The conception of parameters and constraint based models in architectural projects: the case of Lelé's pivot domes

Abstract

Following the increasing interest form architects on the application of generative algorithms in conception of projects, many questions have been raised about teaching this particular way of thinking about forms. One of the main goals on learning the subject is to understand how an idea of a construction element can be translated in to the rules of an algorithm. Others are: to introduce concepts of programming necessary to implement the algorithms and to place this methods in the scope of disciplines involved in the construction activity. This paper propose a example based strategy to discuss this three crucial points, using the pivot domes designed by Brazilian architect Lelé as an object of study.

Keywords: Parametric Models, Geometric Constraints, Generative Algorithms.

1. Introduction

The concepts, techniques and applications of generative algorithms has gain, in the passing of the last decades, a great amount of attention form architectural practices, researchers and teachers [1]. Many of the global leading *bureaus* in the field of Architecture have invested in implementing and maintaining a programming department to collaborate in the creative process of building design [2]. The range of influence of the computer-aided design (CAD) extrapolate the aspect of an applied tool and reaches the aesthetic theories [3] [4] and, considering the integration with computer-aided manufacturing (CAM) systems, the construction activities [5]. The generative algorithm contributed with it's share in this process.

In the academic world, the assimilation of the methods, tools and knowledge involved in the understanding of the generative design paradigm, grants, from representing a great challenge, a latitude of possibilities and an opportunity to rethink the relations among disciplines involved in the process [5] [6].

One of the main issues on teaching generative design to novice users is to make them understand how an initial idea can be translated in rules that generate forms. The understanding of how this forms could be generated is the first step into this different way of approaching the project activity that this algorithms embrace. It's important to remember that this generation of forms in not an objective in itself, but a tool to assist the answering of questions raised form the point of view of the Architecture as an interdisciplinary domain. In that sense, the building of effective generative algorithms also relies in the understanding of witch characteristics of the building or construction element may be explored and

tested by the process. How different parameter will act together in a way that the variant forms generated by those will establish a set of possible solutions, each one with particular characteristics, to be disposed or adopted by the user.

This paper proposes a example-based strategy to introduce the generative algorithms student in the definition of the rules that governs a conceptual model of an architectural element based on parameters and constraints. In the interdisciplinary scope, the geometrical constructions used, concepts on list manipulation based programming and aesthetics aspects of the generated forms are also presented and discussed.

2. Generative design and generative algorithms

Algorithms are defined as a finite set of rules that describes the steps do solve an specific problem [7]. The technical draw applied in the design of architectural plans are, indeed, algorithms [8, chap. 3] , and the sequential operations with compass, rule and straightedges, used to make any of the geometric constructions, describes it's rules. Words like "perpendicular", "parallel" and "tangent" summarize entire algorithms in the debates that orbit the project activity. The development of CAD technology, started by providing the ability to reproduce this drawing algorithm to the software, and them evolving from that to a "wider graphic vocabulary available to designers, together with a more elaborate syntax—in all, a richer and potentially more expressive graphic and spatial language" [9, chap. 15].

Generative Design, in a broad definition, is the application of algorithms or a " rule based process through which various potential design solutions can be created " [10] and the generative algorithms are the logical engine propelling that design strategy in the computational context. As stated before, all the technical draw constructions are algorithms. The variety an variants of buildings designed by those methods are unmeasurable. The generative design techniques differs from the traditional methods for transforming the methodology of work. The generative systems provides a degree of automaticity to the process, enabling the user to evaluate a greater number of possible solutions to the architectural problem in a shorter amount of time than it wold be possible to do by the traditional methods.

In Fig. 1 [11] a flowchart illustrates the process of creating an operating an generative algorithm in a elucidative way. The real shift promoted by the generative algorithms are in the process of electing the form to be constructed. Starting with an idea of what should be the constructed object, as an abstraction that is translated in a set o rules. An algorithm is implemented in a programming language, code or a visual programmable ambient, obeying the rules and a simulation of the model is generated by setting the parameters values and running the algorithm. The designer evaluates the output and, by changing the parameters values, the rules and/or the code, generates different outputs and chooses the final solution among those. The construable artifact appear only when the designer elect the proposal to be build [12].

Many computational methods where applied in the creation of forms driven by generative algorithms: shape grammars [13] [14] [15] [16], genetic algorithms [17] [10] [18] [19], cellular automata [20], simulated annealing [21] [22] [23], tabu search [24], swarm intelligence [25] [26] and parametric models [27] [28] [29] [30] among others.

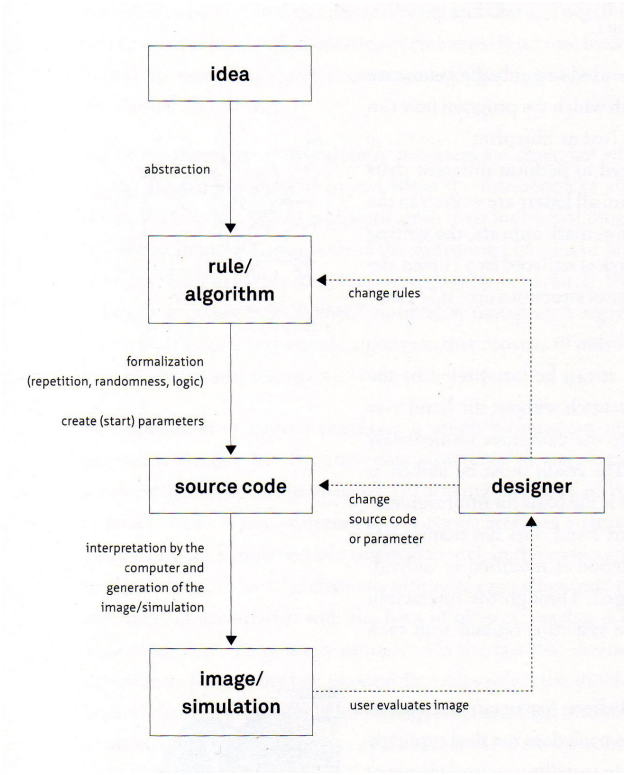


Figure 1: Flowchart of a generative algorithm

3. Parameters and constraints based models (PCM)

All generative algorithms, as well all the computer processing, are based in parameters, although the parametric design differentiate from the others by emphasizing the manipulation of parameters [12]. For drawing a right prism in a cad system, the user has to input values for the width, height and depth. In a parametric design system, every one of this inputs can be modified interactively, and the prism dimensions will be automatically updated in the drawing.

