Visual Representation of Adjacencies

A NetLogo application to turn functional matrices into bubble diagrams

Wolfgang Lorenz[1], Gabriel Wurzer[2]
[1,2] TU Wien
[1,2] {wolfgang.lorenz|gabriel.wurzer}@tuwien.ac.at

This paper is based on the assumption that a key challenge of good design is spatial organization as a result of functional requirements. The authors present a new NetLogo application that assists designers in understanding proposed functional relationships (of spaces) by visualizing them graphically. In more detail, the tool translates adjacency matrices (as representation of functional relationships) into a graph (in architecture known as bubble diagram). The latter can be considered as a pre-step of the actual design, by which the information about functional interrelations becomes more readable and understandable for the designer.

Keywords: *Adjacency, Bubble Diagram, NetLogo, Visual Representation*

MOTIVATION

Design thinking means understanding the design problem. In that context, the designer has to deal with project issues (site, climate, energy, behavioral aspects, costs, regulations and so forth) and their interrelationships. Certain tools are available to analyze and solve those project issues, and every individual tool tries to determine the form in its own way (think e.g. site-specific constraints vs. costs). This work uses space adjacency graphs (White 1986) for capturing spatial interrelationships and aspects of a client's daily operation. It is one out of many possible methods for requirement engineering in architecture, however, we argue that there are many advantages that make space adjacencies especially suited for design thinking:

- Designers are used to think visually; space adjacency graphs depict functional relationships which are often given as spreadsheets visually, making them more accessible during the preliminary design phase.
- Another issue of matrices (i.e. spreadsheets) is that they become unreadable when lots of functional relationships are given; in contrast to that, space adjacency graphs allow for spatial organization of this data (e.g. clustering), thereby reducing its complexity by means of visualization.
- Space adjacency graphs can be used for *more than just functional relationships*: One may e.g. capture *interaction* between functional units as a different form of "adjacency" which comes out of the given operational routines.

To sum up, adjacencies are important for a multitude of planning tasks and stand at the beginning of the planning process. However, they are typically given as matrices or spreadsheets which are hard to understand especially for visual thinkers such as design-

ers. The question of how to transform such adjacency matrices into a graphical representation that can be analyzed and edited during design is thus the key goal of this paper.

BACKGROUND

In general, adjacencies describe relationships between functional units or between organization units that already incorporate different functions. Possible adjacencies are 'cooperation' (see Neufert 2000), 'relationship' (see White 1986) and 'location' (see Schönfeld 1992):

- Cooperation: an interaction matrix defines the degree of cooperation ("undefined", "often", "average", "seldom", "n/a") between functional units with respect to the overall operation of a building (Wurzer 2015). For an example see the cooperation schema in Neufert (2000), where a half-matrix shows the degree of cooperation in specialized medical departments ("intensive", "frequent", "occasional" and "seldom").
- Relationship: there are several reasons for bringing spaces together or for separating them. Positive adjacencies may result from desired short paths for circulation of people and/or material or of security reasons ("critical", "important" or "desirable"); whereas, negative adjacencies may origin from noise, dust or fume. It is the adjacency matrix, the bubble diagram and the zoning diagram that helps the designer to understand the clients operation and to anticipate appropriate design solutions (White 1986).
- Location: Schönfeld (1992) differentiates between function-specific and function-neutral when dealing with activity assignment. As an example serves a dwelling with separated location of rooms depending on individual versus community area. While a function-neutral activity assignment places the kitchen and sanitary facilities into the center; in a function-specific activity assignment the kitchen is rather assigned to the living area and areas of personal care to the bedrooms. Different quantities of such connectivity between rooms are described (in a half matrix) as "spatially connected", "adjacent" and "neutral" (the connectivity can also be named as "near", "neutral" and "far").

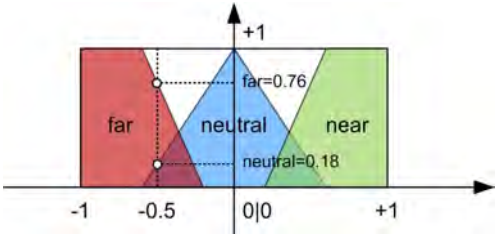


Figure 1
Membership function for the linguistic terms "far", "neutral" and "near".

