TOWARDS RATING OF GENERATED TYPOLOGIES BY MEANS OF ADJACENCY COMPARISONS

Gabriel Wurzer and Wolfgang E. Lorenz

TU Wien
Vienna, Austria
{gabriel.wurzer|wolfgang.lorenz}@tuwien.ac.at

ABSTRACT

Applying different typologies to the same building yields a number of options concerning extensibility and circulation. However, it is hard to rate and compare these in order to find the most fitting one for a specific building task at hand. In this paper, we wish to work towards this goal using a showcase in which we generate and rate a large number of buildings in order to find out whether certain typologies prevail among the fittest solutions. On a technical level, we contribute (1.) a simple cell-space grammar that generates building volumes, given a preference for different axes in which form can develop - orthogonal, diagonal and vertical, (2.) a rating procedure which infers a fitness for every solution, based on (a.) the assignment of functions to parts of the generated building and computation of adjacencies between these and (b.) the extensibility of the building along its axes. Typologies are attributed in a post-step, as part of a manual analysis. Our results show a preference for the compact/central building type, as dictated by the use of adjacencies for rating a building.

Author Keywords

Typology; Grammars, Adjacency Comparison.

1 INTRODUCTION

Typologies are useful for describing a building from the standpoint of circulation and extensibility. Especially in buildings which serve a large amount of users and need to be adapted often - e.g. hospitals or airports, it makes sense to consider this topic early-on. Thinking about possible options (some given in Figure 1) can lead to a form that allows for short paths between adjacent zones and makes future extensions possible. However, this is a highly intuitive process: One can never say for sure whether a certain typology is adequate or even "optimal" given circulative requirements (expressed as adjacencies between zones, e.g. *near*, *average*, *far*) and possibilities for extension (constrained by the building spot). Furthermore, we never deal with "pure" typologies but always mixtures (refer again to Figure 1, left to right, top to bottom): The "T"-type is contained within the "I"-type, which also contains the "C"-type. "C" is also contained twice in the ring ("O"-type), as is the "L" type. The "Y" type is a "T" with diagonal instead of orthogonal axes.

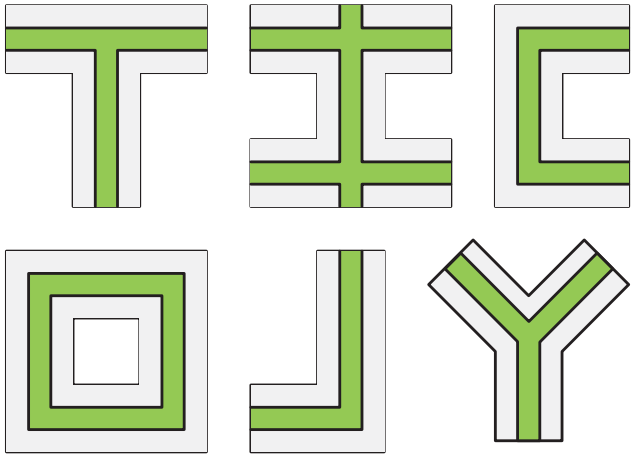


Figure 1. Typologies give options for circulation and extensibility.

To rate and compare such typologies even in the light of such ambiguities is the main goal of our approach. To do that, we proceed in three phases:

1. We first give up the notion of typologies altogether and simply talk of *spatial arrangements* which we generate through a grammar, without loss of generality: All typologies seen in Figure 1 are generated by our program, and we may always name the most fitting one using a post-rationalization step (see Section 3, 'Generating Typologies').
2. We attribute zones to the generated spatial arrangement and compute their adjacency relationships. The calculated values are compared to a preset adjacency matrix and forms the first part of the generated building's *fitness*. The second part comes from the number of possible extensions of the building in relation to perimeter and area left on the building spot (which forms a constraining boundary). Because all of these points require assumptions that are specific for a building type and site, we use the concrete case of a

SimAUD 2016 May 16-18 London, UK

hospital as example (see Section 4, 'Rating by Adjacencies').