Among many computer simulation methods to implement a generative design system, the constraint solver, since its introduction, by Pro/Engineer in the 1980's and its adoption by most of the available cad systems [31], has gain attention by the intuitive and fluid manipulation of the forms calculated by this kind of solver. The methods relays on constraint problems satisfaction, consisting of, in a finite set of variables, each of those assigned to a finite domain and having the values that can be taken together restricted by constraints, finding a combination of this values that satisfies all the constraints [32]. The definition of parameters and constraint based models here proposed is a way of emphasize both the parameters and the constraints that conducts a generative design process.

The creation and manipulation of geometric forms in an ambient that enable the use of a constraint solver is tightly related to geometric theorem proving [31]. Geometry is

a cornerstone in Architecture and construction since the early History of those fields [2], and the teaching of generative algorithms should explore the geometric knowledge from an alternative point of view.

4. Object of study

The Brazilian architect João Filgueiras Lima, better known by the nickname of Lelé, is one of the most important and celebrated architects of the country [33]. Considered by Oscar Niemeyer a " great master of architecture ", and by Lucio Costa " the architect in whom art and technology meet and merge " [34]. His works reveals a strong interest in rationalization in the use of materials, an outstanding development of prefabrication and environmental integration, adopting daylight and air circulation innovatory solutions [35] [36].

Figure 2: Sarah Hospital auditorium, Rio de Janeiro - RJ - www.sarah.br

As an object of study, the pivoting domes used in the auditorium of the Sarah Kubitschek Hospital in Rio de Janeiro (Fig. 2) and in two buildings of the proposal for the Labor Justice Courthouse complex, in the city of Salvador - Ba. In Figure 3, a 3D model of the complex presents the dome of the auditorium open in the right and the dome of the main court closed in the left.

The first box in the flowchart in Figure 1 represents the idea of the family of forms to be generated. There are didactic advantages on presenting image references to illustrate the idea behind the generative algorithms to be build. In stead of describing an abstract idea of a goal, the architectural examples clarify important aspects of the intended solution. The application of this elements in real projects, the aesthetic composition that emerged from the variant forms of the domes and their relation whit the buildings that they are a part of, and the fact that many aspects of the algorithm can be visually identified could be also mentioned. A discussion about an "one sentence definition of the idea" could be conducted in class, or this sentence could be proposed by the teacher. The summarized idea should

Figure 3: Labor Justice Courthouse Complex, Salvador - Ba source: Instituto Habitat archives

point the students attention to the goals of the algorithm in study. "A dome, divided in petals that pivot around axes on their base" is a good example of the idea's definition.

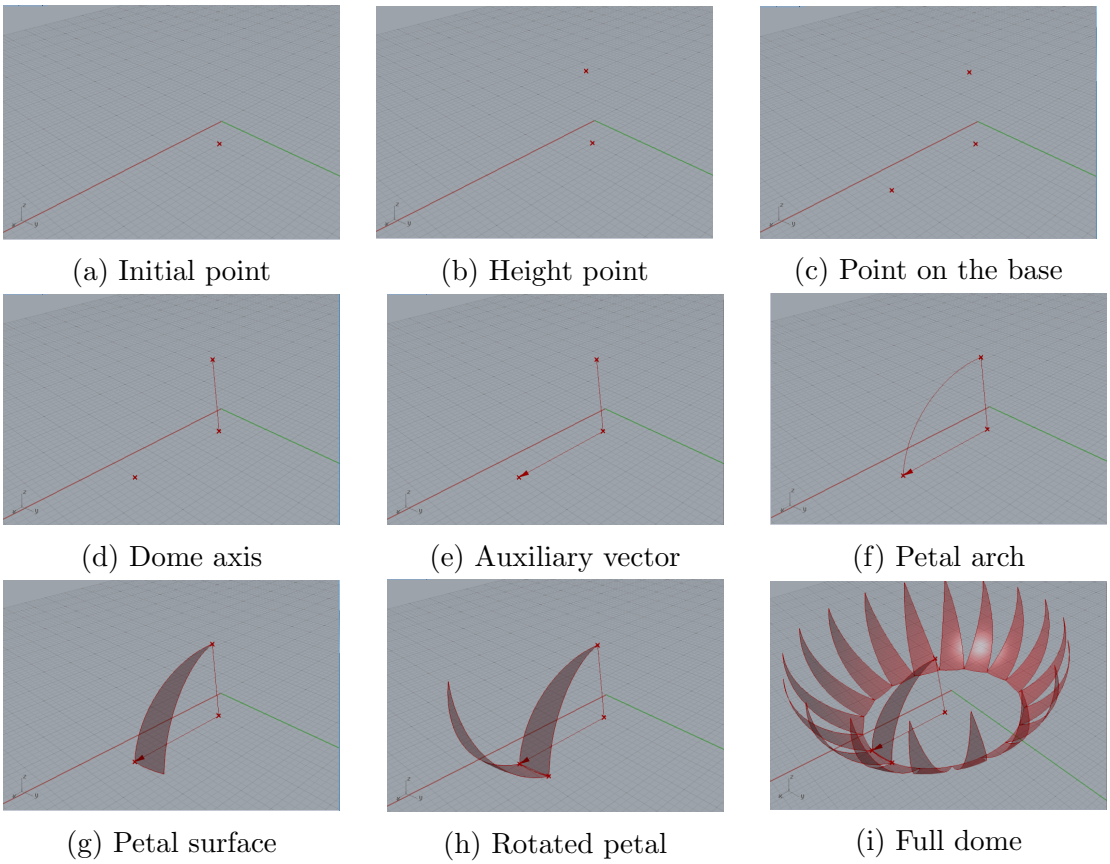
5. Defining the parameters

Starting from the idea, the pivoting domes presented in Figures 2 and 3 should be translated in algorithms that can automatically draw variations of those forms. Analyzing the domes presented in the Figures 2 and 3, the similarities and differences between the elected solutions can be listed:

- all the domes are , when closed, spherical caps;
- the number of petals in the presented examples are different;
- the base radius and height of the spherical cap, in the presented examples, are different;
- the base of the petals are on the base of the spherical cap;
- the top of the petals are on the maximum height of the sphere.

Based on this visual observations, many elements of the algorithm can be defined. The parameters, for instance, can be the **base radius**, the **center point** of the base, the maximum **height** and the **number of petals**. Another parameter should be added to the algorithm to simulate the pivoting movement of the dome: a **rotation angle** parameter.

