A 3-Dimensional Architectural Layout Generation Procedure for Optimization Applications : DC-RVD

Ioannis Chatzikonstantinou
Yaşar University
i.chatzikonstantinou@yasar.edu.tr

A procedure for generating 3-dimensional spatial configurations for optimization applications, termed Dimension Constrained Rectangular Voronoi Diagram (DC-RVD), is presented in this paper. The procedure is able to generate a non-overlapping configuration of spatial units in 3-dimensional space, given a string of real values. It constitutes an extension and adaptation of the Rectangular Voronoi Diagram generating procedure, found in the work of Choi and Young (1991). An extensive description of the procedure, with the relevant pseudocode is included in the paper. The procedure is tested in a stochastic optimisation-based decision support environment. Testing is done using a case study of a medium-sized family house. The result indicate promising performance.

Keywords: *Optimization, Layout, Representation*

INTRODUCTION

The problem of spatial configuration is concerned with finding suitable locations for a set of interrelated objects that meet design requirements and maximize design quality according to design preferences. It is a general problem that applies in many fields of science and engineering. Specifically for architectural design, the problem is of central significance. The spatial configuration of a building or complex is a design aspect that crucially determines it's success.

However, the problem of spatial configuration is very challenging. In order to tackle it, designers have up until recently relied on heuristics and rules-of-thumb, or guidelines developed through personal experience, which helped them produce suitable layouts in the most common design scenarios. While such approaches may be helpful in practice, they do not really address the problem adequately. The reason is that the problem is irreducible to following pre-existing rules of thumb and heuristics (Loemker 2006). As such, different, more general approach is required.

Computational optimization techniques have been applied to layout problems, as an alternative to the traditional methods. A popular category of algorithms are Stochastic Optimization (SO) algorithms. SO algorithms have gained a lot of traction in the research community, because of their flexibility and efficiency. SO Algorithms are able to deal with problems that we have very limited specific knowledge about, are able to explore solution spaces that are highly irregular and fragmented, and to handle multiple, conflicting goals and multiple constraints. This features make them a good candidate for tackling design problems related to spatial layout, since it can be proven that the objective space of layout problems is highly fragmented and fractal-like (Cagan et al. 2002). On the other hand, the computational requirements of SO algorithms are high. However, with today's increase in computing power, the use of SO algorithms is justified if the problem at hand is complex. Stochastic optimization includes categories of algorithms such as Genetic Algorithms (GA), Simulated Annealing (SA), Partice Swarm Optimization (PSO) and more.

Many SO algorithms work by iteratively improving upon a single or multiple solutions at the same time. This happens through a continuous loop of altering decision variables and evaluating resulting solutions. While evaluation is performed on the actual design itself, optimization algorithms perform variations by altering values in a serial representation of the solution, also known in Evolutionary Computation terminology as Genotype. Each solution is then generated according to these values, resulting in what is frequently termed Phenotype. The nature of the representation and the generative process plays a significant role in the efficiency of the optimization process. A representation language's adequacy, as Cha and Gero mention, is a critically important factor (Cha and Gero 1998).

AIM OF THE STUDY

Recognizing the importance of the layout-generating procedure in spatial configuration problems, this study proposes a new procedure for generating 3-dimensional architectural layout configurations, which is suitable for use in computational optimization scenarios. The proposed method is termed Dimension Constrained Rectangular Voronoi Diagram (DC-RVD). The procedure allows generating 3-dimensional non-overlapping configurations based on rectilinear volumes, using an encoding comprising of a series of real-valued parameters.

The proposed method extends and complements the procedure presented in Choi and Kyung (1991) which has been termed Rectangular Voronoi Diagram (RVD), and which itself is an adaptation of the well known Voronoi subdivision method. The RVD method tackles one issue that occurs frequently when dealing with the spatial configuration problem, namely that of overlapping spaces. Instead of treating overlaps as a constraint, which turns out to be a difficult problem to solve, RVD re-distributes the contested space, allowing the overlap to be resolved.

