# Decision Chain Encoding: Evolutionary Design Optimization with Complex Constraints

Patrick Janssen and Vignesh Kaushik

Department of Architecture, National University of Singapore
patrick@janssen.name, vigneshkaushik@gmail.com

**Abstract.** A novel encoding technique is presented that allows constraints to be easily handled in an intuitive way. The proposed encoding technique structures the genotype-phenotype mapping process as a sequential chain of decision points, where each decision point consists of a choice between alternative options. In order to demonstrate the feasibility of the decision chain encoding technique, a case-study is presented for the evolutionary optimization of the architectural design for a large residential building.

**Keywords:** evolutionary, multi-criteria optimization, constraints, encoding, decoding.

## 1 Introduction

Evolutionary design is an approach that evolves populations of design variants in order to optimise certain performance measures. Designs are manipulated by a set of computational procedures, including a development procedure for generating design variants, one or more evaluation procedures that use simulation and analysis for ranking design variants, and a feedback procedure for closing the loop by linking results from evaluation to the input of development.

If the designs being evolved have limited variability, then the developmental procedure can use direct parametric modelling for generating designs. With this approach, genes are directly linked to parameters within the model. However, in cases where complex designs have to be evolved, the development procedure may consist of an indirect rule-based procedure for generating designs Genes are then linked to parameters in the rules rather than in the model, and as a result only affect the final form indirectly via the rules (Frazer 1995, Janssen 2004). In the context of evolutionary design, such rule-based developmental procedures have been referred to as computational embryogenies (Kumar and Bentley 2003).

This paper presents a novel rule-based modelling technique that is particularly well suited for generating complex designs as it allows constraints to be easily handled in an intuitive way. In order to demonstrate the feasibility of the decision chain encoding technique, this paper presents a case-study for the multi-objective evolutionary optimisation of the design of the Interlace, a large residential project designed by OMA and currently under construction in Singapore (OMA 2013). The design uses a 'staggered brick' pattern, where 31 building blocks are stacked up on top of one another in a brick pattern. Previous research by Janssen and Kaushik (2012) described a simulation driven method for optimizing the design through a series of manual iterations. This paper now takes this research further by proposing an automated procedure for design optimisation.

Section 2 describes decision chain encoding in more detail. Section 3 presents the case study, where decision chain encoding is used in a multi-objective evolutionary optimisation problem. Section 4 briefly draws conclusions and indicates avenues of further research.

## 2 Decision Chain Encoding

The proposed encoding technique structures the genotype-phenotype mapping process as a sequential chain of decision points. Each decision point involves choosing one option from the set of all valid options. The set of valid options is created by a set of rules that generate and filter options. The genotype consists of a list of real-valued genes in the range {0,1}. For each decision point, a gene will be used to select an option by mapping it to an integer value in the range {1,n}, where n is equal to the total number of valid options for that decision point. Note that for each decision, the total number of valid options may not be known and may depend on the previous decisions.

### 2.1 Travelling Salesman Problem

As an example, the Travelling Salesman Problem (TSP) may be used. If we have 8 cities (labeled as A to H), then assuming we start at city A, the decision chain would consist of 6 decisions (since the last city does not require a decision as there will only be one city left). For this case, each decision will select one city. For the first decision, there would be a total of 7 cities to choose from, so the gene would be mapped to an integer value between 1 and 7 and the chosen city would then be removed from the list of remaining cities. For the second decision, there would be a total of 6 cities to choose from, so the gene would be mapped to an integer value between 1 and 6. Table 1 shows a set of gene values and the sequence of cities that would be chosen, with the final sequence being A,G,F,H,D,E,B,C.

**Table 1.** The process of selecting cities using the decision chain encoding method

| Decision point | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Gene value | 0.77 | 0.69 | 0.94 | 0.63 | 0.84 | 0.48 |
| Remaining cities | 7 | 6 | 5 | 4 | 3 | 2 |
| Integer mapping | 6 | 5 | 5 | 3 | 3 | 1 |
| Chosen city | G | F | H | D | E | B |

For the typical TSP problem, there is little advantage to this approach over other approaches. However, the decision chain encoding method can easily handle additional constraints. For example, if direct travel between cities F and H is disallowed, then the rules for generating and filtering options could easily be modified. The sequence of cities for the genes in Table 1 would become A,G,F,E,D,H,B,C.