So far, adjacencies are classified as fuzzy terms, such as "critical", "important", "near", "far", "close" or "neutral". In order to determine the degree of membership for linguistic terms, adjacencies have to be mapped to a membership function:

$$\mu_{Ai} : X \rightarrow [-1, 1] \quad (1)$$

where A is the set of linguistic terms and X is the corresponding number, giving the degree of membership for a linguistic term. Figure 1 shows a possible translation of a relationship termed between not too "far" but also far not "neutral" (and under no circumstances "near"). With $X = -0.5$ the term "far" corresponds to 0.76 - which is below the maximum 1 - while "neutral" corresponds to 0.18 (and "near" even equals zero); this means that X fulfils the initial conditions from before (not too "far" but also far not "neutral"). From this it can be deduced that the degree of membership for linguistic terms can, in general, be defined as a function. Moreover, it means that all kinds of adjacency concepts can be mathematically mapped in any range (e.g. in the range of [-1,1]).

Diagramming architectural ideas

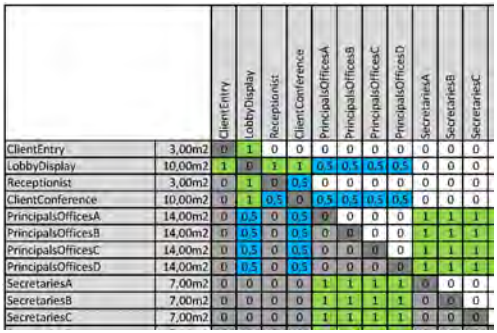
Adjacencies are one way to diagram architectural ideas, described in detail by White (1986). After several principle decisions are made and interrelation-

Figure 2
The adjacency matrix graphically shows relationships between functional units (or between rooms).

ships are defined, the designer inserts all characteristics into a (half) matrix. This matrix provides the information needed to construct the bubble diagram, which is a visual representation of the predefined interrelations. Finally, zoning diagrams superimpose the bubble diagrams; this improves readability and gives a hierarchy for the next step of the design process.

Another field of diagramming architectural ideas focuses on external site conditions; where site is defined as an ongoing set of active networks that are intertwined in complex relationships (White 2004). Such activity networks derive from user requirements which asks for different kind of e.g. natural lightning, views to and from the site, approach direction, direct or indirect access, scale, connectivity to exterior environment, privacy and so forth; whereby the type of information (forces) influences the type of representation: arrows symbolize possible connection to the site, cones provide an opportunity for viewing, waves for noise, hatches for shadowing and so forth. The basic idea is that activities define forces that locate the building on the site. Consequently, contextual analysis represents another approach to enter the design problem.

In the winter term 2018 the authors organized a design studio dealing with a (specialized private) clinic. One challenge concerned the preorganization of the complex design task. The design process started with a story board visualizing the organization and processes of patients and medical staff (as paths), followed by site analysis including information about existing buildings, trees, flow of person on site, links with the public transport system, views to and from the site and so forth (strongly influenced by White 1986, 2004). The findings were transformed into matrices, bubble diagrams and finally into a schema which served as basis for the actual design. Such a structured access to a complex design task improved both, quality and speed of the (design) problem identification. Students quickly gained an impression of how to deal with complexity by matrices, bubble diagrams and zoning diagrams.



Adjacencies

As previously described, relationships between functional units are specified either in fuzzy terms ("close", "distant" or "neutral") or in mathematical terms (relative weights in the range of [-1, +1]). A common and useful form of representing functional requirements and their relationships is a (half) matrix (see figure 2). Numbers and/or colours define the kind and degree of each relation (a positive green number representing attraction, a negative red number repulsion and white or zero a neutral relation); however, adjacency matrices provide a purely abstract form of adjacency representation, see e.g. in Neufert (2000). As a consequence, it is difficult to keep the overview if the number of functional units increases. Because of this difficulty, in current planning practice, planners often look at each adjacency relation in isolation. Apart from clarity, changing values as a result of fine-tuning and visual optimization are nearly impossible. Adjacencies can also be incorporated as a part of automated floor-plan generation for the final design (Elezkurtaj 2002; Lobos and Donath 2010). This approach allows taking other parameters into account as well; e.g. optimal shading of the building, orientation in relation to the sun, views from site and to site, noise, size and proportion (compare White 2004); however, there is no "*unique optimal solution for over-constrained problems*" (Balachandran and Gero 1987). This means that, in order to select the final solution an expert is nevertheless essential (only the architect is able to consider all variables that

have driven the design generation).