3. By iterating generation and rating, we find the most fitting building size and spatial arrangement for the given requirements. The post-hoc attribution of typologies still remains subjective/ambiguous, however, we can now discuss them in the light of an objective measure (see Section 5, 'Results').

2 RELATED WORK

We use a three-dimensional grid grammar [1] to generate spatial arrangements. As always when dealing with grids and update rules, the transition to Cellular Automata (CAs) is fluent, two important differences being: (1.) Our update proceeds sequentially, one placed element after another, while cellular automata update in parallel, and (2.) our elementary element is a 3x3 arrangement of cells, while CAs consider a single cell. Our actual algorithm can be compared to Mitchell and Dillon's work on floor planning [2] (notably also in hospitals!) which agglomerates cells on a grid. By contrast and as novelty, our generation explicitly considers the direction in which this agglomeration evolves (vertical axis, orthogonal axes, diagonal axes). This also works for spatially constrained building spots, meaning that the algorithm takes whatever cell is available if there is none along the preferred direction.

After generation, we subdivide the building vertically into a 'public' part at the bottom (typically used for outpatient departments) and a 'private' part (used for wards). Such a subdivision has also been conducted e.g. by Lopez et al. [3], who also "grow" areas within the building design up to a specific preset size. Doulgerakis [4] furthermore assigns functions ad-post to generated spaces ("embryology"), which might be used for placing sub-departments within the public and private zone in future work.

During our rating step, we compute the *farness* from every cell of an area into every other cell. The reciprocal value of farness is closeness, a measure commonly associated with Hillier and Hanson's integration analysis [5] within their Space Syntax model. Other methods for computing centrality (e.g. eigenvector centrality/random walks) are slowly beginning to catch on the field (see e.g. [6]), however, closeness ("integration") and betweenness ("choice") remain the predominant techniques employed.

Because the (weighted) farness measures between areas are crisp values and not linguistic terms (*near*, *average*, *far*), we need to fuzzify as previously described in [7]. We can then compare the computed adjacencies to a preset adjacency matrix, in order to assign a fitness to each generated solution.

Evaluating fitnesses is a common process in Evolutionary Algorithms (EAs) and optimization. For this study, we have used a brute force approach (parameter sweep) rather than looking into more efficient ways of searching. Admittedly, this is one of the areas we have to look into in future work (a good starting point being e.g. [8]).

3 GENERATING TYPOLOGIES

We use a 3D grid grammar which with only one element, a Megacell consisting of 3x3 cells (refer to Figure 2). We have six cell states, empty, regular, circulation, diagonal connector, orthogonal connector and vertical connector. A Megacell has a circulation in its middle, surrounded by 8 regular cells. This area is the so-called *built area*, which is surrounded by connector cells that extend into directions where a next Megacell could be placed. We have four orthogonal connectors, four diagonal connectors and one vertical connector that is situated above the Megacell's center cell.

A cell has no physical size *per se* - this has to be attributed a posteriori and is only important for the later rating step.

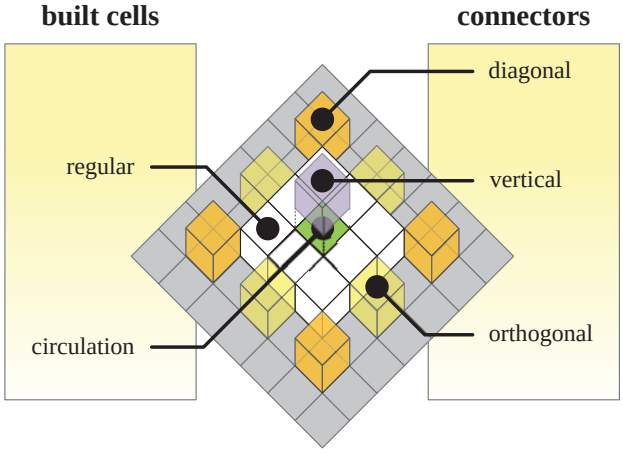


Figure 2. Megacell consisting of 3x3 built cells and connectors.