In general, the proposed decision chain encoding technique is seen to be useful for highly constrained problems. Researchers have identifed four main approaches to handling constraints in evolutionary algorithms: 1) penalty functions that reduce the fitness of invalid solutions, 2) repair functions that modify invalid solutions, 3) specialised reproduction opeators that avoid invalid solutions, and 4) specilaised genotype to phenotype decoder functions that avoid invalid solutions (Eiben and Smith 2003, pp 210-211). The the fourth approach has the advantage of permitting the use of standard variation operators. Handling constraints through decision chain encoding falls into the fourth category.

## 2.2 Evolutionary Design Method

Within the current research, the decision chain encoding technique is used as a way of handling constraints in evolutionary design optimisation. Design optimisation typically requires design solutions that are highly constrained, and therefore decision chain encoding is seen as being an appropriate technique.

The research aims to develop optimisation tools that can be used by designers. It is assumed that the designers using such tools will have limited programming skills, and will therefore need to be able to define the key problem-specific procedures without having to write computer code. The development procedure and one or more evaluation procedures are therefore defined using Visual Dataflow Modelling (VDM) tools (Janssen and Chen 2011).

Visual Dataflow Modelling has becoming increasingly popular within the design community, as it can accelerate the iterative design process, thereby allowing larger numbers of design possibilities to be explored. Modelling in a VDM system consists of creating dataflow networks using nodes and links, where nodes can be thought of as functions that perform actions, and links connect the output of one function to the input of another function. VDM is now also becoming an important tool in performance-based design approaches (Shea et. al. 2005, Coenders 2007, Lagios et. al 2010, Toth et. al. 2011, Janssen et. al. 2011).

In this research, an advanced procedural modelling system called SideFX Houdini is used for both development and evaluation procedures. For the development procedure, VDM networks are created in Houdini to generate the three-dimensional models of design variants. These networks use the decision chain encoding technique for constructing models. At each decision point in the modelling process, a set of rules is used to generate, filter, and select valid options for the next stage of the modelling process, as shown in figure 1. The generate step uses the rules to create a set of options. The filter step discards invalid options that contravene constraints. The select step chooses one of the valid options. In order to minimise the complexity of the modelling process, options are generated in skeletal form with a minimum amount of detail. The full detailed model is then generated only at the end, once the decision chain has finished completing.

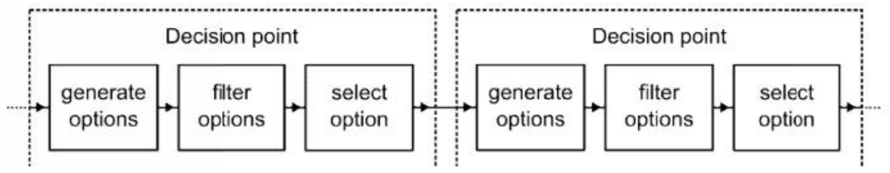


**Fig. 1.** Key steps executed at each decision point in the developmental procedure

## 3 Case-Study

The case study experiment is based on the design of the Interlace by OMA. The design consists of thirty-one apartment blocks, each six stories tall. The blocks are stacked in an interlocking brick pattern, with voids between the blocks. Each stack of blocks is rotated around a set of vertical axes, thereby creating a complex interlocking configuration. An example is shown in figure 2, where 6 blocks are stacked and rotated to form a hexagonal configuration.

