Spatial configuration: Semi-automatic methods for layout generation in practice

Lucy Helme, Christian Derix and Åsmund Izaki
Woods Bagot | Global Studio

Pages: 35-49
Spatial configuration: Semi-automatic methods for layout generation in practice

Lucy Helme, Christian Derix and Åsmund Izaki
Woods Bagot | Global Studio

Despite five decades of research into layout automation that have produced a wide variety of methods for automatically solving layout problems, no automated layout programme is used as standard within the architectural profession. Therefore scope still exists to better formulate these methods in terms of their relevance to architects. In a traditional workflow, architects design by generating ideas and testing them in an iterative process through which a solution gradually emerges. The knowledge he or she has from years of training and experience is fed into the project through this design-evaluation loop. By automating the entire process, many explicit criteria can be successfully built into the model and optimised. However, these tools sit outside of the architect’s normal workflow, and the integration of his or her non-quantifiable assumptions during the design process is lost.

The Computational Design + Research group (CDR) of AedasR&D has developed semi-automated methods in which the designer is central to the process, known in other fields as ‘human-in-the-loop’. We see these methods as hybrids, where the design drivers that can be encoded as rules are built into the resulting tools, and the remaining criteria can be evaluated and optimised by the designer through interaction with the tool in a participatory process. Live evaluation and feedback are therefore fundamental to the tools’ success; the consequences and implications of both the encoded design drivers and of decisions taken while using the tool must be transparent. The problem of developing these methods is then to strike the balance between automation and interaction. If semi-automatic models are the ideal, what should be automated, and where should the designer be able to intervene? What algorithms can be used to aid arrangement while still allowing exploration of the design space? These questions have been explored through the development of a series of computational layout methods over the last few years. This paper presents a selection of these methods as applied to building configurations between 2008 and 2011.

1. Introduction
Spatial configuration, or spatial layout planning, is a ubiquitous problem in architectural design and also one of the most complex. From urban design to building layouts and interiors, spatial configuration is required at every scale: the problem of finding a satisfactory spatial arrangement of elements according to some constraints and objectives. This is by no means restricted to architectural design: similar issues are found in fields from graphic design to circuit-board layout. In some fields the problems are well defined (how to fit several articles onto a page layout for example), and are obvious candidates for solution by computation. In these fields, computer programs exist to automate layout generation and are used as standard by designers. However, in architecture the problems are more usually complex, multi-criteria, and often ill defined; it is not straightforward to frame architectural layout problems with a well-defined set of constraints and objectives suitable for automation by computation.

Despite these difficulties, the first attempts to automate layout generation were made in the
early 1960s and there has been a large amount of research since then; see, for example, Liggett’s review of the field from 1960 to 2000 (Liggett, 2000). However, as Liggett concludes, no development has been successfully integrated by architectural practice: ‘In spite of the long research history associated with automated layout and space allocation systems, in practice these systems have not been utilized to their full potential’ (Liggett, 2000, p. 213).

Much of the research has focussed on optimisation techniques to solve those constraints that can be well defined within a layout problem, while neglecting the creative aspect of ‘designing’. Recently, attempts have been made to resolve this by making applications more interactive and responsive to designers (Arvin and House, 2002; Derix, 2010), but it remains the case that no automated layout programme is used as standard within the architectural profession.

As a research group sitting within an architectural practice, CDR aims to use computation to further architectural research and to develop computational methods to support architects in designing. As part of this remit, the group has been exploring spatial layout problems since 2004. In this paper we will discuss how we have approached these problems, and show examples of simulations developed for projects within the practice.

Defining the problem
Although there is some overlap with the CDR work on spatial configuration in urban design (Derix et al., 2011), this paper will focus on the work on layout generation on a building scale; i.e. the arrangement of spaces within a building, or the generation of a (set of) floorplate layout(s). In practice, this will usually take as input an area schedule for the building, and a set of constraints or objectives (well and less well defined), in addition to the design intentions of the architect.

The constraints (or objectives) are of two types:
- Dimensional constraints (geometrical constraints)
- Topological constraints

Dimensional constraints include the dimensional specifications of each space (area, proportion, orientation), and also constraints such as the boundary outline of the available floorplate. The topological constraints relate to the desired configuration of spaces relative to each other, and will typically include adjacency requirements between areas in the schedule. These adjacency requirements can also be negative, i.e. objectives that require separation of spaces.