The proposed procedure leads to a smooth objective function landscape that may be easier for an optimization algorithm to traverse. Decision variables represent design parameters such as positions and widths, which correspond to meaningful real-world quantities. This allows meaningful interpretation and analysis of data resulting from an optimisation process.

The paper is structured as follows: In Section 3. We present a review of approaches to the Building Configuration problem, focusing on how layouts are represented and generated. In Section 4., the RVD Algorithm from Choi and Kyung is briefly outlined. In Section 5., the adaptation of the algorithm for application in building configurations is discussed. In Section 6., the revised algorithm is presented. In section 7., the case study is presented. Section 8. discusses the results and Section 9. concludes the study.

EXISTING WORKS

There exist a wide variety of works on the problem of automatically identifying optimal architectural layout configurations. The work of Lobos and Donath (2010), presents a good overview of the most prominent approaches. Here we are going to briefly mention a few, focusing on the layout/configuration representation.

Jo and Gero (1998), present a binary encoding scheme, which encodes placement of spatial units within a predefined boundary, by sequentially encoding their positions along a path. The method is used to solve the Liggett problem (Liggett, 1985) of locating departments within a multi-storey building. The encoding they propose uses two bits to encode the possible movements of a "cursor" to neighboring cells in a grid. The spaces are placed sequentially as the cursor moves on the path specified by the encoding.

Elezkurtaj and Franck (1999) make use of a Genetic Algorithm to search for suitable ground plans for architectural applications. The process of genotype-phenotype translation is as follows: First, the outline of the building is specified. Second, the list of rooms to be fitted into the outline and the proportions preferred are entered. Third, the functional scheme of organization and access is specified. While their work is novel, no details are given as to the specifics of the encoding scheme.

In the work of Michalek and colleagues, (2002), stochastic optimization is interweaved with user input, in order to accommodate second-order criteria and preferences. In the layout representation they describe, rooms, hallways, doorways (Accessways), and boundaries are all represented as combinations of orthogonal rectangular Units. Units are represented as a point in space and the perpendicular distance from that point to each of the four walls. This model has more variables than necessary to describe the shape; however, it allows an optimization algorithm to change the position of a Unit independently without affecting its size. Although this model increases the problem dimensionality, it offers a lot of flexibility to make the best design moves at each step of the optimization (Michalek et al. 2002).

Loemker (2006), has formulated the layout problem as a constrained optimization problem, with the objective of maximizing areas, subject to constraints regarding satisfaction of spatial relations between units. The principle of the geometric model adopted was the representation of rooms as rectangular units. The concept was similar to that of Michalek et al. (2002), but differing in that the representation adopted describes a rectangular unit through a reference point, a length and a width dimension.

Kamol and Krung (2005) have used Mixed Integer Programming techniques to address the 2-d architectural layout problem. They formulate functional and dimensional constraints (such as size, proximity and overlap) into linear functions, and make use of a mixed integer programming solver, GLPK (GNU Linear Programming Kit), in order to identify optimal solutions. They conclude with promising results, however the requirement for linear functions is constraining the expression of complex objectives. The encoding of layouts is simply based on x,y origin point coordinates, and widths and heights of spaces.

In the study of Yeh and colleagues, the architectural space assignment problem has been addressed by the use of Annealed Neural Networks. Yeh used an Annealed Hopfield Network, that combines the speed of convergence of a Hopfield net and the global search characteristics of Simulated Annealing. They apply their method on the problem of locating a range of facilities in a hospital building. The problem includes multiple objectives and a constraint, but these are handled in a single-objective fashion, as such, a considerable amount of parameer tweaking (e.g. weights, constraint penalties etc.) is required. They conclude that while the algorithm is efficient, future research is needed to address the issue of obtaining suitable parameter values. (Yeh et al. 2006)

THE RECTANGULAR VORONOI SUBDIVISION METHOD

Choi and Kyung have discussed the Rectangular Voronoi Diagram (RVD) method, and applied it to the problem of digital VLSI circuit design, yielding promising results (Choi and Kyung, 1991). The method takes it's name partially from the well-known Voronoi diagram. A Voronoi diagram is a method of subdividing space into regions, where each point of a region is closest to one of several predefined seed points. In this method, the resulting regions may have arbitrary shape, depending on the distribution of the seed points. While the Voronoi diagram is being routinely used in architectural design to derive interesting forms and structures, it's application in deriving architectural layouts is relatively limited, due to the irregularity of the shapes resulting from the process.