To improve readability and to avoid the problem of over-constraining at the same time, the authors propose a purely abstract graphically representation of adjacencies, as a pre-step of defining form, shape and space (including spatial allocation). The model allows architects not only to preview the preliminary functional layout but also to interact with the system in the early stage of design. It is based on the so called bubble diagram, a representation of adjacencies in nodes (corresponding to functional units) and edges (corresponding to their relationships), see figure 3. In short, the basic idea is to automatically generate a visual representation of the adjacency matrix reflecting attraction and repulsion while avoiding overlapping links (at least minimizing overlapping links). In addition, the model allows to group functional units (known as zoning). The workflow and the implementation by hand are already given in White (1986); while the technical background refers to Arvin and House (1999), who describe functional units as point masses and relationships between them as spring exerts forces.

METHOD

In an nxn adjacency matrix all functional units are connected to each other. This allows specifying every single connection in the range of [-1, +1]. In practice, adjacencies are typically recorded in the form of spreadsheets (figure 2). It becomes obvious that the number of functional units (and links between them) influences the readability and the overall comprehension of interrelations. Thus, requirements of the model implemented in NetLogo (a multi-agent programmable modeling environment) are:

- The possibility to use the data directly from spreadsheets
- The transition of this data into a graphical and comprehensible model (bubble diagram)

Focus of the proposed model is not to define the spatial distribution of functional units, but to create a more abstract bubble diagram with

- Minimal overlapping links
- Sizes of nodes according to their approximate size (defined by their diameter)
- Relative position of nodes according to the adjacency matrix (colour and thickness of links corresponding to the strength of relationship)

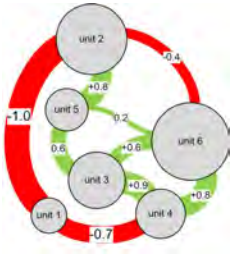


Figure 3
The Bubble diagram is a representation of adjacencies in nodes (diameter corresponding to its approximate size) and edges (visualizing their relations).

In a bubble diagram, bubbles represent functional units which are linked with each other according to their predefined interrelation (see figure 3). Colour and/or thickness of the link indicate the kind (attraction or repulsion) and the degree of relation ("high", "middle" or "low" attraction/repulsion). Such a representation facilitates further interpretation, such as zoning, by which functional units are grouped together. In order to visualize adjacencies in a bubble diagram two steps are necessary:

- Functional units (organization units) and their relations have to be visualized as a graph (bubble diagram)
- Positions of all functional units (organization units) in the bubble diagram have to be changed according to their adjacencies

NetLogo as programming environment

The authors regard a bubble diagram as a graph where nodes are functional units (bubbles) and edges are weighted interrelations between them [-1,1]. In the proposed model, nodes permanently move according to the forces acting on them (as a result of their interrelations). Nodes can subsequently be treated as agents that contain informa-

tion about themselves and about their neighborhood (such as interrelations to other nodes, direction in which to go, size and position). This information is, finally, what influences the movement of each node. NetLogo (Wilensky, U. 1999), a multi-agent programmable modeling environment, supports such functionality.

The author's program itself is divided into two steps, the setup and the actual running.

Setup the model

In the first phase, all nodes (functional units) are created and filled with specific information. The information of each node derives from a spreadsheet exported as csv-file. The spreadsheet is a matrix that lists all functional units in the first column and in the first row respectively (see figure 2). The second row of the spreadsheet defines the proposed size of the unit (e.g. minimal or average required space). Since every functional unit in the first column corresponds to a functional unit in the first row, a diagonal of self reflection separates two triangular fields. Subsequently, two possibilities occur: either the lower triangle reflects the upper one or both halves may contain different numbers. The latter means that the interrelations between A to B may differ from B to A, which might seem unusual (but is theoretically possible).

While importing the semicolon separated csv-file into NetLogo, each line/row (except the first one, which just contains the names of the functional units) corresponds to a specific functional unit. While the first two entries of each row define the name and the size of the unit, subsequent entries in the row correspond to the interrelations with other units (and with oneself which is disregarded). An entry equal to zero represents an exclusive neutral relation, a positive value (between zero exclusive and one inclusive) reflects an attractive relation and a negative number (between zero exclusive and minus one inclusive) stands for a repulsive relation. Nuances are given as fractional amount. Attractive and repulsive relations are stored as separate properties (lists) for each functional unit (giving the kind of interrelation as number, while the position in the list corresponds to the ID of the functional unit). In the attraction list negative numbers are set to zero and, vica versa, positive numbers are set to zero in the repulsive list.

