Analysis

Behavior

The Generative Multi-Performance Design System

Generative

Anas Alfaris

Massachusetts Institute of Technology

Riccardo Merello

Arup

Optimization

Performance

THIS PAPER PROPOSES A FRAMEWORK FOR AN INTEGRATED COMPUTATIONAL DESIGN SYSTEM. **This design system builds on the strengths inherent in both generative synthesis models and multi-performance analysis and optimization. Four main design mechanisms and their mathematical models are discussed and their integration proposed. The process of building the design system begins by a top-down decomposition of a design concept. The different disciplines involved are decomposed into modules that simulate the respective design mechanisms. Subsequently through a bottom-up approach, the design modules are connected into a data flow network that includes clusters and subsystems. This network forms the Generative Multi-Performance Design System. This integrated system acts as a holistic structured functional unit that searches the design space for satisfactory solutions. The proposed design system is domain independent. Its potential will be demonstrated through a pilot project in which a multi-performance space planning problem is considered. The results are then discussed and analyzed.**

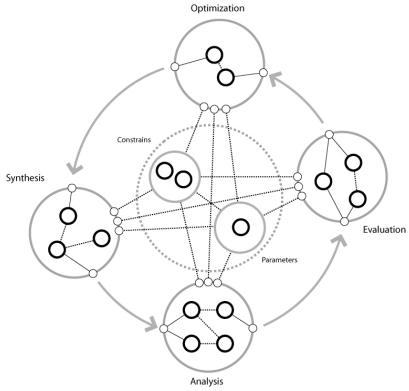


FIGURE 1. THE GENERATIVE MULTI-PERFORMANCE DESIGN SYSTEM IS A COMPOSED OF FOUR MAIN PHASES. EACH PHASE IS A CLUSTER OF MODULES.

1 Introduction

Design involves solving what Herbert Simon terms an ill-structured problem (Simon, 1973). An ill-structured problem is one that cannot be solved by a linear chain of reasoning derived from the problem statement. Furthermore, it might not have a unique solution but a multiplicity of solutions. These design problem characteristics imply the need for many assumptions within the design process that can only be verified after a solution is reached.

This makes computational design systems a difficult area of study. Although recent possibilities provided by advances in computational power and new developments in performance analysis tools offer architects and engineers information that can assist in decision making, insight gained through these tools remains limited. This is due to their inadequate ability of searching the design-space satisfactorily. In addition, since these tools are usually discipline specific, they lack the ability to supply sufficient understanding of the possible tradeoffs between the different disciplines. Creating effective computational design systems requires the development of new ways to represent designs and to evaluate their different disciplines performance collectively.

Generative synthesis models present a powerful formalism that can generate solutions within the design space defined by the system's design language. Current design problems are multi-disciplinary, and for architects and engineers to adapt to a highly competitive global market, the need of integrating various performance criteria increases.

A design system that integrates *generative synthesis models* and a range of performance analysis tools and techniques is required. This paper will present a framework for building such a system, namely the *Generative Multi-Performance Design System*. The potential of the design system will then be demonstrated through a pilot project application.

2 Design Models

Human problem solving including design is done using an iterative process (Simon, 1973; Asimow, 1962; Cross, 1989; Steadman, 1979). Designs typically evolve through a cycle that involves a *synthesis* mechanism and an *analysis* mechanism which is also known as the *generate and test cycle* (Rowe, 1987).