Is important to note that other parameters could replace the elected ones to draw the same forms. The radius of the full sphere, the difference between the cap height and the center of the sphere, for example, could replace the maximum **height**, the base **center point** and the cap base **radius** generating the same geometries. Although equivalent results can be accomplished by the both strategies, its a good practice to think about what characteristics of the construction element are intended to be explored and how the parameters to be manipulated will fit in the process. Assuming that, in the presented example, the main characteristics to be explored are: the area of illumination and ventilation provided by



(a) Initial point (b) Height point (c) Point on the base

(d) Dome axis (e) Auxiliary vector (f) Petal arch

(g) Petal surface (h) Rotated petal (i) Full dome

Figure 4: Steps of the one petal algorithm

the dome when opened and the aesthetic relation between the radius and the maximum height. That assumption justifies the proposed parameters as they directly operate on the characteristics on focus.

6. Geometric constructions and constraints

In a geometric constraint solver environment, the definition of the rules (constraints) can only be conceived by planning the geometric constructions. The first modeling strategy presented in this paper is: to model one petal, define a rotation axis (associated whit the rotation angle parameter) and copy it around a circle.Figure 4 shows steps of the algorithm that will be discussed below.

Starting from the **center point** (fig. 4a) of the base circle and making tow copies of the point: one in the z axis, by the distance inputed in the cap height parameter, named as the **height point** (fig. 4b); and the other in the x axis, taking the value of the base radius as the module of the translation vector, referred as a **point on the base** (fig. 4c). A line between the **base center** and the **height point** is also drawn, and will be called the **dome axis** (fig. 4d).

The next step on drawing one petal is the arch on one side of it. As the arcs are drawn independently, the rules that define them should constraint the closed dome as a spherical cap. One of the ways to effectively constraint that construction is to make the arch tangent, in the height point, to a line or vector parallel to a direction defined as a line drawn from the base center to the point on base. In figure 5 that idea is illustrated, showing that only the blue arcs, that are tangent to the green vectors on the height point, form together a circle arch.

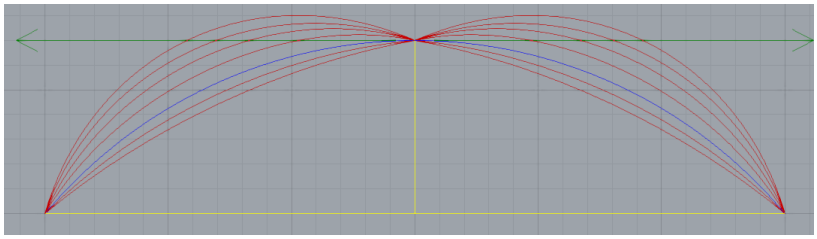


Figure 5: Constraining the arcs as part of a spherical cap

In that step of the algorithm, an opportunity to debate some aspects of the geometric constructions applied in the process can be taken. More precisely, how this arcs can be defined by the classical geometric drawn and how this construction was translated to the computational environment.

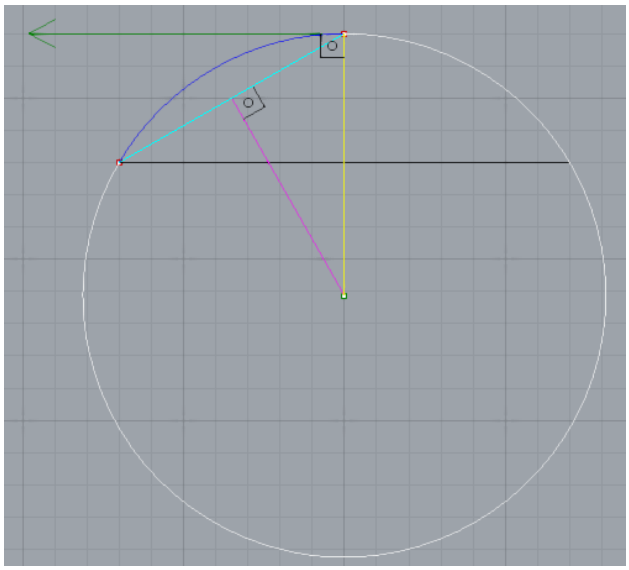


Figure 6: Illustration of the traditional method for drawing the tangent arch

The traditional steps to draw a tangent arch is illustrated in Figure 6. The construction is based on two properties of circular arcs. One is that the tangent line in every point is

perpendicular to the radius in that same point. The other is the most basic property of a circle, that every point on it is equidistant to the center.

Starting whit the two red points and the tangent vector, represented by the green line in Figure 6, and drawing a line perpendicular to the vector in the point that the vector tangency the arch, is safe to state that this perpendicular line (yellow) contains the center of the circle. The next step is to find a point on the perpendicular line that is equidistant to the two points of the arch. This can be done by drawing a segment that connects the tow points (cyan) and draw a perpendicular line in the middle of it (magenta). The point found by the intersection of the line perpendicular to the tangent vector and the line drawn in the middle point of the line that connect the two known points of the arch is equidistant (as are the sides of an isosceles triangle) to this points. So, with a compass centered in that intersection and starting form one of the known points to another, the arch can be defined (blue).

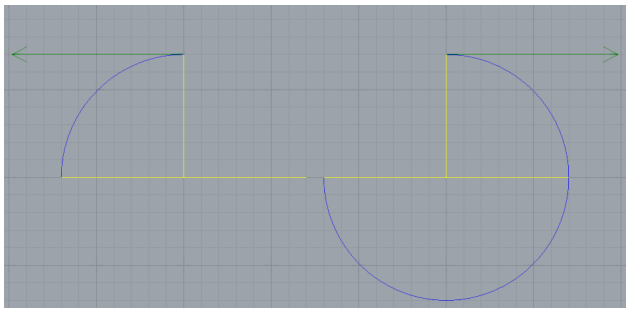


Figure 7: Illustration of the computational method for drawing the tangent arch

Is important to note that the withe arch in Figure 6 is also tangent to the vector and contains the two known points. The decision to draw one arch or the other, in that traditional method is purely human made, relaying on the understanding of the intended construction. Drawing an arch perpendicular to a line considers the line's orientation but not its sense. In the computational environment, when this method of drawing an arch based on two points and a line was implemented, the sense of the line is used to define the sense in witch the arch should be drawn. Figure 7 illustrated the arcs generated by the method with vectors that have the same direction and opposite senses.

In the proposed algorithm the reference vector is defined by an ordinated segment from the **base center** to the **point on the base** (fig. 4e). The command that generates an arch by tow points and a vector have, as inputs, the **height point**, the **point on the base**, and the vector of the tangent in the **height point** (fig. 4f). Constructing the arch this way will constrain the closed dome to a spherical cap whit the height and base radius assuming the values defined by the respective parameters.

The petal is created as a ruled surface, generated by a rotation around the **dome axis** in an angle named as the **petal angle** (fig. 4g). Dividing a 360° angel ($2 \times \pi$ in radians) by the **number of petals** the **petal angle** is numerically calculated.