In the next step all functional units are linked with each other according to the two affected building lists. The edges color derives from the type of interrelation list (green for attraction, red for repulsion) and the edges thickness equals the number in the corresponding list (edges with neither attraction nor repulsion are set to zero and hidden).

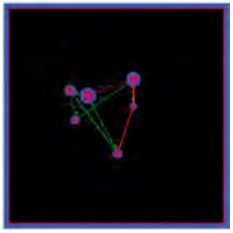


Figure 4
Random distribution of all units, including size of units and their interrelations according to the adjacency matrix.

Figure 4 shows one possible distribution of functional units inside the NetLogo world, defined by a discrete number of patches in horizontal and vertical direction. Functional units are only allowed to remain inside the so-called "playing field", which is bounded by patches called "wall" (magenta) and "outside" (blue). The setup phase results in a principle visual image of all interrelations where links may intersect. Since functional units are arranged randomly they may overlap as well. If a unit overlaps another one, new positions are examined until either a non-overlapping position is found or a maximal number of tries is reached. In the latter case, the result indeed overlaps after the setup phase.

Adjacencies and Newton's law of universal gravitation

Functional units can be interpreted as point masses of a certain size, linked with each other according to the input (adjacencies). Since the whole model deals with masses (sizes of functional units), distances and forces (adjacencies), the authors decided to base the

model on Newton's law of universal gravitation:

$$F = G \cdot \left(\frac{m1 \cdot m2}{r^2} \right) \quad (2)$$

where, in our case, F is the force acting between two functional units, m_1 and $m2$ are the sizes of the units (interpreted as masses), r is the distance between the centers of the two bubbles, and G is a constant.

The (movement) vector of every single functional unit derives from the interrelation with each other, where the direction and intensity (F) to every other functional unit defines the resulting force (movement vector as sum of x- and y-components). One important feature of Newton's law of universal gravitation is its considerations of distances (compare formula 1), where closer units have a greater influence on the final path (resulting force) than distant ones. That means the model focuses more on the immediate neighborhood of units. Walls can also be incorporated, treating them as masses that attract functional units; the influence can be adjusted and should be lower in respect to functional units.

The model is based on a previous work investigating the Newtonian gravitation model to propagate changes of a single relationship immediately to the whole space layout (Lorenz et al. 2015). The here proposed model is not only a further development in the settings and the running, but also an adjustment to the graphical layout of a bubble diagram used by architects. Special attention is also paid on the interaction of the user and the (graphical) output in order to verify the quality of the result.

One complete run

Each run consists of certain steps:

- Find and solve overlaps with others units, see figure 5: at first, all overlaps for each functional unit are stored in a specific unit-related list. Following this phase, each list is gone through step by step. If another functional unit (*room 1*) overlaps oneself (*me*) the other unit is pushed away from ones centre (dashed line in figure 5); except *room 1* is marked as *fixed* or if it is overlapped by another unit (*room 2*). In the latter case, the list of *room 1* (with its own overlapping rooms) is investigated first. In case *room 1* would hit a wall, its direction is changed by random until the unit is able to move forward. The phase "find and solve overlaps" is repeated until all lists are empty. Since all steps of this phase change the positions of functional units the interrelations between all units are, finally, recalculated (vectors as resulting forces).
- Find other functional units in ones way (and if necessary move them out of the way), see figure 6: at first, each unit stores its obstacles (within a certain distance) in a unit-related list. Following this phase, each list is gone through step by step. If the list is empty, there is no other functional unit in the way and the unit can move forward (direction and magnitude given by the vector of resulting forces). In all other cases, the obstacle (the *other-room*) is pushed in perpendicular direction out of one's way (path of *me*).

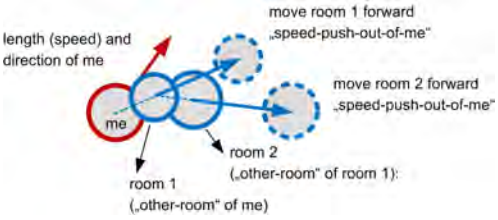


Figure 5
Search for a unit that can be moved in opposite direction and move it.