The characteristic difference of RVD from a classical Voronoi diagram is that, as a result of the RVD process, regions of space of exclusively rectangular shape are produced. This is an important property, because it makes it much more applicable to building layout design than the Voronoi diagram, which produces arbitrary shapes.

The steps of the RVD method can be summarized as follows:

1. A set of points P is considered in 2 dimensions, each of which represent the center of mass for a corresponding rectangle.

2. For each point in p, two subsets are created, Sx and Sy, each of the remaining points ps is placed in one of them according to the following rules:

$$\begin{cases} S_x & if\,|p_x - p_{Sx}| > |p_y - p_{Sy}| \\ S_y & otherwise \end{cases} \quad (1)$$

3. The rectangular areas corresponding to points are derived by determining the middle line between each point p and the points closest to it in all four directions. For edge points, a predefined outline may be considered as a border. As an example, the corresponding rule for determining the left edge of all rectangular spaces may be staed as follows:

```
repeat for each element r {
 Lr = Loutline
 repeat for each element e in Sx{
  Lm = left_midline(r, e)
  Lr = max(Lr, Lm), if Ce < Cr
 }
}
```

where Lr, Le symbolize the left edges and Ce, Cr the centers of the spaces.

The result of the process is a series of rectangular regions that form subdivisions of space. The resulting configuration needs to be bound by a rectangle in order to obtain the limits of the outer regions. An example is shown in figure 1. Essentially, the way the RVD method functions eliminates the need for using overlap constraints for ensuring configuration feasibility. The method instead constrains spatial units to avid overlapping completely. This, of course, has a constraining effect on the area of the spatial unit, but such a constraint, is claimed in this study, is easier to satisfy.

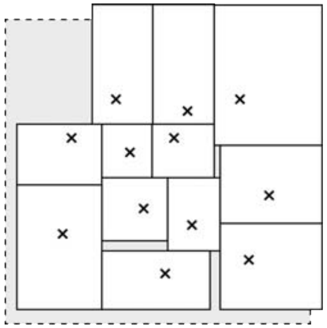


Figure 1
An arrangement resulting from the application of the RVD procedure. The cross symbols represent the centres resulting from the x,y coordinate encoding.

In the study of Choi and Kyung (ibid. 1991), a force-directed packing process was used to optimize the layout with respect to the positions of the seed points. In the present study, this process is substituted in favor of a Stochastic Optimization algorithm. As can be seen from he results, this offers the required flexibility for dealing with the complex objectives of a building configuration scenario.

ADAPTATIONS AND EXTENSIONS

The RVD method presents several advantages that make it attractive for application in building layout design. It makes use of a compact, continuous variable encoding; it offers parameters with high semantic value; and it requires few calculations for generating the configuration from the encoding. However, there are also drawbacks to the method, namely the possibility of gaps appearing within the arrangement and the inability of generating configurations that do not follow a rectangular outline, which should be addressed if it is to be applied in architectural design. Furthermore, the RVD method works in 2-dimensions, but for architectural layout problems, an approach that considers 3-dimensional layout would be preferable.

To address these shortcomings and make the RVD method more applicable to architectural layout problems, a new method, resulting from adaptations and extensions of the original, is proposed.

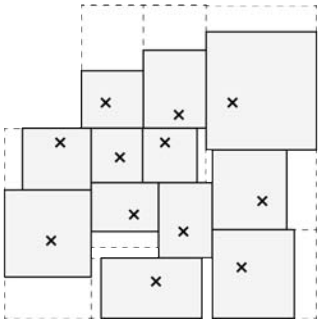


Figure 2
An example of applying the DC-RVD procedure, using spatial unit dimensions to constrain the unit size. In this case, encoding includes x,y, coordinates of the centres, as well as width and height of the shapes.