To simulate the pivoting movement that opens the dome, a rotation axis connecting the

base points of the petal needs to be created. Task that could be performed by rotating the **point on the base** around the **dome axis** taking the **petal angle** as the rotation amount. A line is drawn between the two points and is called the **pivot axis**.

The Petal surface is rotated around the **pivot axis** in an angle defined by the **pivot rotation** parameter (fig. 4h). To complete the algorithm and generate the **full dome** (fig. 4i), a polar array of the **rotated petal** is generated in a 360° angle around the **dome axis** taking the number of petals as the number of the elements of the array.

7. The code

Although any programming language could virtually be used to implement the code, the CAD software Rhinoceros 3D (version 5.11) and it's plug-in Grasshopper (version 0.9.0076), a visual algorithm interface and a constraint solver, where chosen for the intuitive nature of programming thought the connection of graphical components, and the fact the forms will be created inside a CAD system able to conduct architectural projects.

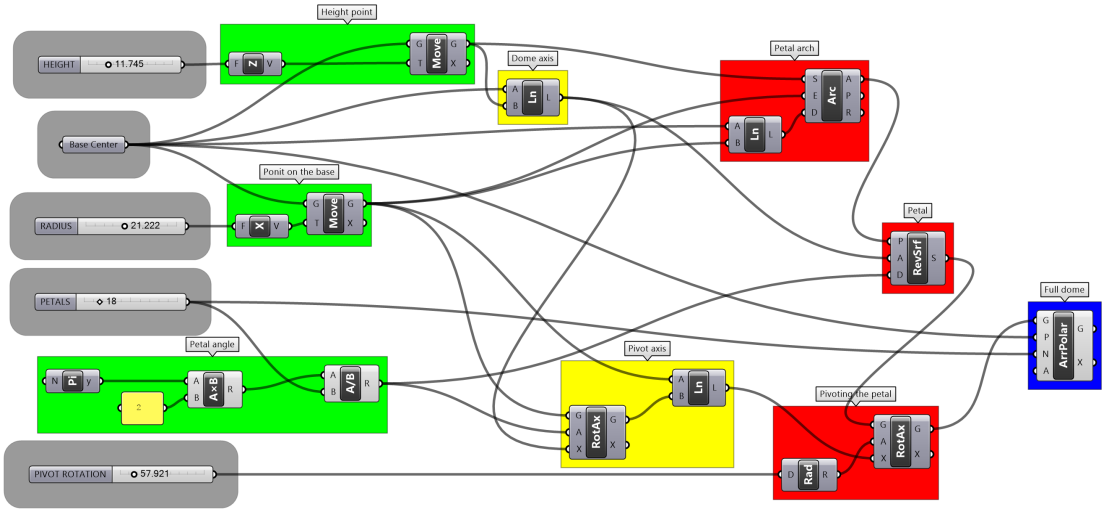
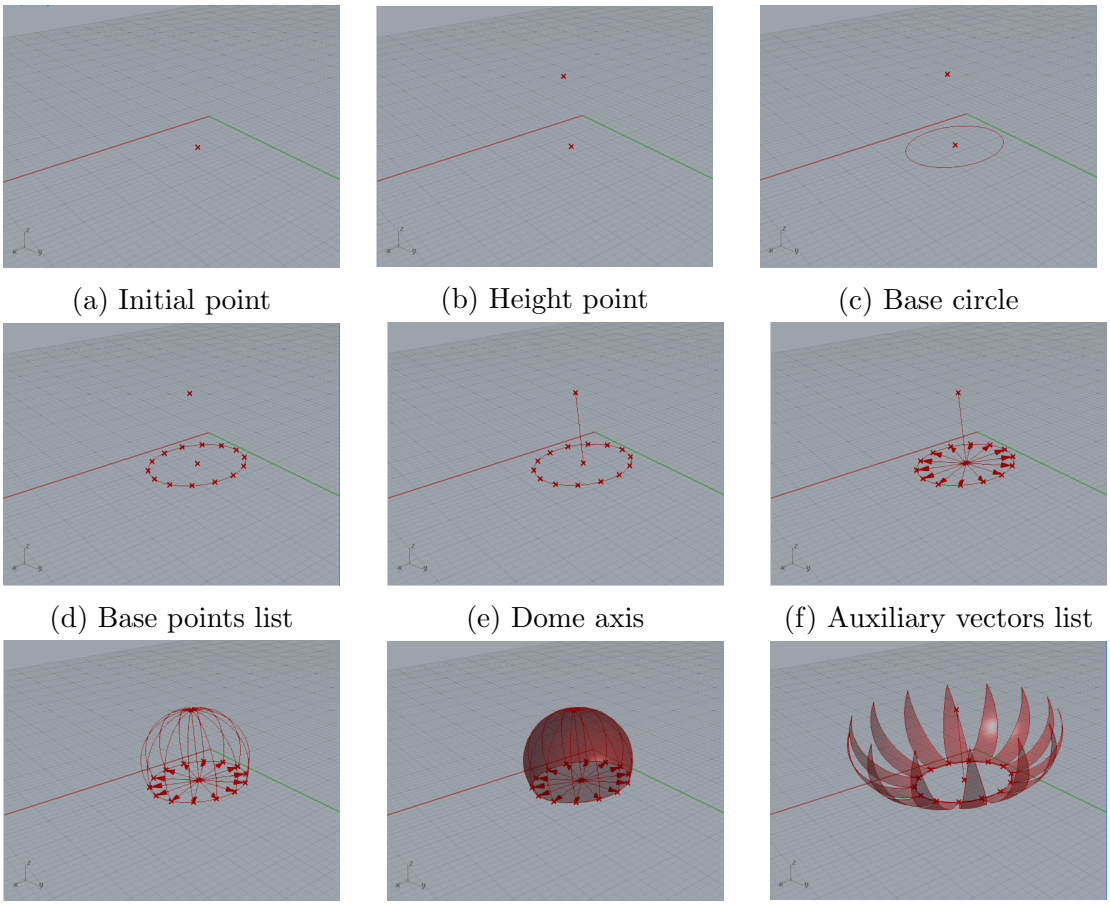


Figure 8: One petal algorithm

Figure 8 is the code implemented in the programming interface and also a flowchart of the algorithm. The data flows from left to right through the wire like connections between the components.

In the grey boxes are the parameters. From top to bottom: **height**, **base center**, **radius**, **number of petals** and **pivot rotation**. the top green box generates the **height point**, the middle green **the point on the base**, the bottom green calculates the **petal angle**. The yellow box on top generates the **dome axis** taking the **base center** and the **height point** as inputs, the bottom yellow outputs the **pivot axis**. The **auxiliary vector** and the **petal arch** are calculated in the top red box, the middle red box perform the



(a) Initial point (b) Height point (c) Base circle

(d) Base points list (e) Dome axis (f) Auxiliary vectors list

(g) Petals arcs list (h) Petals surfaces list (i) Full dome

Figure 9: Steps of the list manipulation based algorithm

revolution that generated de **petal surface** and the bottom red rotates it around the **pivot axis**. The blue box is responsible for the polar array operation that generates the **full dome**.