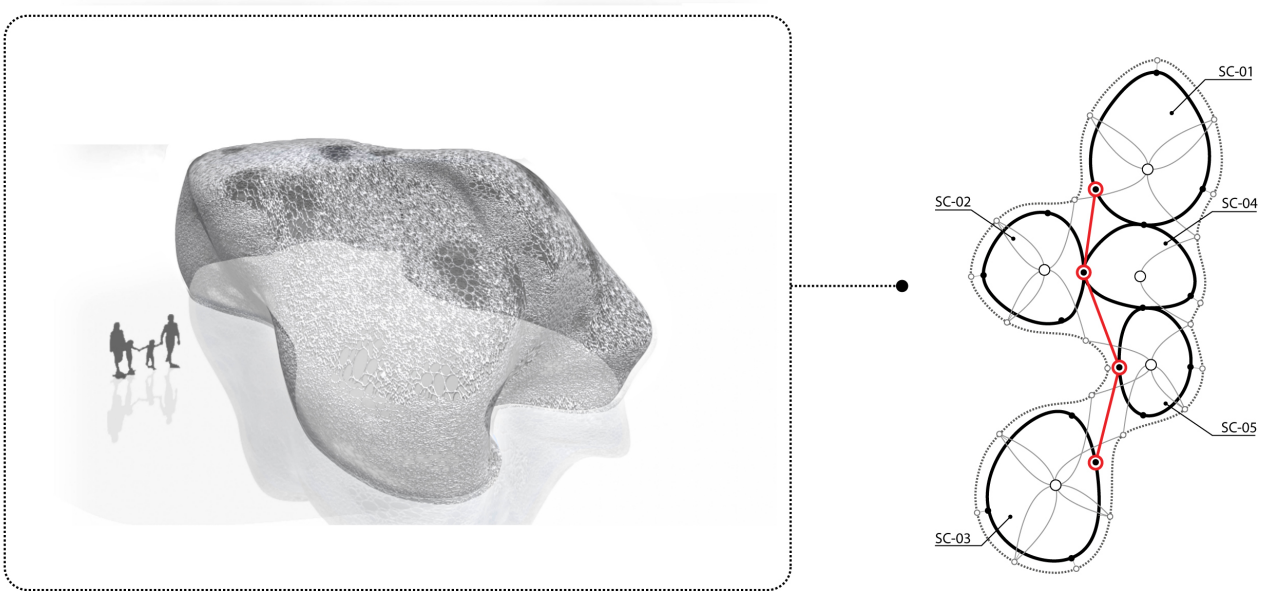
Minsky suggests the need for an additional mechanism which he terms the *progress principle* (Minsky, 1988). This is an *optimization* mechanism that guides the search rather than generate blindly all possible solutions. Progress is easy to understand when only one objective is considered, but when there are many different or even conflicting objectives progress becomes harder to define. An *evaluation* mechanism is needed to handle the decision process and to manage the tradeoffs between the different objectives. Together all four mechanisms represent phases in a *Synthesis, Analysis, Evaluation, and Optimization cycle*.

To understand these design mechanisms we need to model them. Models are an abstract description of the real world that provides an approximate representation of more complex functions of physical systems (Papalambros, 2000). In this paper we are concerned with mathematical models. These are models that can be implemented in a computer environment. We aim at building a series of mathematical models that correspond to *Synthesis, Analysis, Evaluation, and Optimization cycle*.

2.1 SYNTHESIS MODELS

A design concept can be decomposed into a set of *synthesis models* by extracting design intentions and formulating a collection of design parameters, rules or algorithms. This collection provides for a representation of the design language which in turn defines a design space. This mode of representation provides for a formalism that can be used within a computational environment to breed new designs.

Synthesis models should provide for a generative mechanism. This could be done through parametric or algorithmic descriptions. Parametric models provide for a description of the design through parameters and relationships that allow for variation. Algorithmic models give a description of the design through a set of rules and algorithms. A good example of algorithmic models is *Generative Grammars*. These include grammars like Shape Grammars, Lindenmayer Systems, and Cellular Automata.



Generative Grammars have been implemented in a variety of domains (Antonsson and Cagan, 2001). Shape Grammars have been used in the generation of buildings (Stiny and Mitchell, 1978; Downing and Fleming, 1981), and Product Design (Agarwal and Cagan, 1998). Lindenmayer Systems were originally developed to model plants (Prusinkiewicz and Lindenmayer 1991) but have been used in many other fields for design purposes including robotic design (Hornby and Pollack, 2001). CA's have also been used in a variety of domains including building design and city planning (Batty, 2005). These Synthesis Grammars are considered a form of production system (Gips and Stiny, 1980). As production systems they provide the mechanism necessary to execute design goals.

The representation of generative synthesis models should encode design knowledge. The relationship between form and performance should be embedded within the representation formalism. This provides restrictions on permitted designs and ensures that the rules discard designs that do not comply with constraints. However, since synthesis models do not include performance feedback loops, it is difficult for such models to direct the generation and navigate the design space of multi-performance design problems.

2.2 ANALYSIS MODELS

Alexander defines analysis as the measure of how well a given solution or proposed design solution fits the set of goals it is intended to meet (Alexander, 1964). An *analysis model* infers from a design solution characteristics that are relevant to a particular discipline. A design problem usually combines different disciplines, with each discipline developing one or more analysis models. Each analysis model employed should be deterministic, i.e. producing the same result at repeated calculations.

Analysis models range in their amount of required information input and their degree of accuracy output. Low order (low fidelity) models are mainly heuristic and empirical models that derive from observation and approximate data fitting rather than from physics and first principles. High order (high fidelity) models are theoretical models that are physics-based and are derived using first-principle equations like finite element analysis (FEA) and computational fluid dynamics (CFD).