2. Approach

Levels of abstraction
One of the main considerations when working on layout problems has been deciding on the appropriate level of abstraction of the problem. An architect will often start work on a layout problem in a purely topological mode, for example through drawing bubble diagrams. Often this approach of disassociating the topological problem from geometrical considerations is desirable, allowing the designer to explore the logic of the assembly with the minimum distractions of dimensional constraints. However, in some cases the topological diagram must be embedded in a geometric representation of some level of detail in order for the resulting arrangement to be meaningful. If the spatial geometry is a key driver or constraint it may be preferable to work with a more complex hybrid of the topological diagram with a basic geometrical description to ground the model in context. We then often work with rectangular spaces, which allow ‘realistic’ packing of areas within a floorplate, but as a side effect are very easily viewed directly as ‘rooms’ rather than diagrammatic representations of areas.

Notes:
1 See (Arvin and House, 2002) for a full explanation of geometric versus topological design objectives as applied to a physically-based space planning model.
then encounter requests from some designers who would like the ability to add, for example, L-shaped rooms to the model. Additions like this do not make sense at this diagrammatic level of many of the early stage layout simulations: the L-shaped rooms would suggest a stage further down the workflow, where a different simulation may be appropriate, with close packing of rooms. At this later stage there are other considerations: the configuration and geometries of circulation spaces must be addressed, the designer must already have decided on the locations of access points within rooms, and so forth. These stages of the design process are clearly different, and the level of abstraction in a layout simulation should be appropriate to the stage of the problem at hand.

Another question is one of scale. If we take a hospital layout, for example, is it necessary to consider the layout of rooms within one department at the same time as considering the configuration of departments within the hospital? Others have argued that, since building layouts are complex arrangements of nested elements, within a model it must be possible to create adjacency objectives that span different scales. While it is conceivable that we may find problems where this is necessary in the future, to date we have had greater success in focussing on one scale for one simulation, or at most allowing the option to switch between scales while working on them independently.

In general we have found that small lightweight simulations are the most effective way to embed computational design methods into the design workflow (see Derix, 2009²), and the spatial layout models have been no exception. Our aim has been to pare down the simulation to contain only essential information, avoiding the increase in computation times that come with increased complexity, and allowing the designer to focus on the pertinent variables without distraction.

### Notes:

² This ‘parsimonious’ approach has been developed by CDR, building on the work of people such as Paul Coates at the University of East London.

---

**Automation vs. interaction**

There is a long history of attempts to automate layout solutions using a variety of methods, as discussed by Liggett (2000), where little or no direct interaction or intervention is possible as the solution is attempted. This approach is one that can work well with well-defined problems where the solution space is simple (if large), and all constraints and objectives can be quantified, and there has been some success in applying these methods to facilities layout problems. The problem of layout generation in an architectural practice is often quite different: the design space is not well defined, and in fact may change throughout the design process. The designer will approach the problem with specific design intentions, and with a wealth of tacit knowledge which cannot be explicitly encoded in the simulation. To integrate these added values there must be a way for the designer to interact with the simulation dynamically, and to affect the outcomes. This is known in some fields of computational design as ‘human-in-the-loop’ simulation, which we have called semi-automated methods. We have made the point before that for the designer to get full benefit from computational models they must be interactive and responsive during use (Derix, 2010). Through instant feedback, the defined constraints and the expression of the underlying heuristics are revealed to the designer, and the effects of design decisions are instantly transparent.

There is then a question with each problem to consider which constraints and objectives should be directly manipulated by the user, which indirectly, and which should be generated automatically. To interact with the simulation fluidly, at least some of the parameters should be directly manipulable within a graphical interface in an intuitive way.

Design is a creative act, and there is a question of how the creative aspect of layout generation is captured in a computational process. Does the computer generate creative solutions to aid the
designer? Or is the computational aspect of the simulation used to solve technical issues, leaving the creative work to the designer? The examples we will show lie at different points on a scale between these poles.

**Algorithmic / heuristic approaches**

What algorithms can be used to aid arrangement while still allowing exploration of the design space? Liggett (2000) reviews the approaches that have been used to tackle layout generation problems, covering single-criterion optimisation techniques, graph theoretic approaches, and multi-criteria problems. The advantages and drawbacks of these approaches are discussed, particularly in regard to areas of application and feasibility.