Before our generation, we set a *building budget* in cells that is not to be exhausted. Every time a Megacell is built, we deduct 9 cells (i.e. the built area) from that budget.

Our generation (see Figure 3) starts with a single Megacell, always situated in the origin of the grid, which we assume is also the center of the building spot. Then, we consecutively place new cells using Step A-C:

Step A chooses a connector *by probability*: We have three probabilities, P_o, P_d and P_v corresponding to preference for orthogonal, diagonal or vertical connectors. All three are independent, one could in principle also choose 100% probability for all three of them. Typically, though, one will want to use these three global parameters to specify the type of building to generate - a high-riser or flat building complex, more orthogonal (90°) or with 45°-diagonals.

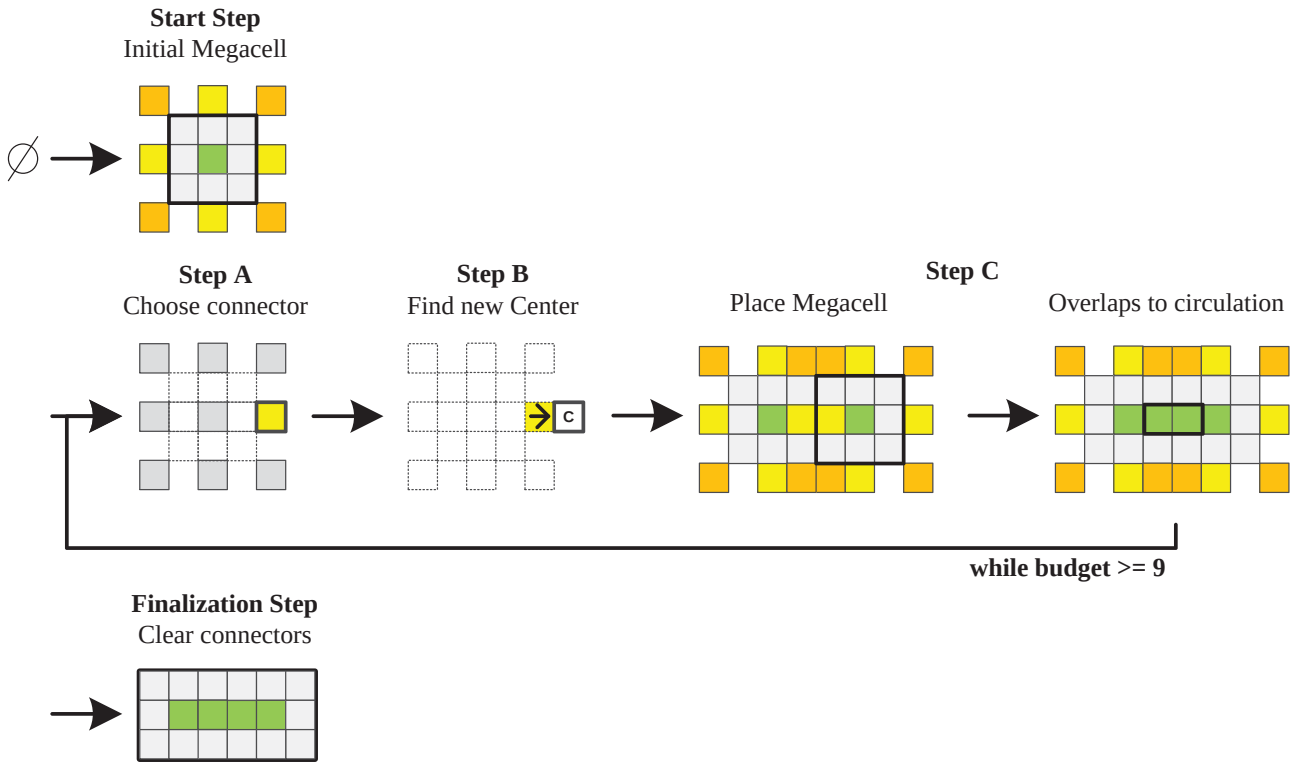


Figure 3. Grid grammar rules.