In choosing a model the designer must select the best compromise between the demand for simplification and the necessity to clearly identify, describe and rate the targeted physical mechanism. A trade-off will have to be made between fidelity and analysis time.

2.3 EVALUATION MODELS

Evaluation models are in essence decision making tools. The evaluation is usually performed by means of an objective function, which consists of a figure of merit describing

FIGURE 2. THE FORMALISM OF THE DESIGN CONCEPT SHOWS FIVE SPATIAL COMPONENTS WITH INTERRELATIONS BETWEEN THEM WRAPPED BY A SKIN

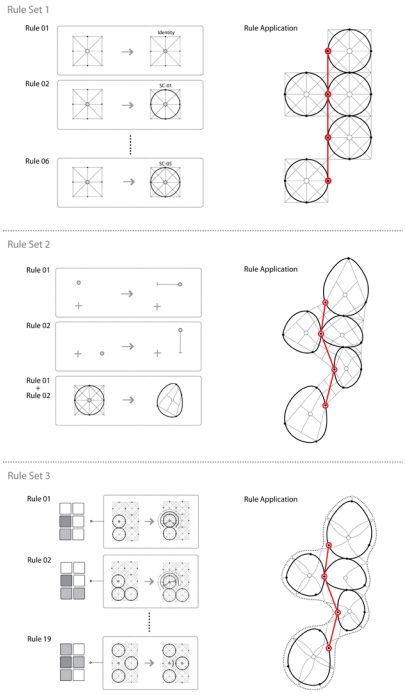


FIGURE 3. THREE RULE SETS DEFINE THE SYNTHESIS GRAMMAR.

the quality of a design solution. The formulation of the objective function is vital to the outcome of the design space search.

In single objective optimization, the search direction can be well defined and a single solution, if exists, could be found. However, in the real-world, decision-making problems are usually too complex and ill-defined, and have several possibly contradicting objectives. This implies that there is no single optimal solution but rather a whole set of possible solutions of equivalent quality (Abraham et al., 2005). In this set, each objective is optimized to the degree that if any further optimization is attempted, the other objectives will be affected as a consequence. Therefore, decisions need to be taken in the presence of trade-offs between conflicting objectives.

Addressing multiple objective problems may require techniques that are different from standard single objective optimization methods. It is generally accepted that an optimization problem should have one objective to be mathematically solvable. This objective is articulated based on the decision-maker's preferences either before or after the search.

When the preference is expressed beforehand, the designer decides how to aggregate different conflicting objectives into a single objective function before the actual search is performed (Horn, 1997). A commonly adopted approach is *scalarization* which consists of combining several objectives into one scalar cost function. There are different scalarization methods such as the weighted-sum approach and the utility function method among others.

When search is performed before decision making, the search is performed with multiple objectives at the same time. The solution space becomes partially ordered with a set of optimal trade-offs between the conflicting objectives. This set is called the Pareto optimal set.

2.4 OPTIMIZATION MODELS

Optimization Models are design space search mechanisms. Searching the design space entails finding the best solution(s) within a domain of feasible solutions. An optimization model seeks to minimize or maximize an objective function that depends on a number of continuous or discrete values by varying the values of those variables within an allowed domain. The choice of an appropriate search algorithm depends on several factors including the design synthesis model, the nature of the analysis models, the number of design variables, the existence of constraints, and the linearity of either the design variables or constraints.

Optimization techniques could be divided into general numerical optimization techniques and heuristic algorithms. Some numerical optimization techniques that handle constraints include: the simplex method, sequential quadratic programming, and the exterior and interior penalty methods among others. Numerical optimization techniques that handle unconstrained problems are generally gradient based algorithms. These include Newton's method, steepest descent, and conjugate gradient among others.

Heuristic algorithms are generally non-gradient methods like evolutionary algorithms, simulated annealing, and tabu search. The appeal of these methods stems from the fact that they do not require any gradient of the objective function in order to find the optimum.

However, no existing optimization technique is guaranteed to find the global optimum of a nonlinear, non-convex problem. Gradient-based methods find local optima with high reliability but might not escape a local optimum. Heuristic algorithms might find a good solution, but its optimality can not be guaranteed since they often tend to find a different design each time they are run.

3 The Generative Multi-Performance Design System (GMPDS)

Given a design concept with a sufficient degree of refinement our intent is to build a *Generative Multi-Performance Design System* (GMPDS) that can generate intelligent variations of the concept. This system should correspond well to the *synthesis, analysis, evaluation, and optimization cycle* discussed earlier.