8. List manipulation based algorithm

The one petal algorithm presented in the previous sections satisfies the conditions and objectives of the proposed generative algorithm. Is an efficient and linear way of defining the rules. The list manipulation based programming strategies are current related do the Lisp programming paradigm proposed by John McCarthy in 1959 [37]. Tightly associated whit the artificial intelligence field [38], the list manipulation has also many applications in CAD environments.

Taking as example a line creation command having as inputs points A and B, drawing a line that goes from start point A to end point B. If the second input is substituted by a list of points B, C, D, E, the result of this list operations e a list of lines $\bar{AB}, \bar{AC}, \bar{AD}$ and $\bar{AE}$, created by the application of a single line command. If two lists are used as inputs

$(A, B, C, D, E$ and $F, G, H, I, J)$ the result will be a list of lines between the equivalent points in each ordinated lists $(\bar{AF}, \bar{BG}, \bar{CH}, \bar{DI}$ and $\bar{EJ})$.

A variant implementation of the dome algorithm based on list programming is presented in this section to exemplify and introduce the concept. In the one petal algorithm example the steps generates one element at time, in the list manipulation based algorithm lists of elements will be created in a way that corresponding elements of every petal will be created simultaneously and all the petals will be generated and modified in parallel.

Figure 9 shows the steps of the list manipulation based strategy, and starts from the **base center** (fig. 9a) and creates the **height point** (fig. 9b) in the same way it is created by the previous example. A circle is drawn using the **base center** and **base radius** parameters as inputs (fig. 9c). The point on base of the first algorithm is replaced by a **base point list** (fig. 9d), all of them constricted to the radius distance from the **base center**. The dome axis (fig. 9e) is generated as it was in the previous rule set. In stead of a single vector, an auxiliary vectors lists (fig. 9f) is generated by connecting the **base center** to the **base point list**. The **petal arcs list** (fig. 9g) is drawn taking the **height point** as the start, the **base points list** as the end and the **auxiliary vectors list** as the tangents directions and sense.

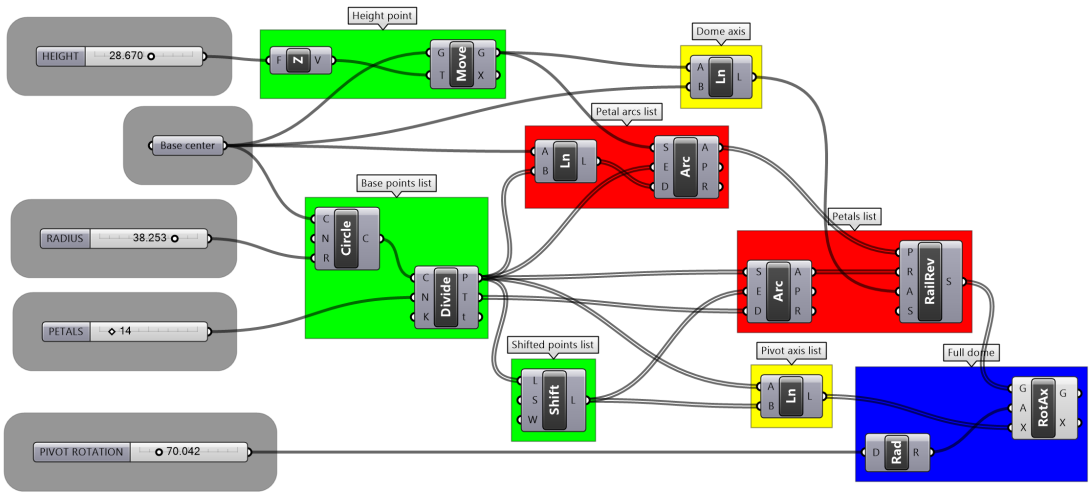


Figure 10: List manipulation based algorithm

Up to this point the ordinated lists are used to perform commands that operates on respective elements of the lists and no explicit manipulation was required. A simple but important operation on the **base points list** is part of the proposed rules, and is used to generate the forms and exemplify the applications of this programming methodology. The idea is to extract the first element of a list and append it back on the end. In a proposed points list represented as A, B, C, D, E, this operation will output B, C, D, E, A. Generating lines between the equivalent points in the two lists will result in a list of lines that connects

adjacent points: ($\bar{AB}, \bar{BC}, \bar{CD}$ and $\bar{DE}$). In Figure 10 this operation is represented in the bottom green box, taking the **base points list** (middle green box) and outputting the **shifted points list**.

The **shifted points list** is directly used in two steps of the algorithm. The first is to create a **pivot axes list** by drawing lines between the **base points list** and the **shifted points list** (fig 10 bottom yellow box). The second is to generate auxiliary arcs to the **petal surface list** creation (fig 10 red bottom box). The middle green box on Figure 10 generates the **base points list** by dividing the **base circle** in the given **number of petals**, the grasshopper curve division component also outputs a list of the tangent vectors of the curve in the division points. Arcs are drawn whit start point provided by the **base points list**, end points defined by the correspondent elements in the **shifted points list**, and the sense and directions of the equivalent tangent vectors. This auxiliary arcs are used in replace of the **petal angle** in the first algorithm to define the revolution amount angle of the petals surface (fig 10 red bottom box) around the **dome axis** (fig 10 yellow top box).

The petals surface list (fig. 9h) are rotated around the correspondent **pivot axes list** in the angle provided by the pivot rotation parameter, drawing the **full dome** (fig. 9i).

9. Analysing the algorithms outputs

The flowchart in Figure 1, after the idea is established, the rules are planned, the code is implemented and a first is generated by the first set of parameters; the designer is in charge of evaluate the result, reset the parameters and/or modify the code and continuously repeat this operations until a constructive form ends up being elected as the one to be build.

Figure 11 shows a set of solutions generated exclusively by modifications made over the values of the parameters. One of the decisions that the designer should made is the size of the **base radius**, strongly related to amount of light and dynamics of the air that should circulate in and out of the building it is attached on. This evaluation could relay on the designer's experience and intuition or calculations that could be performed whit ou without the aid of computational analyses tools. Since the variations of the forms are automatically generated, a tool that analyses 3D models tends to be more efficient for the task.