Arvin and House (2002) give a good overview of physically-based space planning which is a technique that we have also been applying for some time. A physically-based model is one in which elements are modelled as bodies subject to physical forces, and the ‘automation’ of the layout is a result of movement of these elements calculated using principles of physical laws. A major advantage of this approach is the transparency of the design objectives: adjacency relations can be related directly to forces of attraction, and their effects are immediately evident. Another advantage is the simplicity of implementing this approach as a dynamic simulation, updated over small time steps, allowing the designer to directly interact with the model as the layout is being generated: there can be a continuous interplay between designer and computer. However, there are limitations with the applicability of physically-based models, as will be highlighted in the examples.

A very different approach is that of evolutionary search. Here a whole population of initial solutions is generated, and a genetic algorithm is applied to optimally evolve the solutions according to a specified fitness function\(^3\). In this approach, a large portion of the design work is the specification of the fitness criterion; the designer’s interaction with the model is indirect. In this sense the approach sits towards the ‘automated’ end of the scale. However, these methods also allow the search of a much wider solution space and are applicable to problems that cannot be tackled with other approaches, such as packing problems with multi-criteria objectives.

**Defining the simulation**

The simulations presented here mainly fall into the category of meta-heuristic models, which have abstracted some heuristic and generalised it into a search procedure that will always attain a meaningful result, subject to valid data and constraints. Meta-heuristic simulations are not parametric, in that there will never be a one-to-one correspondence between the input constraints/objectives and the final solution. The aim is not to find the one ‘optimal’ solution, but to find good solutions where the process of the search represents the computational model of a creative designing method. Meta-heuristics also have to be adapted to brief; especially since the aim is often to correlate heuristic design processes, there needs to be an alignment mechanism between processes built into the simulation. This means that the design search algorithm (the meta-heuristic) retains its logic but needs to be opened for parallel threads of performances that co-evolve. As we are aiming for lean and transparent models, those might also be separated into linked simulations rather than integrated into large opaque models.

To allow designers to intuitively participate in any of the simulations – automated or highly interactive – there is a requirement to make the algorithmic procedure visually accessible by rendering spatial changes in the calculation process explicitly visible, and in the process to reveal information about the underlying assumptions and model.

---

**Notes:**

3 See Liggett (2000) for a description of genetic algorithms that have been applied to layout automation problems, and Derix (2008) for examples of genetic algorithms applied to produce creative solutions to architectural space design.
3. Examples / case studies
To illustrate some of the points above we present some examples of spatial configuration models that have been developed by CDR of Aedas|R&D between 2008 and 2011. All models have been implemented on real projects within the practice, which has served as a crucial test bed for development.

**Topological models: Automated bubble diagrams**
The automatic layout of bubble diagrams has been explored within the group using agent-based models, graph grammars, attract-repel models and spring systems. These topological layout models are intended to aid the designer in visualising and working with the configurational structure of the building at the conceptual design stage: a dynamic digital heuristic equivalent to the architect’s bubble diagram. Circles are used to represent spaces in the area schedule, the aim being to explore and solve the topological objectives of a layout, without the distraction of geometric constraints.

The dynamic modelling of the bubble diagram is relatively simple: spaces are represented as circles (whose area corresponds to the desired area of the space), and adjacency objectives are shown as lines connecting them. Figure 1 shows the second of these developments from 2007-8. In this graph grammar model, an attractive force acts along the adjacency lines, and a repulsive force ensures there is no overlap between areas, within the 2D plane. Additionally some vertically-stacking elements are included (stairs and lifts). Direct manipulation allows the user to tweak the diagram layout by dragging the circles.

The designer interacts with the model by entering spaces in the graphical user interface, which is configured to allow the designer to define groups of elements, and to build up the schedule as a nested graph, based on recursive embedding of graph descriptions. Adjacency objectives are defined between elements. The designer cannot explicitly define on which floor each element will lie; this is defined automatically by the program by traversing the graph, with only the possibility to specify relative vertical relations. The method for inputting information therefore highlights the underlying graph syntax, and the program leads the designer to better understand the computational logic of the topological structure. While the layout of the elements can be ‘tweaked’ by dragging them within their 2D plane, this is primarily generated automatically according to the underlying graph description. This also means that although a configuration can be saved using the graph syntax, the spatial layout of the bubble diagram will be regenerated when it is reloaded.