Initially, through a top down approach the design concept will be decomposed iterative-

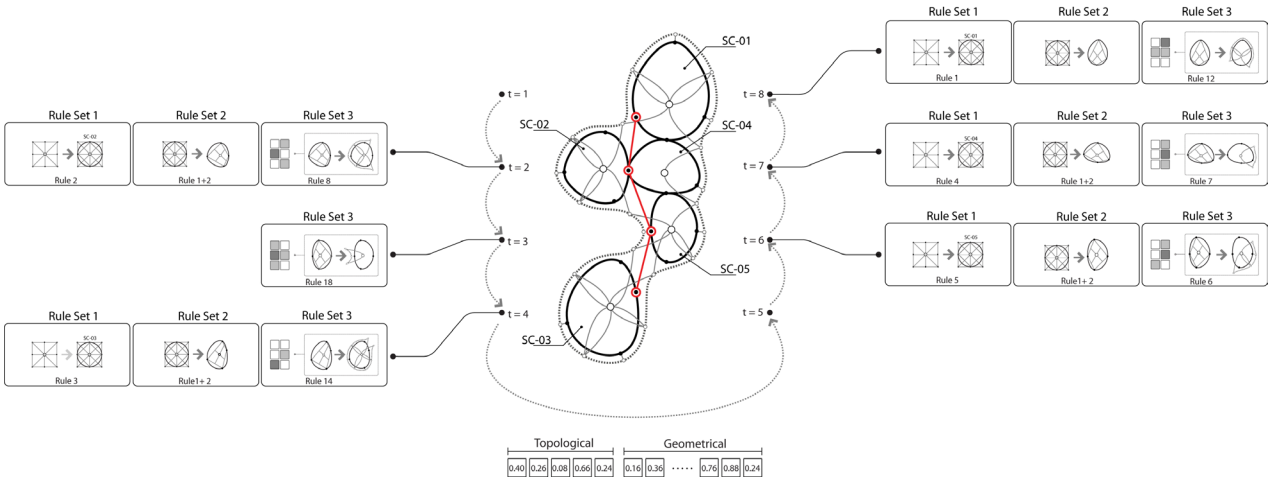


FIGURE 4. THE SEQUENCE OF APPLICATION OF THE THREE RULE SETS. STARTING WITH THE FIRST CELL AT TIME T =0 AND ENDING WITH THE LAST CELL AT TIME T =8.

ly by each discipline involved in the design. The decomposition is carried from a high-level diagram ending in manageable elementary modules. Each module will represent one of the four mathematical models. Modules for data storage can also be included. These are modules that store the parameters and constants of the design system. Modules that include the design system constraints could also be implemented.

Each module has a boundary that cuts across its links to the environment defining the module's input and output. Each module acts like a black box transforming data from one form to another. The behavior of each module contributes not only to the design discipline it is modeled after, but to the design system as a whole.

A *data flow model* of the design system is created with links that represent the interactions between the modules. These links allow the flow of information between the different modules. The design system becomes a set of interrelated modules that collectively can produce a design solution (Figure 1).

Within the proposed design system, clusters of modules can act like the synthesis, analysis, evaluation, and optimization phases discussed earlier. Together these four clusters form a high level view of the design system. Each cluster could also include cliques of nested smaller design systems.

Domain knowledge of each discipline involved in the design informs the synthesis modules to create meaningful designs and representations. The outcome of the synthesis modules is analyzed by the different discipline analysis modules to predict the properties of a particular solution. The evaluation modules then handle the multi-objective nature of the design. The optimization modules search the design space and automate the synthesis, analysis and evaluation of new solutions. The process continues until the optimization has converged and a family of acceptable solutions is found.

The design system becomes a dynamic and complex whole, interacting as a holistic structured functional unit. The system emergent properties are not detectable through the properties and behaviors of its modules, and can only be enucleated through a holistic approach. The solution found by this system is expected to be superior to the design found by solving and optimizing each discipline sequentially, since it can exploit the interactions between the disciplines.

