time procedural generation of building floor plans," *arXiv preprint arXiv:1211.5842*, 2012.
[11] R. Schaffranek, "Parallel planning: An experimental study in spectral graph matching," in *Proceedings of the 10th International Space Syntax Symposium*, 2015.
[12] D. Nagy, L. Villaggi, J. Stoddart, and D. Benjamin, "The buzz metric: A graph-based method for quantifying productive congestion in generative space planning for architecture," *Technology— Architecture+ Design*, vol. 1, no. 2, pp. 186–195, 2017.
[13] D. Newton, "Generative deep learning in architectural design," *Technology— Architecture+ Design*, vol. 3, no. 2, pp. 176–189, 2019.
[14] P. Merrell, E. Schkufza, and V. Koltun, "Computer-generated residential building layouts," in *ACM SIGGRAPH Asia 2010 papers*, 2010, pp. 1–12.
[15] D. Camozzato *et al.*, "A method for growth-based procedural floor plan generation," 2015.
[16] C. M. Herr and R. C. Ford, "Adapting cellular automata as architectural design tools," 2015.
[17] Z. A. A. A. Baki, H. A. Abdulbaqi, and Y. M. Mohialden, "A novel interior space planning design based on mdb-fa method."