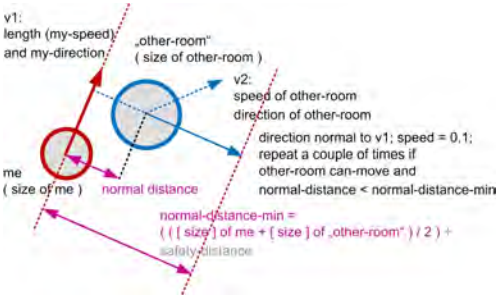
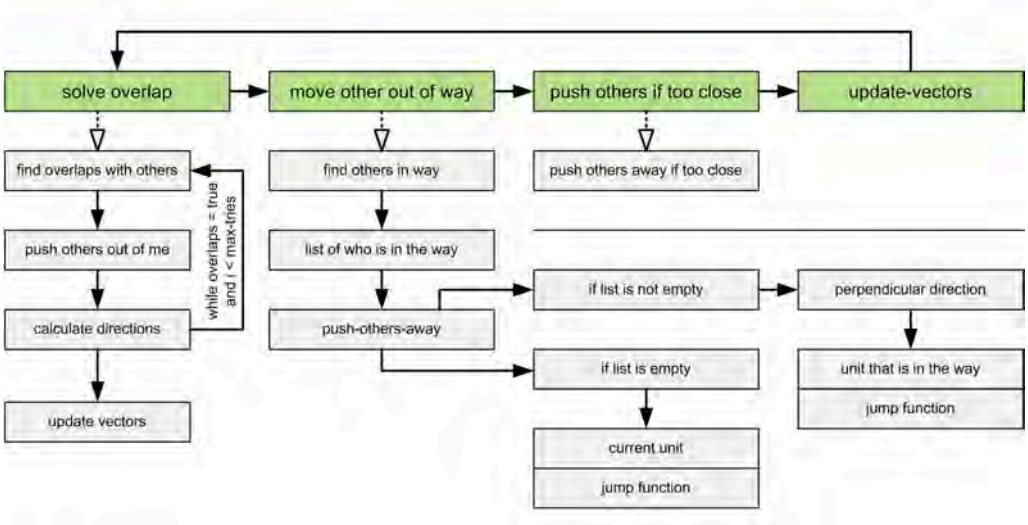


Figure 6
Units that are in ones way are perpendicular pushed aside.

Figure 7
Schema of a complete run.



- Push others away if they are too close: a predefined safety distance guaranties that units don't come too close to each other (which improves readability). This time other functional units within a certain radius are pushed outward by a predefined distance (the direction is defined by the two centre points).
- Finally, direction and intensity - in short, the movement vector of each functional unit - are recalculated according to adjacencies and Newton's law of universal gravitation.

After one pass, the cycle restarts with the first step, see figure 7.

Settings

There are a couple of possible adjustments that influence the result or at least the speed of coming up with a (intermediate) solution. Above all, the size of the world is defined manually in line with the size of the graph (quantity of nodes), so that all functional units can be placed and moved according to their adjacencies. Other functionalities can be set and controlled by parameter sliders (see figure 8):

- At first the user can adjust the thickness of the border (at least one patch) as well as the theoretical mass of a single "wall" patch.
- Two attraction sliders define the "range of balance" concerning units that attract each other. The lower bound defines the area of repulsion (to guarantee a certain distance between attracting functional units), while the upper bound defines the area of attraction. Within the "range of balance", no gravity is taken into account (the two units are, so to say, in balance).
- Two repulsion sliders define the "range of balance" concerning units that reject each other. This time the lower bound defines the area of repulsion, while the upper bound defines the area of attraction (in order to prevent drifting apart).
- Independently from the lower bound of attraction, the user defines a minimal distance between functional units as well. This guarantees that two units are not sticking together (which improves readability).