4 Pilot Application

We now demonstrate the applicability of a *Generative Multi-Performance Design System* through a fictional case. The design concept of our fictional case includes a simple allocation of discrete but interdependent spatial components within a grid of cells located on a rectangular site with its long edge spanning north to south. The shape of the spatial components was chosen based on aesthetic preference. These spatial components are wrapped within a skin that defines their interface with the environment (Figure 2). The Spatial Components are allowed to relocate and deform to satisfy multiple performance and

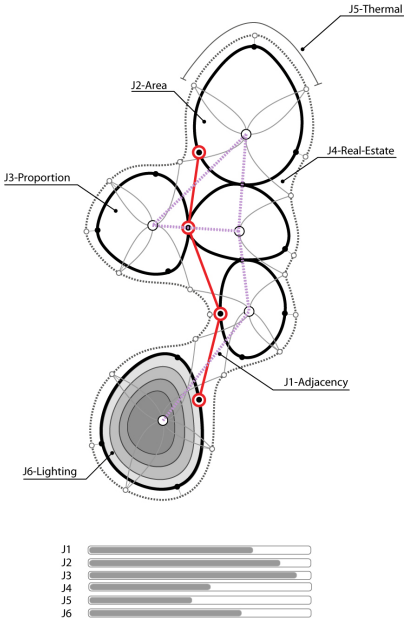


FIGURE 5. THE ANALYSIS PHASE INCLUDES SIX ANALYSIS MODULES

objective requirements such as adjacency, area, real-estate, proportion, thermal, and daylighting. This design concept entails a multi-performance space planning problem (Buffa et al, 1964).

4.1 SYNTHESIS MODULES

The synthesis phase will consist of three modules that function together as a system. These are: the *data structure module*, the *rule set module*, and the *inference engine module*. Three main data structures are implemented. These are: *Cells*, *Spatial Components*, and *Skin*. The data structure module is organized hierarchically with feedback loops between the different data structures.

A *Cell* represents the elementary unit of the space where the spatial components are to be allocated. It is defined by a set of control and boundary points and construction lines. It has knowledge about its location and its neighboring cells. It also knows the status of its occupancy and by which component. A *Spatial Component* grows in a cell, inheriting all its base geometry. When the skin is generated, the component knows which skin regions it is associated with. The *Skin* is generated from the geometric configuration of components. All three data structures also have a set of geometric attributes that include lengths, areas, and orientations.

In regards to the *rule set module*, our approach draws from shape grammars pioneered by Stiny and Gips (Stiny and Gips, 1972). A class of shape grammars that is applicable to computer implementation is set grammars (Stiny 1982). *Set grammars* consider shapes as symbolic objects and therefore do not require difficult sub-shape matching procedures. This is the approach used here. The grammar implemented is based on three fundamental design-rule sets (Figure 3). These rules draw from knowledge built into the data structures and are also organized hierarchically.

The first set of design rules deals with the allocation of spatial components. The second set of design rules deals with the deformation of the spatial components by altering the coordinates of the control-points that define the spatial components in both the x and y directions. The wrapper skin uses a set of parametric rules that can generate the skin directly. These rules compose the third rule set.

Each rule in the third rule set is applied locally to each cell and each component in a counterclockwise fashion generating the skin supports from which the skin grows. There are nineteen rules in this set that can capture all the different generated configurations. Each rule divides the skin into *Regions*. This partition of the skin is useful for many of the analysis modules implemented, since it determines each component exposed regions in an additive piecewise manner (Figure 4).

The design vector that provides the inputs to the synthesis system is divided into two types of variables, namely topological and geometrical. They are handled within the synthesis system by the first and second rule sets respectively. The third rule set builds on the output of the first and second rule sets.

The inference engine includes a *scheduler* and an *interpreter*. The scheduler applies the rule sets sequentially in an orderly manner. The interpreter searches each rule set for the matching rules of the current state and fires them when appropriate. The rules of the third rule set are context sensitive and function like a simple two dimensional cellular automata that analyses each neighbor's occupancy and decides which rule to apply. All three synthesis modules were implemented in the CATIA VBA environment.

4.2 ANALYSIS MODULES

In the proposed experiment, the design concept will be broken down into multiple single-disciplinary analysis modules in order to evaluate how well it performs from the point of view of each discipline separately. These modules include: an *Adjacency module*, an *Area module*, a *Real-Estate module*, a *Proportion module*, a *Thermal module*, and a *Lighting module* (Figure 5).

Since we are working at the design concept stage, the level of detail of the overall design constitutes a simplification of reality. Any rigorous analysis may go beyond the scope and the precision of the overall design description. Therefore, the models we will use for the different discipline modules will be based on heuristics or simplified representations to

test the feasibility of design solutions. The modules were implemented in VB Scripts and VBA Scripts built in Excel.

A functional rationale determines the adjacency requirements that the spatial components have to comply with. These requirements are treated and quantified as a set of "bond forces" that tie together all components, pair-wise. In the *Adjacency module*, the actual design—in terms of the location of the spatial components—is examined and rated against these requirements.