As a side-note, this step only considers connectors from which it is possible to build; extending beyond the grid, for example, is not possible. Furthermore, all types of horizontal connectors must check whether there is a Megacell underneath (we currently allow a maximum of 1 cells empty space to allow for cantilevers; an exception is the ground floor, on which we can always build).

Step B retrieves the cell C lying adjacent (and in "the direction" of) the connector. In the example shown in Figure 3, this is the cell east of the connector.

Step C Places a new Megacell with center C into the grid. Any overlaps between connector cells and built cells are turned into circulations.

The process is repeated as long as the cell budget is not exhausted (i.e. there are still 9 units left).

As finalization step, we clear all connectors to obtain the final spatial arrangement which can now be post-rationalized, naming the set of typologies that fit. Figure 4 gives examples of the generated spatial arrangement and shows that this is not always easy: While the left building is clearly a high-rise with type "C" base, the right building is a mixture between lots of types such as "Y" and Ring.

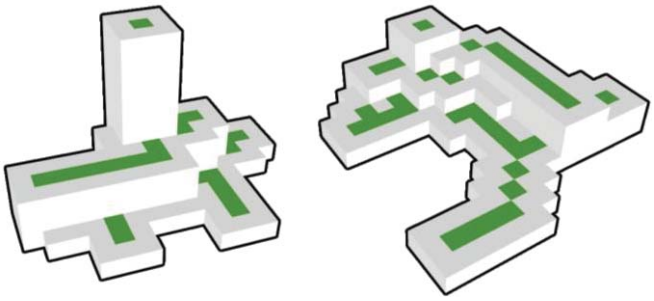


Figure 4. Examples of buildings generated by our approach

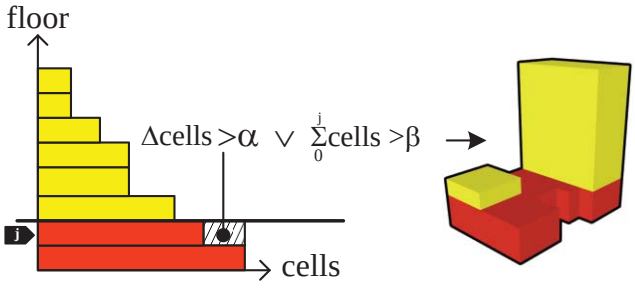


Figure 5. Private/public zone split using a histogram approach

4 RATING BY ADJACENCIES

This step is essentially occupied with transforming the spatial arrangement into a adjacency (half-)matrix and comparing that with preset requirements to compute a fitness. However, until this point the algorithm knows nothing about zones which it needs in order to form the adjacency matrix. We need to attribute such zones to the built, which is of course specific for every type of building. In our case, we chose a hospital with only two types of zones: the public (basement) zone in which outpatient care takes place, and the private zone above in which wards are placed. Our attribution proceeds by forming a histogram containing the number of cells per floor (refer to Figure 5): We go through this histogram from the lowest to the highest floor and test if the difference in the number of cells exceeds a given threshold α or the sum of cells until here exceeds a threshold β. If that is the case, we have found our split.

In each zone's gravity center, we place a node (bubble) and connect that via an edge (adjacency relation; see Figure 6) which we now compute via a circulation graph. In each built cell, we place a graph node which we connect to each built neighbor's node in the same level, assigning a weight of 1 to the edge. Circulation cells additionally check whether there is a vertically adjacent circulation cell to which it then connects using a weight of 2 (penalty for vertical travel). As a side-note, the vertical connection algorithm also enforces a (configurable) minimum separation between vertical connections, as in elevator and stair shafts.