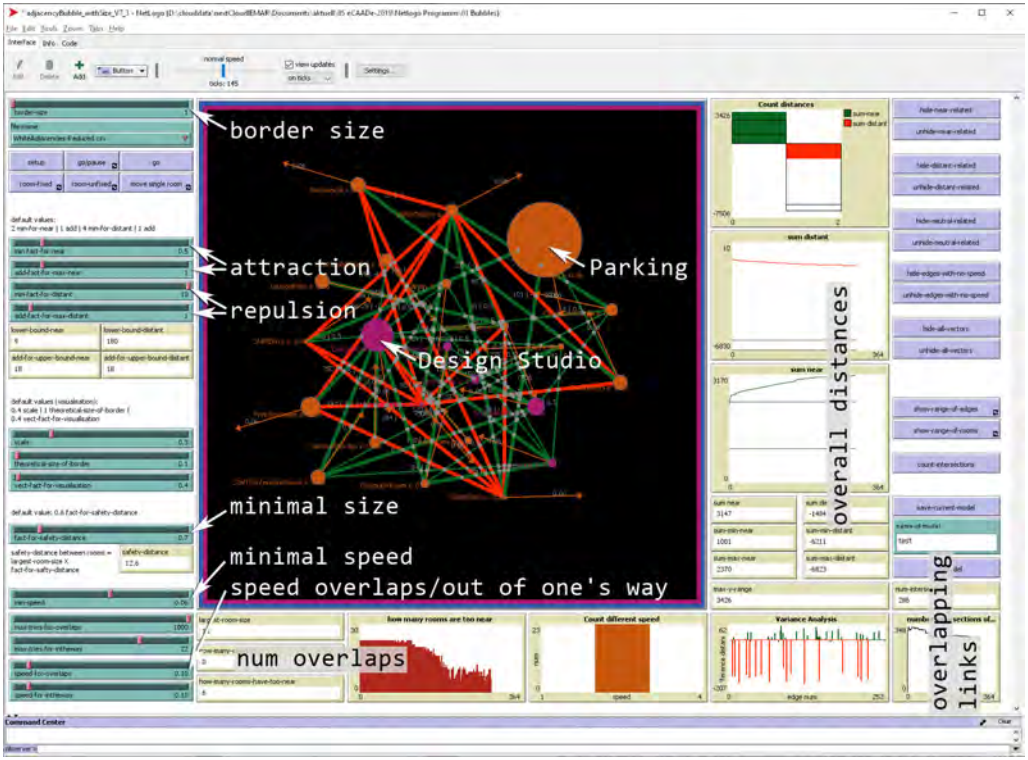


Figure 8
Result of a setting for a design studio given in White (1986).

- The user can also adjust the general minimal speed which accelerates slow moving units.
- The phase of solving overlaps and of pushing other units out of one's way ignore the calculated speed of the movement vector but provide their own speeds instead. Both of them can individually be adjusted by the user.

In general, sliders that define distances are given as a factor that is multiplied by the largest size of units (which is equivalent to the largest mass).

Monitoring

While flowing through the program, the data is continuously monitored; e.g. single and overall distances are checked whether they are too short or too large in respect to the initial settings (range of attraction and repulsion). Furthermore, the number of rooms that are too close to each other, the number of overlapping units and the number of intersecting links are permanently displayed and provide an indication about the fitness of the (intermediate) result (see figure 8).

Interaction with the model

Since the input configuration may not provide a proper solution, the user/designer can not only manually change single relationships (value and/or connection) but he/she can also move single functional units - in doing so, links and movement vectors are permanently updated. In the end, the user is given a

graphical output of all functional units and their relationships. To improve readability it is possible to separate "near" from "distant" relations and vica versa, see figure 9. Manually changed relationships are exported as csv-file and written to a new spreadsheet. This ensures coherence between input and output. In order to go back on existing outcomes, each step can be saved and reloaded into NetLogo as well. Finally, the result (bubble diagram) serves as a graphical basis for the next steps in the design process (the actual design).

CASE STUDIES

Case study design studio

The authors verified the general usability of matrices, bubble diagrams and zoning diagrams in a design studio in 2018. The results were promising, since the later specified forms and spatial allocations were always in line with the required adjacencies. Students benefit from this stepwise approach and could handle even complex tasks such as a (specialized private) clinic.

Several of the student's contributions served as a test case for the proposed model; even those with very detailed and complex adjacency matrices which entail many intersecting links. In the case of a confrontation clinic, considering single representations of the result led to a better understanding of the underlying structure. Especially zoning and the separate viewing on repulsion and attraction enfold the hidden structure (compare figure 9).

Case study office

In 2016 the authors analyzed an adjacency graph of a design studio given in White (1986). Graph measures included closeness, betweenness, Eigenvector, degree, density, clustering coefficient and critique analysis (see Wurzer, Lorenz 2016). The basic intention was to uncover as much hidden structures as possible to improve adjacency requirements before they are used during the ongoing design process.