Not only the building itself, but the characteristics of the environment should be considered in this choosing process, but some relations between the parameters result in relevant changes in the outputted geometries. In the first row of Figure 11 the domes have bigger values for the **base radius** than the ones of the **height** parameter, the second line shows equal values and in the third row the **radius** is smaller than the **height**. This fact could be easily demonstrated geometrically, but can also be experimented by the parameters modifications.

The aesthetic perception of the different simulations can be observed in the images of figure 11 but is better understood when compared to the domes and respective buildings of Figures 2 and 3. The first one presents a bigger value for the **height** parameter than the **radius**, producing a fluid relation between the curves of the building's forms and the dome as a finishing element. An inverted configuration of parameters in the relations between this values will generate a brusque or even flat interruption on the forms composition. In

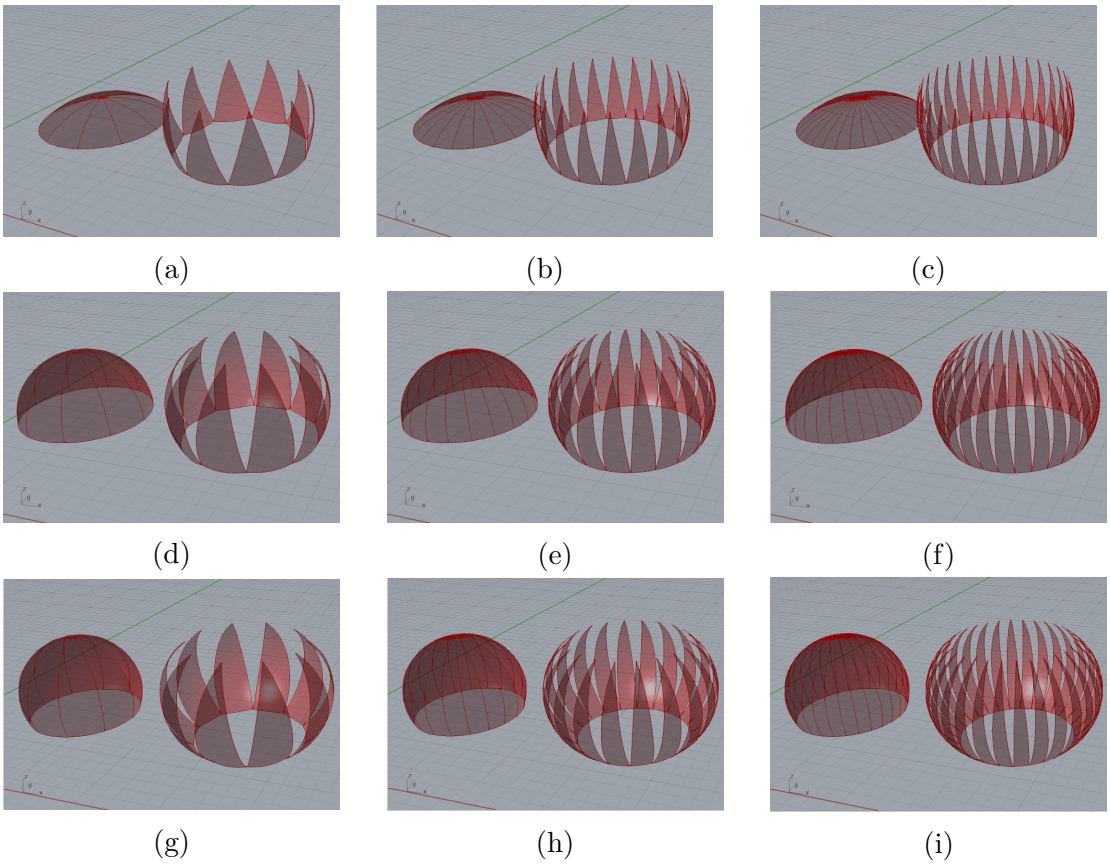


Figure 11: Variant domes created by the algorithms

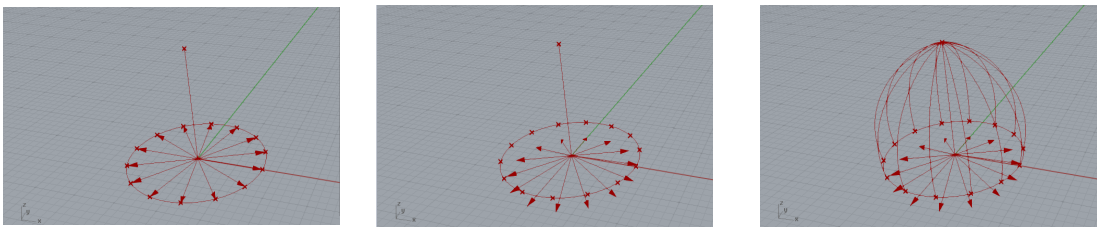
the second, the domes adopts a more discreet elevation from it's base, dialogging with the almost flat roof solution of the buildings.

Wile the quantitative investigations can be numerically evaluated, the qualitative aspects, as the aesthetics, can only be approached and addressed by the raising and answering of some loosely defined questions. The simple analysis proposed aims to exemplify how this subjectives requisitions can be handled by the generative algorithm's methodology.

10. Modifying the code

Modifications in the parameters are the fastest way of generating alternative results in a generative design system. The results registered in Figure 11 could be generated by both of the algorithms presented, but the flowchart in Figure 1 foresees that the code and the rules could be also modified in the scope of this work methodology. To illustrate this possibility some changes in the list manipulation based algorithm are proposed to enable the code to create not only spherical cap domes, but also geometries derived from lancet arcs.

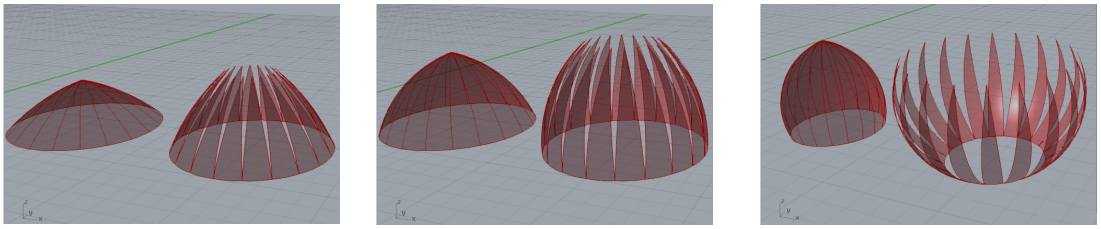
Few modifications are need to complete this task. The steps presented in Figure 9 are followed in the exact order until the **auxiliary vectors** are created (fig 9f). Then the vectors suffer a rotation defined by a new parameter called **tangent rotation** and the **petals arcs**



(a) Auxiliary vectors list (b) Rotated tangents list (c) Petals arcs lists

Figure 12: Steps of the lancet dome algorithm

list is created taking this new vectors as inputs. After that, **the petals surface** list and the **full dome** are created by the same procedures represented in figures 9h and 9i.