In the circulation graph, we compute the *farness* between each pair of zones, including the zone and itself:

- All nodes of the *source zone* compute the average shortest path lengths to all nodes of the *target zone*, giving an *average farness per node*. However, path lengths are not abstract, so we multiply this with an assumed cell length (8m in our case).
- The *average farness* between source and target zone is computed by averaging the average farnesses of all nodes contained in the source zone.

The computed values are quantities and not qualitative statements (*near*, *average*, *far*). Using three *linguistic membership functions*, we can find the most plausible term with which to describe the obtained average farness (see again Figure 6: near < 100m, average around 100m, far >100m). Admittedly, these values are based on the estimation of an advisor in the hospital we are working on rather than proper data in a large survey of building users. To be accurate, we would need to perform such a survey with many users, and from different perspectives (patients [capable of walking, wheelchair, in bed] and staff members).

We have two constituents for computing the fitness of a solution:

Adjacency fitness. We compare the computed adjacency matrix to a preset adjacency matrix (which in our case was set to *Private-Private: near*, *Public:Public: near*, *Private-Public: average*). For scoring, we represent *near* as -1, *average* as 0 and *far* as 1, and apply the following algorithm to each computed/preset adjacency pair:

```
diff = preset - computed
if computed <= preset then
  fitness = fitness + max(1, abs(diff))
else
  fitness = fitness - max(1, abs(diff))
end
```

Algorithm 1: Comparing computed/preset adjacencies

Extensibility: A building needs to be able to grow over time. We take the building's base floor for checking this. There are two different types of extensibilities, (1.) the ratio between extension points and perimeter and (2.) the area covered by extensions versus the total area left for extension.

Ratio extensions to perimeter. We count the perimeter of the building at ground floor by first counting all cells surrounding the building (depicted red in Figure 7). Then, we go through each connector at the ground level, testing for two different criteria:

- Is the cell at the opposite side of the Megacell built? Example: For a north connector, there must be a Megacell, in order to emphasize the circulation axes of the building when extending.
- When shooting a ray in the connector's direction, there must be no intersections with the building. This is to ensure that extensions are not filling up holes between parts of a building but continue naturally along given axes.

Connectors surviving these two tests are counted, their number is divided by the number of perimeter. This ratio is invariant to scale (e.g. the ratios of the two "C"-types shown in Figure 7 are the same).

Ratio extensions to area left.

For a concrete building site, we have to consider not only potential extensions but also how well they cover the remaining area on the building spot. By using a flood-fill algorithm, we are able to get all cells that are still open for development. Shooting a ray through each available connector in the direction it is pointing to, we count the cell area that would be covered until either a) the site boundary or b) obstacle is hit. We divide the number of covered cells through the number of cells available for development to give this ratio. In short, this determines how well the building's extensions fit into the building site.

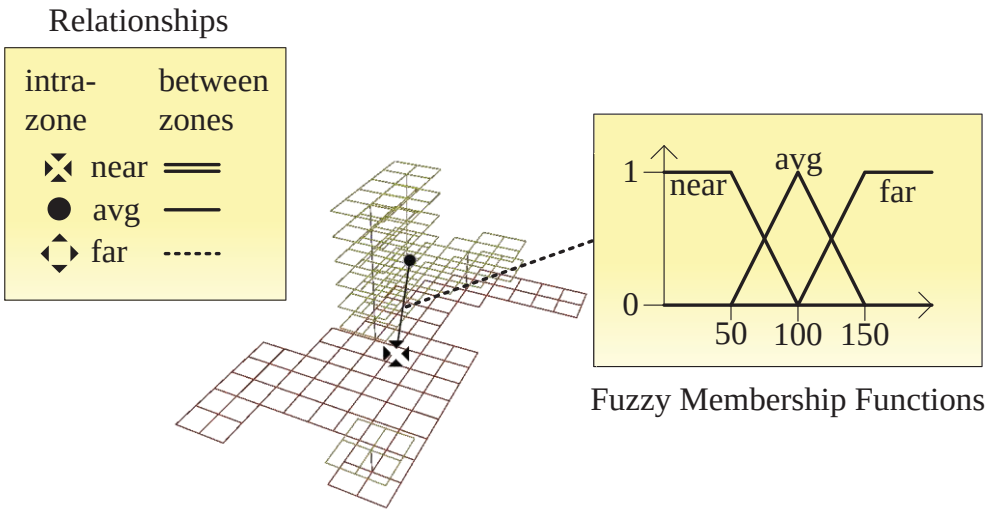


Figure 6. Assigning near/average/far to relationships between zones (visualized through symbols and edge styles).