Notes:
4 This version was developed by Pablo Miranda at Aedas|R&D.
The tool also contains an analysis of the ‘depth’ of the spaces, or topological distance into the building, calculated using the exterior space as the root. This is equivalent to a measure that has been used as standard by space syntax and others; see, for example, the description of J-graphs as an illustration of depth in *Space is the Machine* (Hillier, 1996, Chapter 1). In this model the depth is represented by colour, stepping down from darker to lighter blue as depth increases, and giving an instant visual feedback on the accessibility of spaces within the building.

**Massing approximations**

A clear intention when developing the bubble diagram models was to avoid mixing the problem of solving the topological configuration with consideration of geometric details. In an architectural project the diagrammatic ‘bubble diagram’ stage allows the designer to concentrate on the logic of the assembly, without clouding the problem with geometric constraints. However, there may be instances where it is necessary or desirable to consider the geometry, or an approximation of the massing, from early in the design process.

One major limitation of the bubble diagram approach is that it is difficult to gauge areas. A key dimensional criterion in layout generation is the constraint of the floorplate shape and size, and it soon becomes necessary to estimate percentages of available area that are occupied within the simulation. This can obviously be achieved simply by outputting a single figure for ‘occupied area’ to compare with the available area, but it is more desirable to visualise the result directly. Since rectangles close-pack, and since most architectural floorplates contain orthogonal boundaries, it makes sense to try a rectangular representation of the areas. The available floorplate can then be represented by an outline boundary, which may act as a constraint to the areas, and it is simple to evaluate visually the achieved areas as the design develops.

This approach was implemented for a competition project in Abu Dhabi, where the architect’s intention was to create an orthogonal ‘aggregate’ model of spaces, with a buffer zone surrounding the accommodation for climate control (Figure 2a). A bespoke application was designed to take the area schedule as input, and load each area as a rectangle with forces of attraction and repulsion representing the desired adjacency relations. This attract-repel approach to automated arrangement from bottom-up principles has been widely used in architectural computation; see, for example, the descriptions of early models developed at the Centre for Evolutionary Computing in Architecture (CECA) of the University of East London (Coates, 2010).

Figure 3a shows the areas overlaid onto the floorplate outlines of the allowable gross-floor area (GFA) envelope as they load into the application. Blue squares, for example, represent units of open-plan office space, while the large grey areas are car parking. Here, both the areas of the accommodation schedule and the adjacency objectives are loaded...
Spatial configuration: Semi-automatic methods

Helme, L., Derix, C. & Izaki, Å.

into the model from CSV files, so the designer can specify desired adjacencies using an adjacency matrix. The model interprets the desired adjacencies by establishing an attractive force between the two spaces. The spaces are also required to stay within the floorplate boundary. Circulation cores are represented by vertically stacked spaces. These are moved manually, not automatically positioned, but other spaces may have attraction forces to the cores. Other circulation space within each floor is allowed for by adding 20% extra area to each space, rather than attempting to solve the layout of circulation spaces concurrently with other variables.

Due to the specified attractive and repulsive forces the areas attempt to rearrange, although it is necessary for the designer to ‘nudge’ the application in order to find a solution (Figure 3b). The designer is also able to gauge the massing as the model evolves by switching between the diagrammatic plan and volumetric views (Figure 3c). Since both the dimensional objectives and adjacency objectives are loaded from file, the designer’s interaction with these constraints is indirect: only the positions can be manipulated directly. However, moving one space immediately sets off a process of automatic reconfiguration, so the effects of the adjacency constraints are ‘felt’ by the user, with instant feedback: the model is fluid and dynamic.

The CSV file format allows the user to save the current spatial layout so that schemes can be reloaded exactly to continue work.

The success of this simulation for this project lay partly in the synergy with the designer’s original intention (Figure 2 shows (a) the original concept sketch next to (b) the final render). Another reason that the attract-repel model worked well was due to the large floorplate available for the defined area schedule: there was space for the elements to ‘flow’ around each other to find suitable configurations, but this excess space was then cut back to give the desired ‘crunchy’ exterior envelope (see Figure 3b), avoiding the common problem of ‘space left over after planning’ (SLOAP).