As another test case, the authors used the very same example by White (1986) in order to prove functionality of the proposed model. One possible intermediate result is given in Figure 8. The graphical output is in principle similar to White (1986) with "parking" located at the edge and the design studio located in the centre. All functional units that should be close are locally concentrated, while separate units are indeed segregated. From the graphical output it becomes evident that even though the graphical output is much more readable than the spreadsheet input (compare figure 2) further adjustments are necessary. Therefore, apart from the separate focus on attraction and separation it is also possible to hide/unhide all edges and all vectors. This allows the user to solemnly focus on the distribution of all functional units. Another point concerns complexity. In order to reduce complexity the authors consider it necessary to incorporate hierarchy. This would allow combining functional units and adjacen-

Figure 9
One possible intermediate result; colored according to zones (left), showing only near relations (middle) and showing only distant relations (right).

cies in a higher ranked node (so to say, nodes consist of smaller components with their own adjacency); however, this is not included yet.

DISCUSSION

Functional requirements and their relationships are an integral part of design; however, when dealing with adjacencies two major problems occur:

- If an nxn adjacency matrix is used as representation, one might quickly loose overview when the amount of data increases.
- If adjacencies are part of an all-in-one solution (which incorporates all possible information e.g. in the early stage of design) general information (site analysis, adjacencies) are dealt with at the same time as form. This might constrain creativity.

As consequence, it seems more practicable to prepare the information about functional units and their relationships in a way architects can easily capture; this is to say in a graphical way (as bubble diagrams). The proposed model supports this approach. It could be shown that

- the graphical output reflects the principle layout done by hand (compare White 1986)
- the model is suitable for educational purposes since it improves the understanding of interaction between functional units

The current model does not directly avoid intersections of edges; however, it monitors the number of intersections. This means, that in the end of the day, the best solution can be selected out of many runs of the same configuration. For the moment, this seems to be a viable way.

OUTLOOK

The here presented model solely focuses exclusively on the early stage planning. Such an approach lacks feedback whether the following phases of the design workflow still fulfill the predefined requirements (adjacencies); however, it is conceivable that the (manually changed) adjacency matrix, in the end, serves as input to prove the final design. The feedback loop between the adjacency model and the design process is therefore one aspect of future work.

The authors also point out that it is, in principle, possible to handle hierarchical adjacencies. That means units may contain sub-units with their own adjacencies. This would improve readability, since sub-units are combined to nodes of higher hierarchy. This feature is not yet implemented, but an important task for future work.

REFERENCES

Balachandran, M and Gero, JS 1987, 'Dimensioning of architectural floor plans under conflicting objectives', *Environment and Planning B*, 14, pp. 29-37

Elezkurtaj, T and Franck, G 2002, 'Algorithmic Support of Creative Architectural Design', *Umbau*, 19, pp. 1-16

Lobos, D and Donath, D 2010, 'The problem of space layout in architecture: A survey and reflections', *arquiteturarevista*, 6(2), pp. 136-161

Lorenz, WE, Bicher, M and Wurzer, G 2015 'Adjacency in hospital planning - Using Newton´s differential equation', *Proceedings of Mathmod 2015: Abstract Volume - ARGESIM Report No. 44*, Vienna, pp. 1-6

Neufert, P (eds) 2000, *Bauentwurfslehre, 36th edition*, Friedr. Vieweg & Sohn, Braunschweig/Wiesbaden

Schönfeld, JW 1992, *Gebäudelehre*, Kohlhammer, Stuttgart; Berlin; Köln

White, ET 1986, *Space adjacency analysis: Diagramming information for Architectural Design*, Architectural Media, Tallahassee, USA

White, ET 2004, *Site Analysis: Diagramming information for Architectural Design*, Architectural Media, Tallahassee, USA

Wilensky, U 1999, *NetLogo. http:// ccl.northwestern.edu/netlogo/*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wurzer, G 2015, *Habilitation: Agent-based simulation for early-stage planning of complex buildings*, Vienna University of Technology, Vienna

Wurzer, G and Lorenz, WE 2016 'SpaceBook: a case study of social network analysis in adjacency graphs', *Proceedings of eCAADe 2016*, Oulu, pp. Vol.2; 229-238