The *Area module* compares the areas of the spatial components generated by the synthesis system with the areas prescribed by the architectural area program. It favours solutions with a high compliance with the program and flags solutions that show a worse compliance.

The *Real-estate module* compares the floor plan's net area to its gross area. It aims at minimizing the space between the spatial components. These are areas that are allocated to circulation, but have a lower real estate value.

The shape of the spatial components is determined by the location of the control points. A spatial component may have a multitude of possible shapes due to the location of the control points. The *Proportion module* aims at promoting more skewed or slanted shapes that produce aesthetically more appealing layouts, while keeping elongation and distortions within acceptable limits. This module filters any regular or fairly irregular shapes and discourages highly irregular forms.

Assuming that we are building in a cold climate, it is important to provide a design that minimizes thermal losses and favours solar gain. A simple *Thermal module* was devised to measure the energy balance. A few simple assumptions were adopted, due to the minimalism of the design model. Yet these simple assumptions proved capable of capturing the fundamental relationship between the shape of a building and its environmental performance.

The day lighting performance is assessed by adopting a simple geometric model. The *Lighting module* measures, for each exposed region in a spatial component, the fraction of area that is exposed to sunlight, and multiplies it by a coefficient that depends on orientation. A number of physical simplifications were also adopted.

There are two main constraint modules implemented. The first is a *topological constraint module*, and the second is *geometric constraint module*. Both modules act on the design vectors and not on the design solution generated from the synthesis system.

In the topological constraints module the spatial components locations are tested against the adjacency requirements specified by the designer. The topological constraints ensure that the number of violated strong-bond relationships does not exceed a pre-set threshold. The geometric module tests the locations of control points and prevents any excessive distortion of the grid that might create non-convex spatial components.

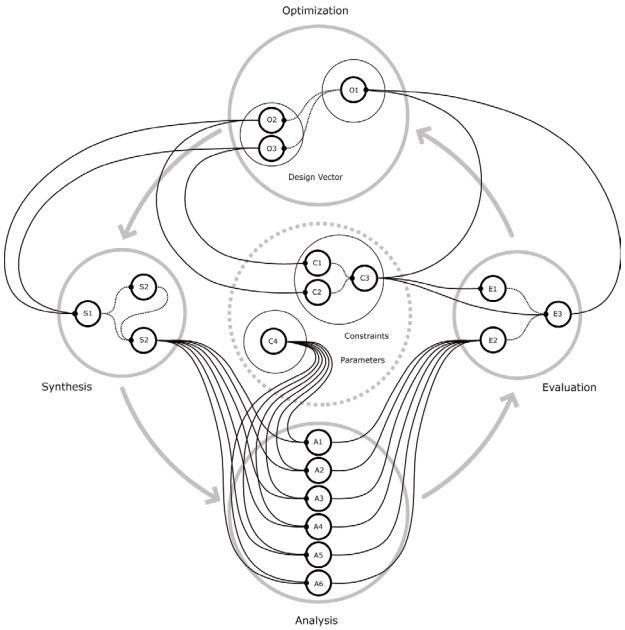
In addition to the constraints modules, a *constants and data storage module* was also implemented. This module contains all the constants and parameters used by the different modules such as location, climate, and area program among others.

4.3 EVALUATION MODULES

There are three evaluation modules implemented. The first is a *flow control module* that evaluates if the design vector violates the constraint modules in the analysis cluster. It acts as a switch directing the data flow to one of the other two evaluation modules. The other two modules are the *feasible design* and *infeasible design modules*. The infeasible design module is triggered by the flow control module if the constraints are severely violated. If the constraints are not violated the feasible design module is triggered.

Although the constraints are handled by the optimization modules, the flow control module is important from a design-process management point of view. If the design vector is infeasible the flow control module would bypass the synthesis and analysis modules saving extensive computational time. The infeasible design module simply signals the violation to the optimization modules and ranks the design solution in proportion to the number of violated constraints.