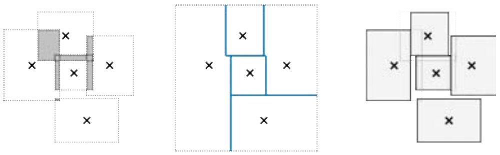


Figure 3
Spatial overlaps, RV regions and resolution of overlaps in the DC-RVD procedure

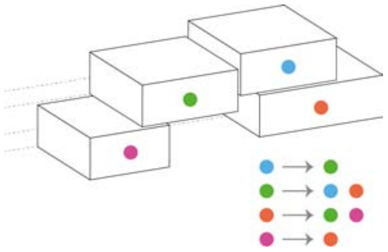


Figure 4
Neighbor relations in a simple case of vertically distributed spatial units

As a first modification, predefined element dimensions are incorporated in rule #3. The new rule will only constrain spaces' dimensions if the closest midline to a neighboring space is found to be closer that the space's original value for that dimension. The inclusion of dimensioning information extends the set of required decision variables for the description of a spatial unit by two, it's length and width. Reiterating the above example of finding the left border, with the new rule:

```
repeat for each element r {
  Bd,r = Bd,original
  repeat for each element e in Sd {
    Bd,m = midline(r, e)
    Bd,r = max(Bd,r, Bd,m), if Ce < Cr
  }
}
```

It should be noted here that constraints are not symmetric for each side of the space, as such it's positioning may be implicitly altered by the constraining process. An example of the procedure applied in 2-d is shown in figure 2. An example of overlap resolution, including spatial segmentation, is available in figure 3.

Secondly, the algorithm is modified to support generation of three-dimensional configurations. The intuition behind the proposed procedure is to constrain each of the volumes only for those of it's neighbors with which a vertical intersection is present. By doing so, the space is only constrained as much as required in order for it not to overlap with it's neighbors. To achieve that, we create separate sets of neighbors for each unit and perform the constraint operation there:

```
repeat for each element r {
  Bd,r = Bd,original
  repeat for each element e in Sr,d {
    if vertical_intersection(r, e) {
      Bd,m = midline(r, e)
      Bd,r = max(Bd,r, Bd,m), if Ce <
       ↪ Cr
    }
  }
}
```

where vertical_intersection would be a function to detect whether the two spaces share some interval in the z (vertical) axis. This way of extending the RVD algorithm ensures that spaces of arbitrary dimensions (albeit still rectilinear) may be accomodated, e.g. vertical circulation spaces, such as staircases, double-height spaces etc. The formation of neighbour relationships during the process is illustrated in figure 4.

Thirdly, the issue of leftover space within the arrangement is addressed. The issue may be either explicitly treated (by employing some post-processing step), or left to be treated implicitly as part of the optimization process, considering that leftover spaces negatively affect the performance of a configuration. With explicit treatment, leftover spaces may be redistributed to neighboring spatial units. However, in the case study that follows, leftover spaces are left to be treated implicitly during the optimization. It turns out that such an implicit treatment offers satisfactory results.

OVERVIEW AND PSEUDOCODE OF PROPOSED ALGORITHM

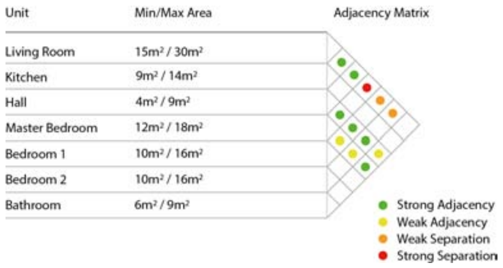
Having summarized the extensions to the RVD algorithm, we may now write down the complete proposed algorithm in the form of pseudocode:

```
repeat for each element r {
  repeat for each element e excluding
   ↪ r {
    Sr,l += e, if rx - ex > |ry - ey|
    Sr,r += e, if ex - rx > |ry - ey|
    Sr,b += e, if |ex - rx| < ry - ey
    Sr,f += e, otherwise
  }
}
repeat for each direction d (among
 ↪ left (l), right (r), front (f) and
 ↪  back (b)) {
  repeat for each element r {
    Bd,r = Bd,original
    repeat for each element e in Sr,d
     ↪ {
      if vertical_intersection(r, e) {
        Bd,m = midline(r, e)
        Bd,r = max(Bd,r, Bd,m), if Ce
         ↪ < Cr
      }
    }
  }
}
```