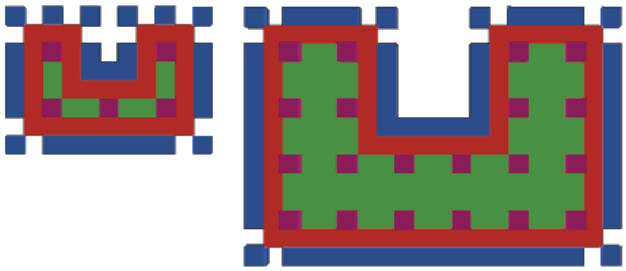


Figure 7. Ratio extensions (blue showing candidate cells) to perimeter (shown in red).

5 RESULTS

We have conducted a parameter sweep experiments in three scenarios, using the values provided in Table 1.

Parameter	Settings
B (building budget)	50, 100 or 150 cells
Po (orthogonal probability)	10%, 50% or 100%
Pd (diagonal probability)	
Pv (vertical probability)	

Table 1. Parameter settings used in the sweep experiments.

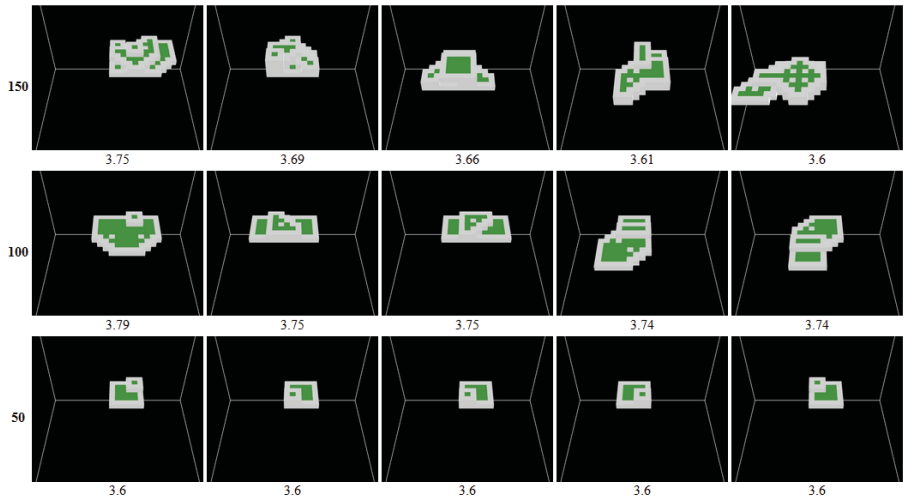
For the first scenario, we used *adjacency* and the *ratio extensions to perimeter* as fitness (n=761 experiments total, 10 repetitions per parameter setting). The results shown in Figure 8a clearly indicate that the adjacency measure keeps the building shape compact (i.e. "O"-type with two levels).

For the 50-cell spatial arrangement, this is most clear (the ideal arrangement is always the same "O"-pattern, even though rotated). As we go into 100- and 150-cell arrangements, there are additional diagonal arrangements protruding from the central building. However, these are not formulated out so as to fit into any other typological category than the central type.