Due to interest within the practice, the approach taken above was modified and further developed to produce a layout ‘tool’, for use in projects such as schools (see Figure 4). Key concepts were retained, such as:

- loading of the schedule from CSV file;
- the ability to save the spatial layout exactly;
- 2D-diagrammatic and 3D-massing views;
- attractive forces along desired adjacencies;
- the option to restrict movement within a boundary;
- and the hybrid of automatic arrangement and direct manipulation to achieve layout.

The simulation uses a damped model of physical forces similar to the previous example, implemented using iterative separation-attraction displacements.
of the elements at each time step as the application runs (see Figure 5). Some improvements have been made, such as the inclusion of alignment objectives; see Figure 5c. One main change to the interaction is that in this development the adjacency links can be modified directly within the graphical interface by clicking on the rooms in question to add and remove links. There is therefore a much more direct interaction between the user and the constraints/objectives they are specifying; when a link is added the spaces immediately try to draw closer together. Lines representing the adjacency links make it easy for the user to evaluate visually how well the adjacency objectives they have specified are being met by a layout. The dimensional constraints can also be modified within the graphical interface, both graphically and through a dialogue box (Figure 6). In fact, there is the option to add all spaces directly through the graphical interface if the user prefers.

Figure 4:
Physically-based layout tool, applied to a school.
Left: view of one floor; right: screenshot of tool in use showing floor layouts for all floors extruded to give an indication of massing, and readout of areas achieved.

Figure 5:
Elements of the interactive physically-based model used in our layout tool. At each time step, two updates are made to the position of each room element.

(a) Separation: any overlapping elements are moved apart along the line joining their centres.
(b-d) Attraction: each room element is moved along a vector that is the sum of attraction vectors to all elements linked by adjacency links. Attraction vectors between two elements linked with an adjacency relation lie along the line between their centres (a) unless the elements are in contact, in which case the vector is oriented to achieve alignment; (b) the distance moved along the resultant attraction vector is subject to a damping factor that smooths the real time behaviour of the application.
Movement driven layout
A feature of the above models is that the issue of circulation space is avoided, other than the inclusion of vertical cores. In the next example we experimented with how the configuration of space can influence, or be influenced by, the movement of people, and how an approximation of a circulation diagram might be used to drive the spatial layout within a building.

This project was a research collaboration between CDR and the Fraunhofer IAO to explore the potential for combining computational design methods with immersive technologies like Virtual Reality.
Reality (VR) in early-stage building design (Krause et al., 2011). Three applications were developed to test different stages of the design process and tested within the VR setup developed by the Fraunhofer IAO team, one of which relating to generation of spatial layout is described here.

In this application the desired area schedule is again loaded from a CSV file, and the floorplate extents from DXF. Each ‘use’ within the area schedule (e.g. office space, laboratories, lobby) is associated with one of the more cubic nodes, which mark regional centres. These can be moved in space by the designer by dragging within the graphical interface (or VR interface in the final application), but are not automatically arranged by the application.

The user can add movement routes between the centres of use which are visualised as blue ‘threads’ (see Figure 8a). We experimented with using these movement routes to generate a first approximation of circulation spaces within the building. An inspiration for this approach was the work of Frei Otto and others at the Institute for Lightweight Structures (ILEK), Stuttgart, who experimented with wool threads in water as an analogue model to compute minimal path systems (Otto, 2008). Our digital model shows similar qualities (Figure 9): routes draw together to create an aggregated system which is overall of shorter length, but at a time cost for movement between any two of the linked nodes. This aggregation process is simulated dynamically by modelling each unit of ‘thread’ as connected to adjacent units on the same route by springs, and with forces of attraction to units of thread on other routes\(^6\).

The resulting circulation diagrams are not meant to be taken literally, but are used to ‘carve out’ areas of circulation space within the possible built volume (Figure 8b, c). Movement patterns can then be further analysed to suggest emergent unplanned ‘meeting points’ within the circulation space that in turn might influence the behaviour of employees (Figure 8d). Those meetings points are interpolated.
at intersections of routes with a minimum flow, so that encounters between occupants are highly likely and social situations occur. This emergent proposition by the simulation reflects a currently perceived creative rule-of-thumb in workplace design for face-to-face communication, which is meant to stimulate productivity. Clearly, the selection of a reduced set of such locations and dimensioning would currently have to be done manually as the simulation only helps to identify potentially valuable social areas. The remaining volume is attributed to each ‘use’ within the area schedule to try to fulfill the area objectives, using the corresponding node positions to preferentially locate area (Figure 8b shows the volume divided by coloured volumes representing areas, as shown in the key).