After the algorithm has finished, we end up with a configuration of non-overlapping spaces with areas equal or less to the predefined ones. As a final step, assignment of functions to spaces may be allowed, based on, e.g., sorting according to a separate list of real-valued decision variables, one for each space. In this case, the number of decision variables is increased by one for each space. Function assignment allows in some cases radical alterations to take place in the design (Merrell et al. 2010).

APPLICATION IN A STOCHASTIC OPTIMIZATION-BASED DECISION-SUPPORT SYSTEM

In order to investigate applicability and performance, the proposed mehod was used to tackle a design problem: that of designing a simple detached family house, with basic design goals. The brief includes a total of seven spaces: a living room, a kitchen, a bathroom, a hall and three bedrooms. Site constraints are minimal. The site area is 15x15 meters. The objective is to select appropriate positions and sizes of the spatial units, so that their area is maximized, subject to constraints regarding connectivity and proximity of the spaces, as well as minimum areas. The list of spaces, desired areas and connectivity requirements are shown in figure 5.



Unit	Min/Max Area
Living Room	15m² / 30m²
Kitchen	9m² / 14m²
Hall	4m² / 9m²
Master Bedroom	12m² / 18m²
Bedroom 1	10m² / 16m²
Bedroom 2	10m² / 16m²
Bathroom	6m² / 9m²

Adjacency Matrix

	Living Room	Kitchen	Hall	Master Bedroom	Bedroom 1	Bedroom 2
Kitchen	🟢					
Hall	🟢					
Master Bedroom	🔴		🟢			
Bedroom 1	🟠		🟢	🟡		
Bedroom 2	🟠		🟢	🟡		
Bathroom			🟡	🟢		

🟢 Strong Adjacency
🟡 Weak Adjacency
🟠 Weak Separation
🔴 Strong Separation

Figure 5
Labels, minimum and maximum areas and adjacency matrix for the spatial units included in the case study

A Genetic Algorithm was used for optimisation. The decision variables were the positions, widths and lengths of the spatial units. Separate function assignment was not used. The vertical position of the spatial units was encoded as a real-valued variable, so they could be positioned anywhere between floors. Three objectives were formulated. The first one is to maximize the area, prescribed by the problem definition, for three main spaces: Living Room, Maser Bedroom and Kitchen. The second one was to minimize cost. The third one to maximise either proximity or separation between desired spaces.

The system has been modelled in the Grasshopper parametric design program. DC-RVD was modelled using simple mathematical and geometrical components. The objectives and constraints were also modelled in the same way. A diagram of the parametric model is available n figure 6. For optimization, the NSGA-II Genetic Algorithm was used. NSGA-II is a multi-objective constrained genetic algorithm, developed by Deb and colleagues (Deb et al. 2002). It is a well established optimization algorithm that has seen widespread use in a wide spectrum of applications. An implementation of NSGA-II for Grasshopper, Lotus, implemented in the Chair of Design Informatics in the Faculty of Architecture, TU Delft, by I. Chatzikonstantinou and Dr. M. S. Bittermann was used. Details of the implementation are discussed in the work of Chatzikonstantinou (2011). The model was left to execute on an Intel Core i5 PC, with 4GB RAM, running the Microsoft Windows 7 Operating System.