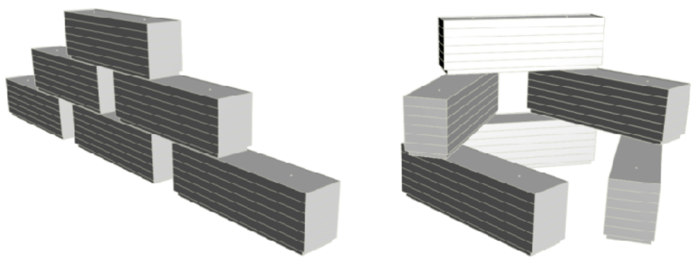


**Fig. 2.** The staggered brick pattern. The diagram on the left shows 6 blocks arranged in a straight line, while the diagram on the right shows the same six blocks folded into a hexagonal pattern.

Each block is approximately 70 meters long by 16.5 meters wide, with two vertical axes of rotation spaces 45 meters apart. The axes of rotation coincide with the location of the vertical cores of the building, thereby allowing for a single vertical core to connect blocks at different levels. The blocks are almost totally glazed, with large windows on all four facades. In addition, blocks also have a series of balconies, both projecting out from the facade and inset into the facade.

The initial configuration, shown in figure 3, is based on the original design by OMA. The blocks are arranged into 22 stacks of varying height, and the stacks are then rotated into a hexagonal pattern constrained within the site boundaries. At the highest point, the blocks are stacked four high.

For the case study, new configurations of these 31 blocks were sought that optimise certain performance measures. For the new configurations, the size and number of blocks will remain the same, but the way that they are stacked and rotated can differ.

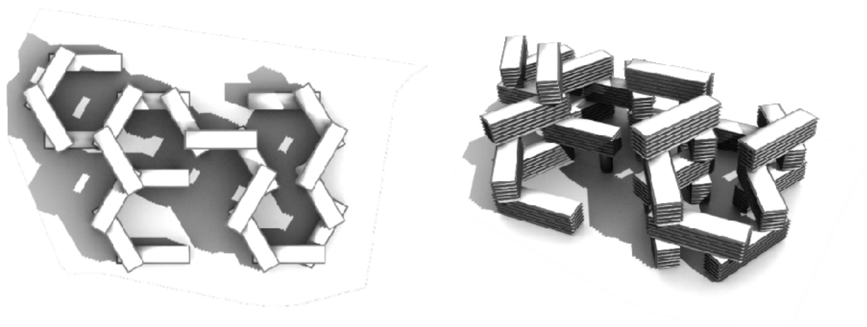


**Fig. 3.** The initial configuration based on the original design, consisting of 31 blocks in 22 stacks of varying heights

## 3.1 Development

The developmental procedure is defined using decision chain encoding. In this procedure, the placement of each of the 31 blocks is defined as a decision point. The process places one block at the time, starting with the first block on the empty site. At each decision point, a set of rules is used to generate, filter, and select possible positions for the next block. Each genotype has 32 genes, and all are real values in the range {0,1}.

In the generation step, possible positions for the next block will be created using a few simple rules. First, locations are identified, and second orientations for each location are identified. The locations are always defined relative to the existing blocks already placed, and could be either on top of or underneath those blocks. The orientations are then generated in 15° increments in a 180° sweep perpendicular to either end of the existing block. In the filtering step, constraints relating to proximity between blocks and proximity to the site boundary are applied, thereby ensuring that only the valid positions remain. In the selection step, the decision gene in the genotype chooses one of the valid block positions.

When generating a new design variant, the first decision point involves selecting a starting point on the site from a set of possible starting points. These starting points are generated by overlaying a grid over the site and then filtering out all points that lie outside the boundary of the site. The next four decisions points are show in figure 4. In the diagrams, the numbered lines are used to indicate possible valid block positions, so that the next block could be placed on any of those lines. The selected option for that decision point is shown as a thicker line.

- For decision point 2, the first block needs to be placed on the starting point. The generative rules create 12 possible positions for the block, orientated around the starting point, labelled 0 to 11. The gene selects position number 2.
- For decision point 3, the second block has to be placed on either end of the first block. The generative rules find two locations where blocks can be placed, and they create 7 positions at each location, resulting in total of 14 possible positions. The gene selects position number 1. Note that this block now has one end unsupported.