Notes:
7 A polyomino is a generalisation of the domino to a collection of $n$ squares of equal size arranged with coincident sides.
8 The genetic algorithm employed uses fitness-proportionate selection (or roulette wheel selection).

Figure 9:
Circulation tests on a network with five nodes: dynamic simulation of ‘attraction’ of routes to each other approximates minimal paths.
From left to right, total length of routes decreases, but with a cost to journey time between any two points.

**Packing models**
As discussed earlier, there are situations where physically-based models with forces of attraction are not suitable. We give just two examples here of projects where the spatial layout problem was quite different: one of packing residential units into a tower. The first was a competition project, where the apartment layouts were three-dimensional; the second a residential high-rise with a catalogue of apartment types to arrange on each floorplate with access constraints. Here a computational model was needed to creatively solve a combinatorial problem that was almost impossible to complete by hand, and the input of the designer was almost entirely constrained to defining the objectives of the simulation, with no direct manipulation during the solution.

Figure 10 shows sketches made to define the parameters of the simulation. The concept was for units within a cylindrical residential tower to be composed of segments of the circular floorplate, with two, three and four-bed apartments being made up of differing numbers of segments. Since the segments could be arranged on more than one floor, and obviously must be contiguous, the resulting profiles resemble ‘polyominoes’.” The problem of how to close-pack polyominoes is well known from the game Tetris in which the player aims to close-pack tetrominoes, but becomes much more complex when more than four-sided pieces are introduced. The problem is in fact found to be NP-complete. The ‘close-packing’ objective relates to maximising available floorplate area, a key objective for developers. On top of this, the aim was to introduce other objectives, such as a preference for larger (and more expensive) apartments to gravitate to the top of the tower, for better views.

In order to find a good (but not optimal) solution, an evolutionary algorithm was employed. Evolutionary algorithms as applied to computational design problems are described fully in Derix (2008). A population base of 200 was used to evolve the solution, with a standard evolutionary approach of mutation and crossover, and using the total area achieved as the fitness criterion. Figure 11 shows
the best of the generation at six points through the simulation: it is evident that the floorplate area and number of units improves, although the packing ratio never reaches 100% with this method. Since the project was for a competition on a tight timescale, the application could not be pushed further in the time available. However, the aim would have been to refine the genetic algorithm, balancing the fitness criterion between the desire for the correct mix of apartment types, use of all the available floorplate area, and objectives such as the larger apartments lying higher within the tower. Eventually, the elevation of the tower was ‘un-rolled’ and a set of clearly specified apartments used as a catalogue to help resolve the genetic algorithm without updating the selection and fitness functions. Several fully target

<table>
<thead>
<tr>
<th>Generation</th>
<th>Units</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>717m²</td>
</tr>
<tr>
<td>2</td>
<td>17</td>
<td>893m²</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>1067m²</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
<td>1229m²</td>
</tr>
<tr>
<td>n</td>
<td>87</td>
<td>1520m²</td>
</tr>
</tbody>
</table>

Figure 10: Concept sketch for ‘3D tetris’ packing of 2, 3 and 4-bed residential apartments in a tower, showing viable configurations and desired areas.

Figure 11: ‘Polyomino’ packing of apartments for a residential tower, using a genetic algorithm. Images are of the best individual of each generation shown, selected according to criterion based on maximising total floorplate area.

Figure 12: The steps showing the un-rolled elevation being evolved to pack the polyomino with constraints such as larger apartments at top (left), their equivalent layout in plan (middle) and translated volumetric composition for the final tower design.
compliant solutions were generated that could subsequently be re-translated into the intended volumetric apartment configurations.