DISCUSSION

The optimization process was initially carried out with a population of 300 individuals, for a predetermined amount of 200 generations. It was observed that after about 100 generations, no significant improvement of the solutions can be observed. The optimization run starts with infeasible solutions for the first 15-20 generations. After this point, feasible solutions start to appear. As a result f the process, a Pareto front was generated about 60 generations after starting. Smaller improvements to the solutions were being made after this point, but no radical changes occur. A graph showing the distribution in objective space of the non dominated solutions after 100 generations can be seen in figure 7.

The results that were obtained did correspond to reasonable configurations. The final set of nondominated solutions included mostly variations of a single configuration; i.e., not the whole spectrum of feasible and well-performing solutions was attained. In order to improve the diversity of solutions, a different strategy was employed: Instead of a single run with a large population and many generations, the results of shorter runs of smaller populations, with restarts between them, were combined, i.e. an archive of non-dominated solutions was established from the results of multiple runs. Indeed, in this case, diverse spatial configurations complemented the non-dominated set. Three of the resulting configurations from different runs, can be seen in figure 8.

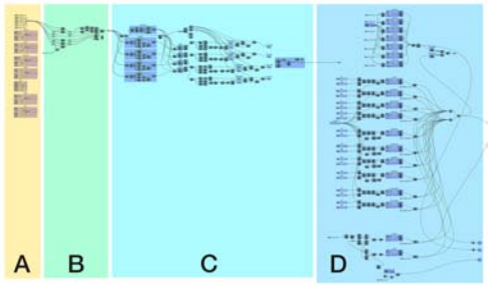


Figure 6
Overview of the implementation of the procedure in Grasshopper. A. Parameters, B. Generation of rectangles, C. DC-RVD procedure, D. Evaluation of performance metrics

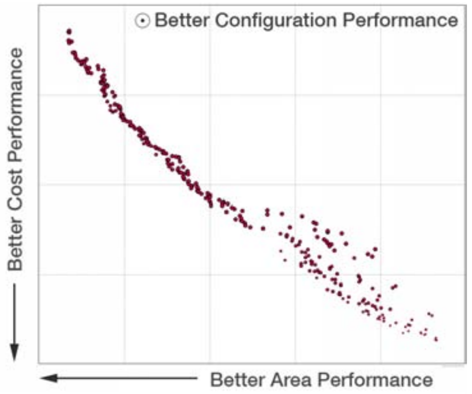


Figure 7
The set of non dominated solutions after 120 generations, using NSGA-II with a population of 300 individuals. Larger dots indicate better performance in terms of spatial relations.