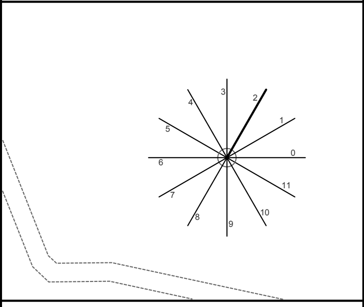
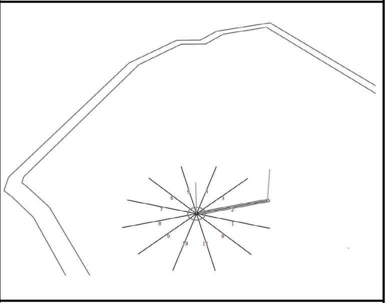
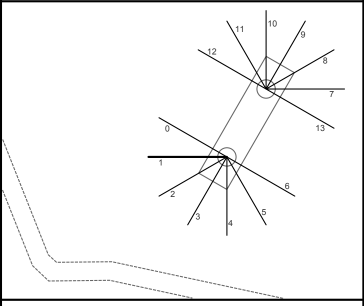
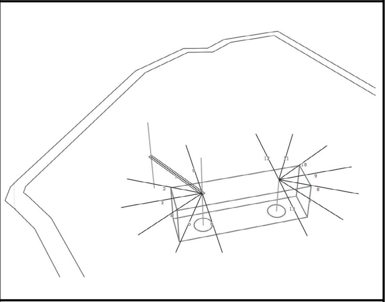
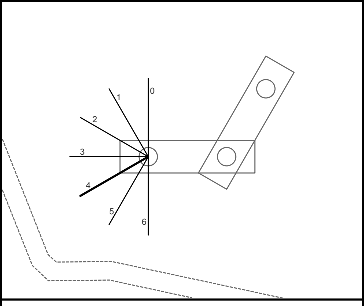
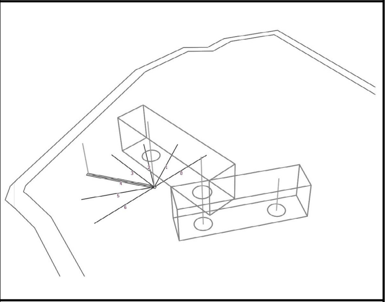
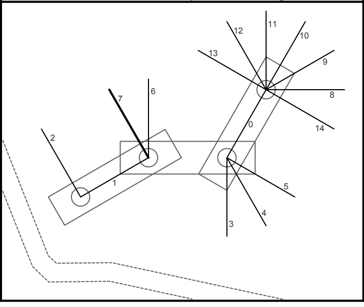
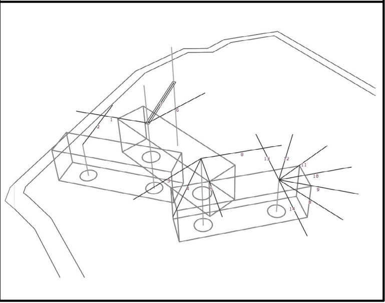
| | Plan view | 3D view |
|---|---|---|
| Decision point 2 |  |  |
| Decision point 3 |  |  |
| Decision point 4 |  |  |
| Decision point 5 |  |  |

**Fig. 4.** Four decision points in the process of mapping the genotype to the phenotype

- For decision point 4, the third block has to be placed. In this case, the generative rules give preference to the location that will result in a support for the previous block. This results in 7 positions, all underneath the previous block. The gene selects position number 4.
- For decision point 5, the fourth block has to be placed. The generative rules find four locations, all on top of the three already placed blocks. The rules create 28 possible positions. However, a number of these positions violate constraints, and are therefore discarded by the filtering rules. Figure 5 shows the filtering stages for decision point 5. Overall, 7 positions are discarded as they would result in blocks that would be too close to the three existing blocks, and 6 positions are discarded as they would result in blocks that extend outside the site boundary. This results in 15 remaining positions, from which the gene selects position number 7.