The use of genetic algorithms for residential unit layout was revisited in more detail in a later project for a residential high-rise in London. This time the problem was simplified in one way as the footprints of each apartment lay within one plane, but the design objectives (and therefore the fitness criteria) were more sophisticated. As input, a possible thirteen simplified apartment layouts were abstracted onto a grid with certain constraints specified with regards to daylight, balconies and entrance location. These are shown in Figure 13. Similarly, the floor that the units would populate had to be specified in terms of the boundary of the floorplate, location of stairwells and service cores, and the program that the floors had to accommodate in terms of number of units of each size. An evolutionary search process was then applied to arrange the specified number of units within the footprint boundary. The program can evolve the layout by replacing units from others in the set of one-bed, two-bed or three-bed apartments, and by applying rotation, translation and mirror operations. The fitness criteria encapsulate the design constraints, such as each apartment’s accessibility from lift shaft and stairwell (the access from apartments to core services is calculated as permeability on the underlying grid), daylight available to window areas, and balcony areas situated at the boundary to the building. Figure 14 shows two examples of layouts generated with the model.

Figure 13:
The 13 possible apartment layouts, showing constraints on positions of balconies, and daylight requirements.

Figure 14:
Examples of apartment arrangement options generated for two floors,
(a) with four 2-bed and three 1-bed units, and
(b) with nine 1-bed units.
4. Conclusion
The methodology of CDR is to produce small, modular applications to assist the designer in specific problems:

‘The lightweight applications must be limited in functionality (parsimonious) and visualize in real-time the “intentions” of the simulation as it searches, to render its heuristics transparent.’

(Derix, 2010, p. 64)

We have found this to be true for spatial configuration problems as much as any other. While other researchers may strive to achieve the ultimate goal of one application to suit all layout problems, we believe this is neither achievable nor even desirable within a live design workflow.

We have had a lot of success with physically-based systems for layout generation, as these offer good scope for dynamic user-interaction and clear interpretation of constraints such as desired adjacencies. However, there are plenty of cases where these are not applicable: they fall down where close-packing is needed, or a more detailed geometrical description of the problem, and for these problems we have looked to other algorithms/heuristics. On the other hand, most examples given have produced surprising propositions that could be considered as creative solutions, as designers would have not anticipated them. Obviously, the opportunity for emergence is encoded but the interpretation of local conditions and triggering a proposition is then up to the simulation. Both the evolutionary and the interactive case studies encoded the opportunity for surprise solutions that, however, comply with the full set of constraints. Essentially, the methods described are amalgamating design stages that are otherwise isolated, such as the circulation and area allocation simulation for the Fraunhofer Institute. Synthesising constraints and design procedures via physical simulation and virtual immersion allows for unknown states to emerge, producing creative configurations from a creative software design.

Additional constraints applied to generative systems – be they physically-based or on other heuristics such as the evolutionary – are shown that represent complementary systems such as simulating access pattern diagrams. Those complementary evaluations do not simply represent constraints but co-evolving systems that allow less rigorously specified architectural aspects of configurations like public spaces to be computed from hard constraints like areas and adjacencies. In practice, this approach appears promising, as performances of spaces always need to be represented as evidence of explicit targets.

We conclude that it is necessary to have the flexibility of multiple approaches available for application to the wide variety of spatial configuration problems that present themselves in architectural practice.

About the authors:
Lucy Helme is a senior designer within the Aedas R&D Computational Design + Research group, having joined the practice in 2009. Since then she has been developing and applying computational techniques on a wide range of architectural projects, from masterplanning to data visualisation. She also regularly communicates the work of the group through lectures and university teaching.

Prior to joining Aedas, Lucy completed an MA in Industrial Design Engineering at the Royal College of Art, graduating in 2008. During her final year project she developed a system for the design and manufacture of customisable small buildings for the domestic garden using generative design and computer aided manufacturing techniques. Lucy also has a strong background in science, completing a PhD in materials physics in 2006 and publishing in leading journals in the field.

Christian Derix (christian.w.derix@gmail.com) is director of the Design Futures Lab (DFL) at WoodsBagot Architects. Previously, he founded and directed the Computational Design Research group (CDR) of Aedas Architects in London (UK) since 2004. CDR and now the DFL develops computational simulations for generative and analytical design processes with an emphasis on spatial configurations and human occupation. Derix studied architecture and computation in Italy and the UK and has taught the subject at various European universities since 2001, including University of East London, University College London, Milan Po-
Spatial configuration: Semi-automatic methods

Helme, L., Derix, C. & Izaki, Å.

References