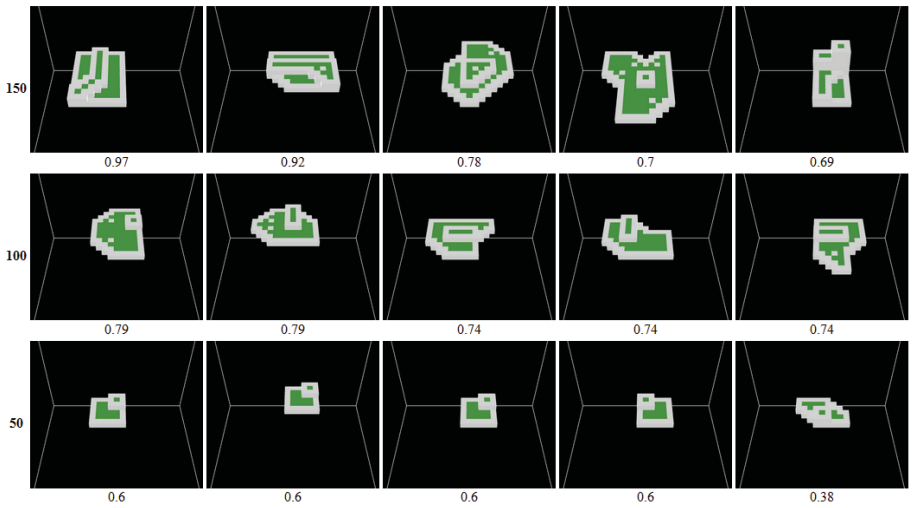
Another striking result of the adjacency measure is that the building is always two levels high. Anything else would be penalized the weight that was added to the vertical circulation, in order to simulate lifts and stair movement. The choice of adjacency matrix as well as the rather strict fuzzy membership functions further contribute to the preference for compact, flat buildings.

In scenario 2 (n=243 runs, 3 repetition per parameter setting), we have thus chosen to not take the adjacency into account but only rely on the ratio of extensions to perimeter. In that case (Figure 8b), the grammar is allowed to build higher structures (see rightmost building in the 150-cell category), however, it still seems to avoid these in favor of a large building base which translates into many extensions (and thus higher ratio).

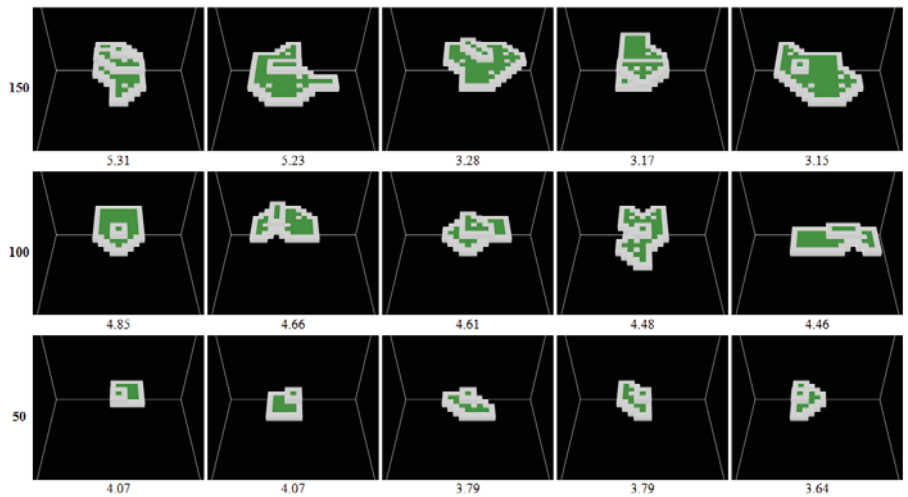
Taking adjacency and both ratios into account leads to scenario 3 (Figure 8c; n=243 runs, 3 repetition per parameter setting): While similar to scenario 1, we see more diversity even in low cell budgets. Furthermore, the ratio extensions to area left seems to prefer diagonals (more directional freedom to claim the remaining area of the building site), it seems.



(a) Scenario 1: Fitness calculated by adjacency and ratio of extensions to perimeter



(b) Scenario 2: Fitness calculated only by ratio of extensions to perimeter



(c) Scenario 3: Fitness calculated by adjacency, ratio of extensions to perimeter and ratio of covered vs. remaining cells

Figure 8. Top 5 spatial arrangements for cell budgets 150, 100 and 50 (top-down), in descending order of fitness (left-right).