The process shown in figure 4 is continued until all 31 blocks are placed. The developmental procedure using the decision chain encoding technique ensures a match between the space of all possible genotypes and the space of all valid phenotypes. This results in two critical features: first, it guarantees that any genotype will map to a valid phenotype; second, it guarantees that the all valid phenotypes can be generated from a genotype. It is typically very difficult to create genotype-phenotype decoding procedures that appropriately control the variability of three-dimensional design objects with complex relationships and constraints. Decision chain encoding has enabled the variability problem (Janssen 2004) to be overcome.

As a final stage of the developmental procedure, cores and facades are added to the blocks. The cores (which would contain the lifts, service shafts, and escape stairs) are added to the interior of all blocks, and in some cases also inserted below blocks in those instances where a void remains below the block (in order to provide support for the block, and to ensure that the flats are accessible from the ground floor). For the façades, the windows on each of the blocks are analyzed, and both sunshades and glazing systems are created. The size of the shades and the glazing system types both depend on the amount of solar radiation incident on the windows throughout the year. This is calculated in a two-step process, using the Radiance simulation program (Janssen and Kaushik 2012). In step 1, the solar radiation incident on the windows without

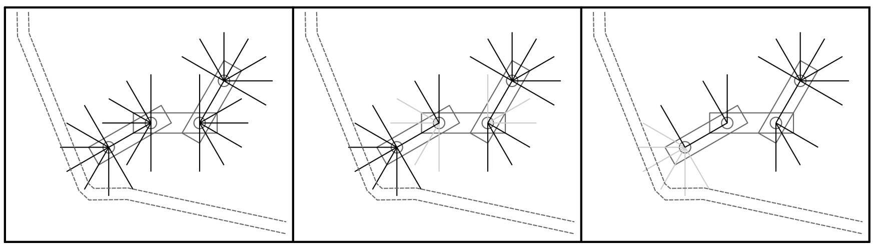


**Fig. 5.** Filtering of positions that violate constraints for decision point 5. (a) All 28 possible positions; (b) 21 positions after block-based filtering; (c) 15 positions after site-based filtering

shading is simulated, and shades are then generated so that windows with more sun will get larger shades. In step 2, the solar radiation incident on the windows with shading is simulated, and windows which are still receiving too much solar radiation are then assigned more expensive glazing systems which are able to limit the amount of solar radiation.

## 3.2 Evaluation

For the multi-objective evaluation, three procedures are defined for evaluating performance of the building. The building is located in the tropics in Singapore, and two key requirements are to maximize the amount of daylight and to minimize the amount of solar radiation entering the windows (which is seen as the worst case scenario - see Janssen and Kaushik (2012) for more details). Both these factors are affected by the orientation of the blocks relative to the north direction, and relative to one another (due to inter-block shading).

For daylight, an evaluation procedure is defined that executes Radiance in order to calculate the amount of light reaching the window on a cloudy overcast sky. The amount of light entering each window is then adjusted according to the visual transmittance of the glazing system for that window. The performance criterion is defined as the maximization of the total number of windows where the light entering the window is above a certain threshold level for reasonable visual comfort, referred to as ‘good daylight windows’.

As described in the previous section, the minimization of solar radiation entering the building is already tackled by the developmental procedure by adding the sun shades and high performance glazing systems. However, these additional systems have a significant effect on the cost of the façade, and therefore the performance criteria in this case actually focuses on the cost of the façade. For façade cost, an evaluation procedure is defined that calculates the total cost of all the glazing systems and shading systems for all 31 blocks. The performance criterion is defined as the minimization of this cost, referred to as ‘façade cost’.

Lastly, one more performance criterion is added, relating to the cores. As discussed in the previous section, the developmental procedure will generate design variants where additional cores need to be inserted. These cores will add significant additional cost, and therefore need to be minimized. The final evaluation procedure therefore calculates the total length of core for all the blocks. The performance criterion is defined as the minimization of the total vertical core length.