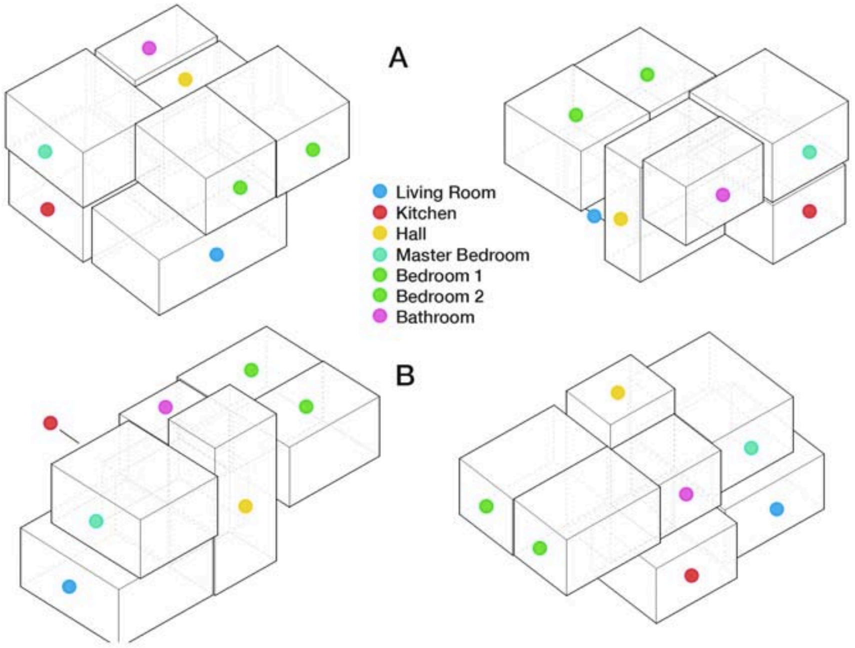


Figure 8
Layout diagram of two of the resulting solutions

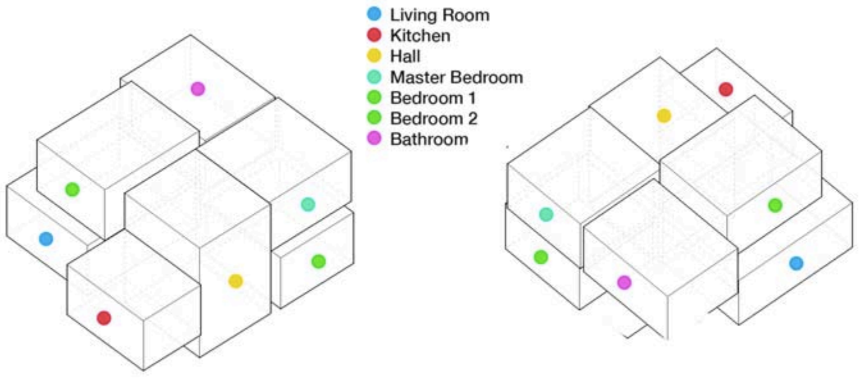


Figure 9
Layout diagram of a solution with four half-floors

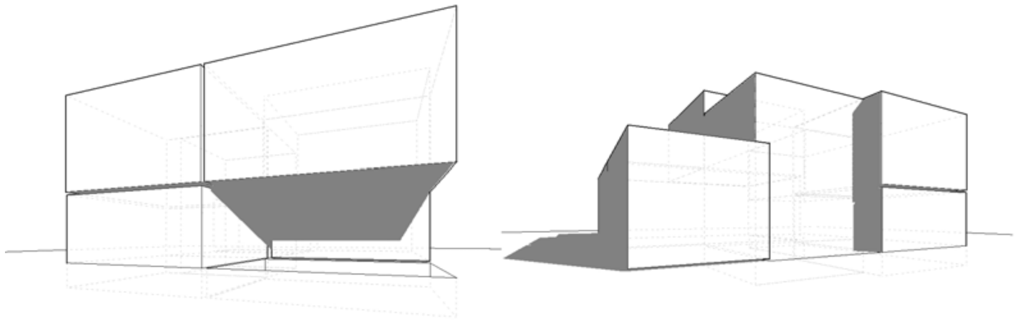


Figure 10
Volumetric study, resulting from the arrangement in figure 9

In order to further evaluate the potential of the procedure, a comparison was made with a simple procedure of placing unmodified spatial units, defined by their positions and size, and including their overlap as an extra constraint, alongside the ones descibed earlier. The optimisation process was confiured with the same parameters, i.e. population of 300 individuals. It was observed that in the latter case, the optimization process was not able to converge to a feasible solution at all, after 80 generations. In contrast to that, the proposed procedure was able to reach the first feasible solution on average at the 24th generation. Not only that, but in the case of the unmodified procedure, the solutions converged to a very specific area of the objective function space. This gives a hint that, even if a feasible solution were to be found, it would not be possible to improve it much further.

We observed that distribution of spatial units in the resulting solutions is generally according to the requirements, and in some cases demonstrates innovative arrangements. However, it should also be noted that in some cases, the results after several generations are unsatisfactory, leading to the conclusion that in those cases, the process is trapped in local optima. An example of an interesting result is shown in figure 9. It represents a design with four half-floors. In this solution, the Master bedroom and bathroom take the upper level, together with one of the bedrooms. The living room and kitchen take the mid-level. The other bedroom takes the lower level. Since in this solution no requirement for proximity between bedrooms was specified, it was considered as a good solution. A volumetric study based on this solution was performed as an indicative next design step, resulting in the model of figure 10.