The feasible design module on the other hand triggers the synthesis and analysis mod-



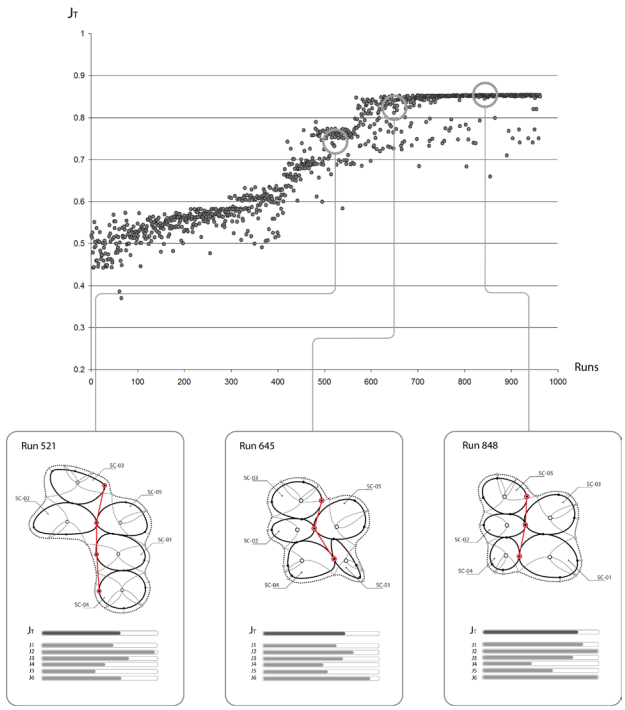


FIGURE 6. THE GMPDS OF THE EXPERIMENT

FIGURE 7. THE EVOLUTION OF SOLUTIONS. SOLUTIONS IN THE FINAL RUNS TEND TO BE MORE COMPACT IN THEIR SHAPE.

ules. All the ratings (J_{area}, J_{circ}, etc.) of the disciplinary performances that originate from the analysis modules converge into the feasible design evaluation module, where they are aggregated to generate an overall evaluation of the design, according to the standard scalarization approach.

4.4 OPTIMIZATION MODULES

The optimization modules consist of two groups of modules. The first contains the optimization algorithm, and the second handles the design vectors. The design vector contains the design variables. It guides the design solutions by informing the synthesis system, like the DNA of an organism. Two major categories of design variables have been considered in our experiment and are implemented in two different modules these are: the *topological variables module* and the *geometric variables module*.

The topological variables module defines the cell location of each spatial component. Instead of creating constraints that prohibit the allocation of two different spatial components in the same cell, this check is performed implicitly within this module. This guarantees that no two spatial components are placed in the same cell.

The geometric variables module on the other hand guarantees that the control points in a row or a column remain distributed in an organized manner in order to minimize singularities while generating the cells.

Due to the nature of the design space, the search algorithm implemented should not be limited by restrictions of continuity or existence of derivatives. Therefore a Genetic Algorithm (GA) was implemented. The evolution starts from a population of randomly generated design solutions, in addition to a few seeded acceptable design solutions to guide the evolution. The evolution happens in generations. For each generation, the fitness of every design solution in the population is evaluated. In our experiment this fitness is represented by the multi-disciplinary performance *J* formulated in the evaluation modules.

Constraints are implemented using penalty functions. If a solution does not comply with the constraints in the system a penalty is added to the fitness of the design solution according to the degree of violation.

Due to the existence of multi-objectives the aim is not to produce a global optimum solution, but rather to direct the evolutionary process to produce populations of good

solutions. These solutions would be used to study the tradeoffs between the different objectives.

4.5 BREEDING DESIGN SOLUTIONS USING GMPDS

As opposed to the decomposition of the design concept into modules, the process of assembling the *Generative Multi-Performance Design System* is a bottom up approach. After all the modules discussed earlier have been built and their validity verified, the *data flow model of the design system* is implemented (Figure 6).

The three synthesis modules namely: the rule set module, the data structure module, and the inference engine work together as a unit forming a synthesis subsystem. This synthesis system receives the decoded design variables from the design vectors module in the optimization cluster and outputs a design solution (phenotype) that can be analyzed.

Within the analysis cluster, each of the six analysis modules receives from the synthesis system the relevant data. Each module then provides a measurement of the design performance for that module.

The evaluation modules controls the flow of data, by making sure that those designs that do not comply with the constraints are not sent to the synthesis system to be further developed into a full design solution (phenotype). If the constraints are not violated, the performance measurements generated by the different analysis modules are then aggregated into an objective function that acts as a figure of merit.

The GA in the optimization cluster then evaluates the fitness of the design solutions in the population. Several solutions are chosen based on their fitness and undergo genetic transformations to form a new population. The GA runs until satisfactory fitness levels are reached. The model was implemented in the Model Center environment.