## 3.3 Results

The evolutionary process was executed using Dexen, a distributed execution environment for population based optimisation algorithms (Janssen et. al. 2011). A set of 10 networked PCs was set up consisting of one server and 9 compute nodes (each with 4 slaves). The execution time to develop and evaluate a single design variant on one machine was close to two minutes, but when it was run using Dexen with the 9 compute nodes, it was reduced to approximately 18 seconds.

The population size was set to 200 and a simple asynchronous steady-state evolutionary algorithm was used. Each generation, 50 individuals were randomly selected from the population and ranked using multi-objective Pareto ranking. The 4 individuals with the lowest rank were killed, and the 4 individuals with the highest rank (rank 1) were used as parents for reproduction. Standard crossover and mutation operators for real-valued genotypes were used, with the mutation probability being set to 0.01. Reproduction between pairs of parents resulted in 4 new children, thereby ensuring that the population size remained constant.

The evolutionary algorithm was run for a total of 16,000 births, taking approximately 80 hours to execute. In order to calculate the progress of the evolutionary algorithm, the Hypervolume metric was used (Zitzler and Thiele 1998). At each 100 births, the non-dominated Pareto set was found. For each Pareto set, the performance scores were normalized, and the good daylight window score was inverted so that all scores are being minimized. The Hypervolume was then calculated using the tool developed by Fonseca et. al. (2006). The graph in figure 6 shows the increase in Hypervolume as evolution progresses.

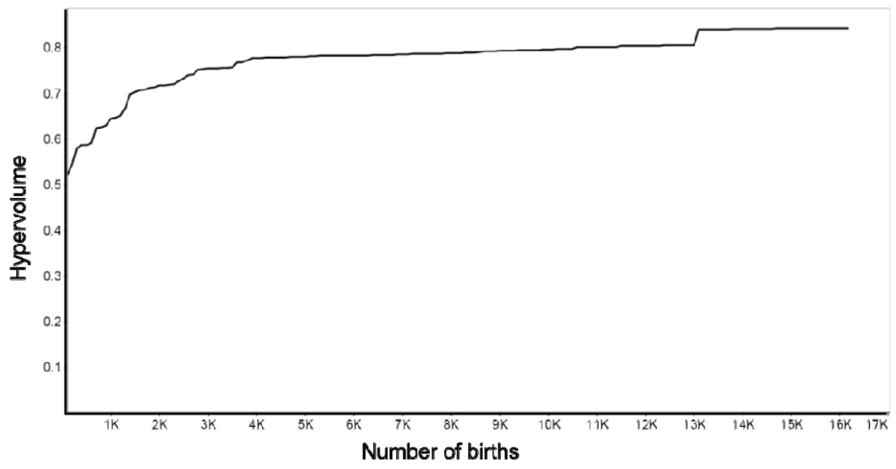


**Fig. 6.** The Hypervolume graph for a run of 16,000 individuals

The final non-dominated Pareto set for the whole population contains a range of design variants with differing tradeoffs between performance and cost. One of the design variants from this non-dominated set is shown in figure 7. The performance scores for the initial design shown in figure 2 are: good daylight windows: 70 %; façade cost: SGD 44.5 million; and core length 1481 meters. For the design shown in figure 7, the performance scores are as follows: good daylight windows: 83 %; façade cost: SGD 42.3 million; and core length 1504 meters. The optimized design is therefore cheaper than the original design, but also performs better in terms of daylight performance.