(a) (b) (c)

Figure 13: Steps of the lancet dome algorithm

If the tangent rotation parameter is equal to zero, the results are constrained to geometries similar to the ones presented in Figure 11, else they assume the pointed end of a lancet arch as Figure 13 illustrates.

The visual code presented in Figure 14 presents this changes in comparison with Figure 10. The **tangent rotation** parameter is added (bottom grey box) and the **auxiliary vectors** are rotated around the z axes of a list of planes created with origin in the **base center** x axes along the direction and sense of the **auxiliary vectors** and y axes following the **dome axis** oriented line, this rotated tangents list is inputed as the vector required by the **petals arcs list** creation component (top red box).

11. Conclusion

The proposed exercise discussed along this article succeed in approaching the basics aspects of the generative design methodology, encompassing all the possible variants of the flowchart in Figure 1, explaining the most important geometric constructions used, providing an overview of some programming procedures and presenting how the election of constructive forms could be performed in both qualitative and quantitative aspects.

The one sentence definition of the idea purposely omits the fact that, in that stage of the analysis of the domes references, they are all spherical caps. If that was stated, the modification of the algorithm in section 10 will also modify the idea. Although this modification is not described by the flowchart (fig. 1) it is indeed a possible movement in

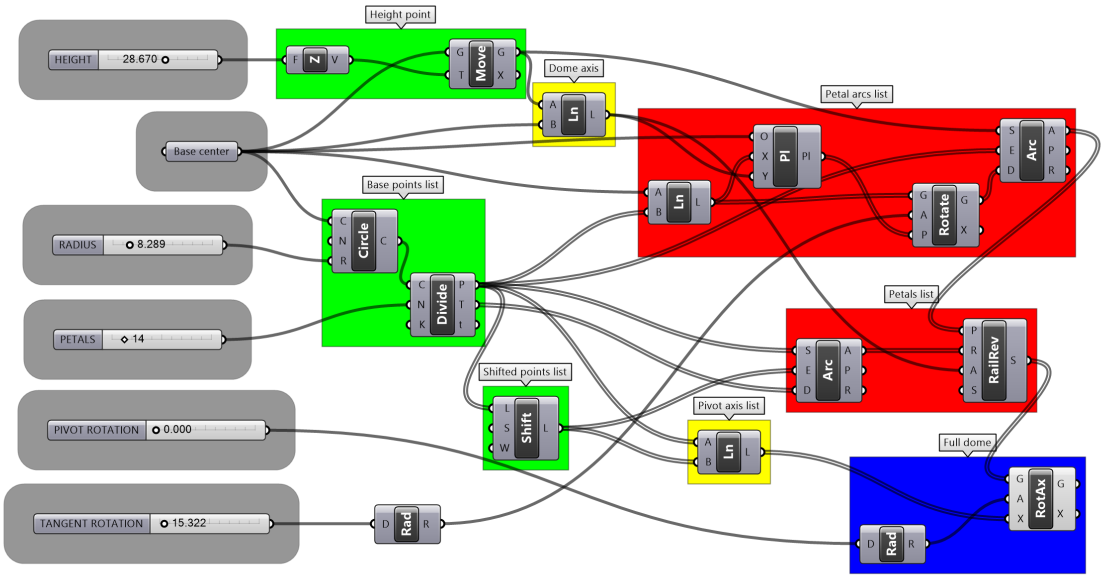


Figure 14: Lancet arch algorithm

the work process of a generative algorithm. Since code reuse is a common programming activity, the case of a stated characteristic of an idea been modified along the way should not me treated as an issue, but as a natural possibility.

The interdisciplinary nature of Architecture meets an also interdisciplinary domain that a comprehensive understanding of the generative design systems demand. The scope of all the different disciplines interrelated in this exercise was not by far covered, but the effort of bringing those notions together in a coherent context should guide future works of presenting the subject and a virtuous way of establishing new discussions on the secular field of Architecture.

References

[1] S. Krish, A practical generative design method, Computer-Aided Design 43 (1) (2011) 88–100. doi:10.1016/j.cad.2010.09.009.
URL `http://linkinghub.elsevier.com/retrieve/pii/S0010448510001764`

[2] C. Ceccato, The Master-Builder-Geometer, in: C. Ceccato, L. Hesselgren, M. Pauly (Eds.), Advances in Architectural Geometry 2010, Springer, 2010, pp. 9–14.

[3] R. Oxman, Theory and design in the first digital age, Design Studies 27 (3) (2006) 229–265. doi:10.1016/j.destud.2005.11.002.
URL `http://linkinghub.elsevier.com/retrieve/pii/S0142694X05000840`

[4] A. Picon, Ornament and its users: from the Vitruvian tradition to the digital age, Le Visiteur (17) (2011) 176–180.
URL `http://dash.harvard.edu/handle/1/12638041`

[5] B. Kolarevic, Architecture in the digital age: design and manufacturing, London, 2003.