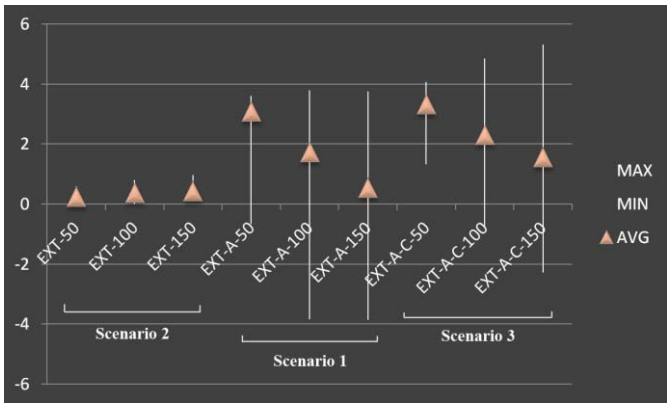


Figure 9. Fitness distributions for the 3 experiment scenarios EXT (extensibility ratio), n=243 runs, fitness in [0, 0.97], EXT-A (ext.ratio and adjacency), n=761, fitness in [-3.87, 3,79], EXT-A-C (ext.ratio, adj. and coveragte), n=243, in [-2.28, 5.31].

Under these circumstances, it is hard to infer different "typologies" that are ideal - adjacency really dictates that the building should be a sphere, and the ratios furthermore work towards a ground floor of maximum extensibility. In order to allow for results as given in Figure 4, one would need to relax the required adjacencies and shift the fuzzy membership functions to allow for greater distances.

Comparing all three scenarios (see Figure 9), we can see that Scenario 2 has a relatively minor, positive range. Adding adjacency (Scenario 1) makes the fitness range oscilate between +/- 4. The second ratio is an additive term, hence Scenario 3 is shifted upwards.

CONCLUSION AND OUTLOOK

We have presented an approach that is capable of generating spatial arrangements resembling typologies, based on three-dimensional grid grammars. Rating each generated solution by adjacency and extensibility, we wanted to see whether the same typologies appear in all of the highly-rated buildings. Our current results show a high preference for compact/central building types, as dictated by the use of adjacencies during the rating process. However, results could differ in cases where the building spot is highly constrained by pre-existing buildings and/or regulations. Since this is nearly always the case during real projects, our next step is to apply these concepts in practice.

Another future extension, which is currently being developed, is concerned with using building physics calculations as additional fitness criterium.

REFERENCES

1. Knight, T. W. Shape grammars: six types. Environment and *Planning B: Planning and Design 26*, 1 (1999), 15–31.
2. Mitchell, W.J. and Dillon, R. A Polyomino Assembly Procedure for Architectural Floor Planning, *Proceedings of the Third Environmental Design Research Association Conference*, (1972), 23-5-1–23-5-12.
3. Lopes, R., Tutenel, T., Smelik, R.M., de Kraker, K.J. and Bidarra, R. A. Constrained Growth Method for Procedural Floor Plan Generation. *Proceedings of GAME-ON 2010*, (2010), LTSDB10a:1-8.
4. Doulgerakis, A. Genetic Programming + Unfolding Embryology in Automated Layout Planning, Master Thesis, University College London (2007).
5. Hillier, B. and Hanson, J. *The Social Logic of Space*. Cambridge University Press, (1982).
6. Fidler, D. and Hanna, S. Introducing random walk measures to space syntax. *Proceedings of the 10th International Space Syntax Symposium*, (2010), 145:1-145:9.
7. Wurzer, G. and Lorenz, W.. From Quantities to Qualities in Early-Stage Hospital Simulation. *Proceedings of I-WISH 2013*, (2013), 22-27.
8. König, R. and Knecht, K. Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 28*, (2014), 285–299..