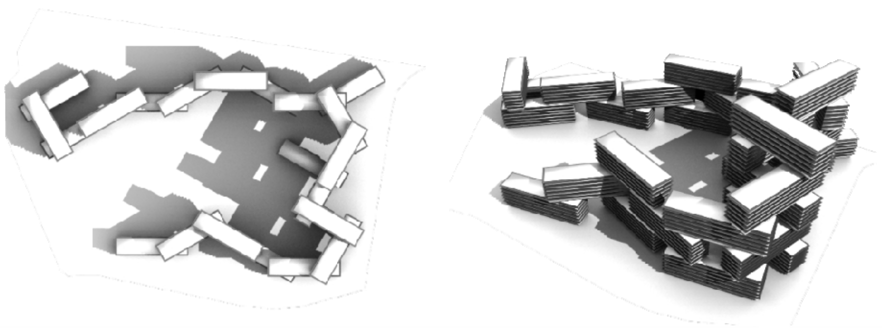


**Fig. 7.** A selected design on the non-dominated Pareto set

## 4 Conclusions

Decision chain encoding is an effective way of handling complex sets of constraints. The case-study demonstration has shown how the decision chain encoding technique can be applied to the evolution of a complex design. Furthermore, due to the simplicity of the way that constraints are handled, developmental procedures using decision chain encoding can be implemented using VDM systems. This allows designers with limited programming skills to engage with evolutionary design methods.

Future research will compare the performance of decision chain encoding techniques to other techniques such random key encoding (Bean 1998). In particular, it is noted that the decision chain encoding technique results in genotypes that have high epistasis, in that genes early in the genotype sequence have a significant impact on the expression of the genes later on in the sequence. Benchmarking experiments will be performed in which evolutionary algorithms using decision chain encoding are compared to evolutionary algorithms using alternative encoding techniques.

## References

1. Bean, J.: Genetic Algorithms and random keys for sequencing and optimization. ORSA Journal of Computing 2(2), 154–160 (1992)
2. Coenders, J.L.: Interfacing between parametric associative and structural software. In: Proceedings of the 4th International Conference on Structural and Construction Engineering, Melbourne, Australia (2007)
3. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 1st edn. Natural Computing Series. Springer (2003)
4. Fonseca, C.M., Paquete, L., Ibáñez, M.L.: An Improved Dimension - Sweep Algorithm for the Hypervolume Indicator. In: Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006), pp. 1157–1163. IEEE Press, Piscataway (2006)
5. Frazer, J.H.: An Evolutionary Architecture. AA Publications, London, UK (1995)
6. Janssen, P.H.T.: A Design Method and a Computational Architecture for Generating and Evolving Building Designs. School of Design, Hong Kong Polytechnic University. Degree of Doctor of Philosophy (2004)

7. Janssen, P.H.T., Basol, C., Chen, K.W.: Evolutionary Developmental Design for Non-Programmers. In: Proceedings of 29th eCAADe Conference, Ljubljana, Slovenia, September 21-24, pp. 245-252 (2011)
8. Janssen, P.H.T., Chen, K.W.: Visual Dataflow Modelling: A Comparison of Three Systems. In: Proceedings of the CAAD Futures 2011, Liege, Belgium, July 4-8, pp. 801–816 (2011)
9. Janssen, P.H.T., Chen, K.W., Basol, C.: Iterative Virtual Prototyping: Performance Based Design Exploration. In: Proceedings of 29th eCAADe Conference, Ljubljana, Slovenia, September 21-24, pp. 253–260 (2011)
10. Janssen, P.H.T., Kaushik, V.: Iterative Refinement through Simulation: Exploring trade-offs between speed and accuracy. In: Proceedings of the 30th eCAADe Conference, Prague, Czech Republic, September 12-14, pp. 555–563 (2012)
11. Kumar, S., Bentley, P.J.: Computational embryology: Past, Present and Future. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computing: Theory and Applications, pp. 461–477. Springer, New York (2003)
12. Lagios, K., Niemasz, J., Reinhart, C.F.: Animated Building Performance Simulation (ABPS) – Linking Rhinoceros/Grasshopper with Radiance/Daysim. In: Proceedings of SimBuild, New York City (2010)
13. OMA (2013), `http://oma.eu/projects/2009/the-interlace`
14. Toth, B., Salim, F., Frazer, J., Drogemuller, R., Burry, J., Burry, M.: Energy-oriented Design Tools for Collaboration in the Cloud. International Journal of Architectural Computing 4(9), 339–359 (2011)
15. Shea, K., Aish, R., Gourtovaia, M.: Towards Integrated Performance-Driven Generative Design Tools. Automation in Construction 14(2), 253–264 (2005)
16. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)