An optimization run of 100 generations with a population size of 150 individuals last approximately 12 minutes. However, we should note that the process, including generation and performance evaluation, was modelled using components found in the Grasshopper parametric Design program. This type of implementation is rather slow compared to code. As such, with a code-based implementation the execution time would drastically drop.

CONCLUSION

In this paper, a novel procedure for generating 3-dimensional architectural configurations for optimization applications has been proposed, termed Dimension Constrained Rectangular Voronoi Diagram (DC-RVD). The method produces 3-dimensional, non-overlapping configurations of spatial units, based on an array of real-valued variables. It is based off an adaptation of the Voronoi subdivision, namely the Rectangular Voronoi Diagram (RVD). The procedure is using a string of values that represent real-world layout parameters, such as positions and dimensions. This facilitates elaboration of the knowledge generated by optimisation processes.

DC-RVD has been tested in an Evolutionary

Computation-based decision support environment, built in Grasshopper. As a case study, the design of a medium-size detached family house has been addressed. The algorithm produces interesting results, although in many cases the presence of local optima is evident. The 3-dimensional representation allows some novel features to be present, such as vertical circulation cores and spaces with vertical relationships, as well as innovative configurations, intermediate level placement of public spaces.

At it's present state, the tool can be used as a design aid for inspiration, as it allows insight into solutions that otherwise may not have been considered. There are certainly numerous perspectives for improvement in this work. Improvement is possible in the evaluation of configurations, so that soft design aspects related to e.g. perception, such as, e.g., those described in the work of Bittermann (2009) may be included. Finally, there is ample room for more extended trials, with building configuration problems of higher complexity.

REFERENCES

Bittermann, MS 2009, *Intelligent Design Objects (IDO): a cognitive approach for performance-based design*, Ph.D. Thesis, TU Delft

Cagan, J, Shimada, K and Yin, S 2002, 'A survey of computational approaches to three-dimensional layout problems', *Computer-Aided Design*, 34, pp. 597-611

Cha, MY and Gero, J 1998, *Shape pattern representation for design computation*, Working Paper

Chatzikonstantinou, I 2011, *Evolutionary Computation and Parametric Pattern Generation for Airport Terminal Design*, Master's Thesis, TU Delft

Choi, SG and Kyung, CM 1991 'A floorplanning algorithm using rectangular Voronoi diagram and force-directed block shaping', *IEEE International Conference on Computer-Aided Design Digest of Technical Papers*

Deb, K, Pratap, A, Agarwal, S and Meyarivan, T 2002, 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation*, 6 (2), pp. 182-197

Elezkurtaj, T and Franck, G 1999 'Genetic algorithms in support of creative architectural design', *eCAADe 1999 Conference Proceedings*, pp. 645-651

Jo, J and Gero, J 1998, 'Space layout planning using an evolutionary approach', *Artificial Intelligence in Engineering*, 12 (3), p. 149–162

Kamol, K and Krung, S 2005 'Optimizing Architectural Layout Design via Mixed Integer Programming', *Proceedings of the 11th International CAAD Futures Conference*, pp. 175-184

Liggett, RS 1985, 'Optimal spatial arrangement as a quadratic assignment problem', *Design Optimization*, 2, pp. 1-40

Lobos, D and Donath, D 2010, 'The problem of space layout in architecture: A survey and reflections', *Arquitetura Revista*, 6, pp. 136-161

Loemker, TM 2006 'Designing with Machines: solving architectural layout planning problems by the use of a constraint programming language and scheduling algorithms', *Proceeding of The Second International Conference of the Arab Society for Computer Aided Architectural Design*, pp. 88-106

Merrel, P, Schkufza, E and Koltun, V 2010, 'Computer-generated residential building layouts', *ACM Transactions on Graphics*, 29 (6), p. 1

Michalek, J, Choudary, R and Papalambros, P 2002, 'Architectural layout design optimization', *Engineering optimization*, 34 (5), pp. 37-41

Yeh, I 2006, 'Architectural layout optimization using annealed neural network', *Automation in construction*, 15, pp. 531-539