The GMPDS demonstrated promising results (Figure 7). It managed to generate interesting solutions to our multi-performance space planning problem. It managed to improve the overall performance of the design solutions beyond our initial seeded solutions. As expected, due to the site restrictions the *spatial components* were forced to span from north to south. However, the design solutions in the final populations tended to be compact in their shape. This implies that the design drivers were mostly the adjacency and real-estate modules which were highly weighted in our objective function. As had been expected, both the lighting and thermal modules were in clear conflict with each other. Since they were both given identical weights, they tended to balance out each other. The program and proportion modules although being satisfied, did not seem to have an obvious effect on driving the design solutions.

5 Conclusion

In this paper a framework for building a *Generative Multi-Performance Design System* was introduced and its advantages investigated. The GMPDS should be considered domain independent. Its potential was demonstrated through a pilot project in which a multi-performance space planning problem was considered. The design system was able to successfully search the design space and find interesting and unexpected solutions. Since performance criteria were a driving force in the generation of design solutions, the system provided insight into the main design drivers. Due to the multi-objective nature of the system, the aim was not to produce a single optimal design solution but instead to generate populations of valuable solutions.

The solutions found by the system are expected to be superior to the design found by solving and optimizing each discipline sequentially, since the proposed design system can exploit the interactions between the disciplines and can recognize the potential benefits of mathematically predicting and analyzing the integrated behavior of design solutions. The argument is that an integrated design system will yield higher quality designs with improved performance.

Based on our findings it is concluded that *Generative Multi-Performance Design Systems* (GMPDS) are capable of producing excellent results. They build on the strengths inherent in both generative synthesis systems and multi-performance analysis and optimization.

Future work on the pilot project would include generating Pareto sets for the multi-objective formulation in the objective function to remove biases towards each criterion. This will help investigate tradeoffs between different performance requirements and analyze sensitivity and design drivers. In addition, we would like to develop the synthesis system to include 3D models of our generated solution and possibly add more detailed analysis modules with higher fidelity.

6 References

Abraham, A., Jain, L. and Goldberg, R. (Eds.) (2005). *Evolutionary multiobjective optimization: theoretical advances and applications*. New York : Springer Science.

Agarwal, M. and Cagan J, (1998) A blend of different tastes: the language of coffeemakers. *Environment and Planning B: Planning and Design* 25(2) 205-226.

Alexander, C. (1964) *Notes on the Synthesis of Form.* Cambridge, MA, Harvard University Press.

Antonsson E.K. and Cagan J. (2001) *Formal Engineering Design Synthesis.* Cambridge, Cambridge University Press.

Asimow, M. (1962) *Introduction to design*. Englewood Cliffs, N.J., Prentice-Hall.

Batty, M. (2005) *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*. Cambridge, MA, MIT Press.

Buffa, E.S., Armour, G.S. and Vollman, T.E. (1964). Allocating facilities with CRAFT, *Harvard Business Review* 42(2): 136-140.

Cross, N. (1989) *Engineering Design Methods*. Chichester, John Wiley & Sons Ltd.

Downing, F. and Flemming, U. (1981) The bungalows of Buffalo, *Environment and Planning B* 8(3) 269–293.

Gips, J. and Stiny, G. (1980) Production systems and grammars: a uniform characterization. *Environment and Planning B* 7(4) 399 – 408.

Horn, J. (1997) Multicriterion decision making, in: T. Back, D. Forgel, Z. Michalewicz (Eds.), *Handbook of Evolutionary Computation*, Institute of Physics. Bristol, UK.

Hornby, G. S. and Pollack, J. B. (2001) Body-Brain Coevolution Using L-systems as a Generative Encoding. *Genetic and Evolutionary Computation Conference* (GECCO) 2001.

Minsky, M. L. (1988) *The Society of Mind*. New York, NY, Simon and Schuster.

Papalambros, P. Y. (2000) *Principles of optimal design: modeling and computation*. Cambridge, Cambridge University Press.

Prusinkiewics, P. and Lindenmayer, A. (1991) *The Algorithmic Beauty of Plants*. Springer-Verlag.

Rowe, P. G. (1987) *Design Thinking*. Cambridge, MA, MIT Press.

Simon, H. A. (1973) *The Sciences of the Artificial*. Cambridge, MA, MIT Press.

Steadman, P. (1979) *The Evolution of Designs*. Cambridge, Cambridge University Press.

Stiny, G. (1982) Spatial relations and grammars. *Environment and Planning B* 9, 313–314.

Stiny, G. and Gips, J. (1972) Shape Grammars and the Generative Specification of Painting and Sculpture. C V Freiman (Ed) *Information Processing* 7. Amsterdam, North–Holland, 1460–1465.

Stiny, G. and Mitchell, W. J. (1978) The Palladian Grammar, *Environment and Planning B* 5: 5-18.