[6] C. Ceccato, Others (Eds.), Towards Teaching Generative Design in Architecture, Springer, 2010.
[7] D. E. Knuth, Art of Computer Programming, Volume 1: Fundamental Algorithms, Pearson Education, 1997.
URL http://books.google.com.br/books?id=x9AsAwAAQBAJ
[8] K. Terzidis, Algorithmic Architecture, Taylor & Francis, 2006.
URL http://books.google.com.br/books?id=yT7NXhZYepwC
[9] W. J. Mitchell, World's Greatest Architect : Making, Meaning, and Network Culture, MIT Press, Cambrige, 2008.
URL http://books.google.com.br/books?id=DxszH9whFSIC
[10] E. Fasoulaki, Integrated Design, Ph.D. thesis, MIT, Cambrige (2008).
[11] H. Bohnacker, B. Gross, J. Laub, C. Lazzeroni, Generative Design: Visualize, Program, and Create with Processing, Princeton Architectural Press, Princeton Architectural Press, 2012.
URL http://books.google.com.pe/books?id=tSS9uAAACAAJ
[12] I. G. DIno, Creative Design Exploration By Parametric Generative Systems In Architecture, Metu Journal of the Faculty of Architecture (2012) 207–224doi:10.4305/METU.JFA.2012.1.12.
[13] G. Stiny, J. Gips, shape Grammars and the Generative Specification of Painting and Sculpture, in: Petrocelli (Ed.), The Best Computer Papers of 1971, Auerbach, 1972, pp. 125–135.
[14] G. Stiny, W. J. Mitchell, The Palladian grammar, Environment and Planning B 5 (1) (1978) 5–18.
URL http://www.envplan.com/abstract.cgi?id=b050005
[15] J. P. Duarte, A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira, Automation in Construction 14 (2) (2005) 265–275. doi:10.1016/j.autcon.2004.07.013.
URL http://www.sciencedirect.com/science/article/pii/S0926580504000810
[16] B. Tepavcevic, V. Stojakovic, Shape grammar in contemporary architectural theory and design, Facta universitatis - series: Architecture and Civil Engineering 10 (2) (2012) 169–178. doi:10.2298/FUACE1202169T.
URL http://www.doiserbia.nb.rs/Article.aspx?ID=0354-46051202169T
[17] L. Caldas, L. Norford, A Genetic Algorithm Tool For Design Optimization, in: Media and Design, Acadia'99, Salt Lake City, 1999, pp. 260–261.
[18] L. Troiano, C. Birtolo, Genetic algorithms supporting generative design of user interfaces: Examples, Information Sciences (2012) 1–19doi:10.1016/j.ins.2012.01.006.
URL http://linkinghub.elsevier.com/retrieve/pii/S0020025512000242
[19] L. G. Caldas, L. K. Norford, A design optimization tool based on a genetic algorithm, Automation in Construction 11 (2) (2002) 173–184. doi:10.1016/S0926-5805(00)00096-0.
URL http://linkinghub.elsevier.com/retrieve/pii/S0926580500000960
[20] C. M. Herr, T. Kvan, Adapting cellular automata to support the architectural design process, Automation in Construction 16 (1) (2007) 61–69. doi:10.1016/j.autcon.2005.10.005.
URL http://www.sciencedirect.com/science/article/pii/S0926580505001494
[21] B. Ceranic, C. Fryer, R. Baines, An application of simulated annealing to the optimum design of reinforced concrete retaining structures, Computers & Structures 79 (17) (2001) 1569–1581. doi:10.1016/S0045-7949(01)00037-2.
URL http://linkinghub.elsevier.com/retrieve/pii/S0045794901000372
[22] V. M. Correia, C. a. M. C. A. Mota Soares, Refined models for the optimal design of adaptive structures using simulated annealing, Composite Structures 54 (2-3) (2001) 161–167. doi:10.1016/S0263-8223(01)00085-X.
URL http://www.sciencedirect.com/science/article/pii/S026382230100085X
[23] L. Lamberti, An efficient simulated annealing algorithm for design optimization of truss structures, Computers & Structures 86 (19-20) (2008) 1936–1953. doi:10.1016/j.compstruc.2008.02.004.
URL http://linkinghub.elsevier.com/retrieve/pii/S0045794908000448
[24] J. Bland, G. Dawson, Tabu search and design optimization, Computer-Aided Design 23 (3) (1991) 195–201. doi:10.1016/0010-4485(91)90089-F.
URL http://www.sciencedirect.com/science/article/pii/001044859190089F

[25] G.-C. Luh, C.-Y. Lin, Y.-S. Lin, A binary particle swarm optimization for continuum structural topology optimization, Applied Soft Computing 11 (2) (2011) 2833–2844. doi:10.1016/j.asoc.2010.11.013.
URL http://linkinghub.elsevier.com/retrieve/pii/S1568494610002905

[26] M. Yahya, M. Saka, Construction site layout planning using multi-objective artificial bee colony algorithm with Levy flights, Automation in Construction 38 (2014) 14–29. doi:10.1016/j.autcon.2013.11.001.
URL http://www.sciencedirect.com/science/article/pii/S0926580513001945

[27] M. Turrin, P. von Buelow, R. Stouffs, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, Advanced Engineering Informatics 25 (4) (2011) 656–675. doi:10.1016/j.aei.2011.07.009.
URL http://linkinghub.elsevier.com/retrieve/pii/S1474034611000577

[28] T. Fischer, M. Burry, J. Frazer, Triangulation of generative form for parametric design and rapid prototyping, Automation in Construction 14 (2) (2005) 233–240. doi:10.1016/j.autcon.2004.07.004.
URL http://linkinghub.elsevier.com/retrieve/pii/S0926580504000780

[29] L. Lachauer, H. Jungjohann, T. Kotnik, Interactive parametric tools for structural design, in: Proceedings of the IABSE-IASS Symposium 2011, London, UK, 2011.
URL http://www.schwartz.arch.ethz.ch/Publikationen/Dokumente/InteractiveTools.pdf

[30] S. Milena, M. Ognen, Application of Generative Algorithms in Architectural Design, in: Advances in Mathematical and Computational Methods, 2010, pp. 175–180.
URL http://www.wseas.us/e-library/conferences/2010/Faro/MACMESE/MACMESE-27.pdf

[31] C. M. Hoffmann, R. Joan-Arinyo, A brief on constraint solving, Computer-Aided Design and Applications 2 (5) (2005) 655–663. doi:10.1080/16864360.2005.10738330.

[32] R. Barták, Theory and practice of constraint propagation, in: In Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control, 2001, pp. 7–14.

[33] S. Philippou, The primitive as an instrument of subversion in twentieth-century Brazilian cultural practice, arq: Architectural Research Quarterly 8 (3-4) (2004) 285–298. doi:10.1017/S1359135504000302.

[34] M. C. Ferraz, G. Latorraca, I. G. Ferraz, E. S. de Freitas, S. Birkinshaw, K. Szabó, João Filgueiras Lima, Lelé, Arquitectos brasileiros - Brazilian architects, Editorial Blau, 2000.
URL https://books.google.com.br/books?id=hHJEAQAAIAAJ

[35] G. Campagnol, Positive Distraction and the Rehabilitation Hospitals of João Filgueiras Lima, HEALTH ENVIRONMENTS RESEARCH & DESIGN JOURNAL 8 (1) (2014) 199–227.

[36] A. A. Maciel, B. Ford, R. Lamberts, Main influences on the design philosophy and knowledge basis to bioclimatic integration into architectural designThe example of best practices, Building and Environment 42 (10) (2007) 3762–3773. doi:10.1016/j.buildenv.2006.07.041.
URL http://www.sciencedirect.com/science/article/pii/S0360132306003052

[37] J. McCarthy, Recursive functions of symbolic expressions and their computation by machine, Part I, Communications of the ACM (April) (1960) 1–34.
URL http://dl.acm.org/citation.cfm?id=367199

[38] G. L. Steele, R. P. Gabriel, The Evolution of Lisp, ACM, New York, NY, USA, 1996, Ch. The Evolut, pp. 233–330. doi:10.1145/234286.1057818.
URL http://doi.acm.org/10.1145/234286.1057818