Project Discover: An Application of Generative Design for Architectural Space Planning

Danil Nagy, Damon Lau, John Locke, Jim Stoddart,

Lorenzo Villaggi, Ray Wang, Dale Zhao and David Benjamin

The Living, an Autodesk Studio
New York, NY USA
life@thelivingnewyork.com

ABSTRACT
This paper describes a flexible workflow for generative design applied to architectural space planning. We describe this workflow through an application for the design of a new office space. First, we describe a computational design model that can create a variety of office layouts including locating all necessary programs and people using a small set of input parameters. We then describe six unique objectives that evaluate each layout based on architectural performance as well as worker-specific preferences. Finally, we show the use of a multi-objective genetic algorithm (MOGA) to search through the high-dimensional space of all possible designs, and describe several visualization tools that can help a designer to navigate through this design space and choose good designs. We conclude by discussing the future of such computational workflows in design and architecture. Our hope is that they go beyond basic automation to create an expanded role for the human designer and a more dynamic and collaborative interaction between computer design software and human designers in the future.

Author Keywords
Parametric modeling, simulation, genetic algorithms, multi-objective optimization, evolutionary design, generative design, architecture

ACM Classification Keywords
I.6.5 SIMULATION AND MODELING - Model Development

1 INTRODUCTION
Computers and computer-aided design (CAD) software have had a dramatic impact on architectural practice since the emergence of computers in academia in the 1950s, and especially since the introduction of personal computing in the 1980s. Although early researchers envisioned a wide-ranging future interaction between computers and human designers [10], the first computer tools to be widely adopted by architectural designers were computerized versions of traditional drafting and rendering tools. While they allowed designers to produce content much faster than with traditional methods, they did not fundamentally change the process of design.

1.1 Parametric design
In the past decade, a new type of design software has emerged which is fundamentally changing the way designers use computers to develop and refine their designs. Known as *parametric design software*, these tools allow the designer to not only define a final geometric solution, but to describe the entire system behind how a design is generated. Within this larger system description, the designer can expose specific parameters, or values that drive different variations of the design.

Although such a model takes more work initially to describe, it offers the designer many advantages. First, the parametric approach makes it easy to create variations and custom adaptations of a design. Instead of manually creating multiple versions for different applications, the designer can expose the critical parameters that drive different variations and automatically generate different versions by changing those parameters. Second, a well-structured parametric model is more adaptable to change in the future. Since it is defined by a series of operations, the design can be easily adapted to changing conditions instead of rebuilding the model from scratch each time.

Most importantly, the parametric approach allows the designer to think through design solutions in a deeper and more dynamic way than possible with traditional methods. In a traditional approach, the designer studies the design problem, internalizes all of its constraints and objectives, and then uses their skill and experience to craft a single design solution, or a handful at most. With the parametric approach, the constraints and goals of the design problem can be directly embedded within the parametric model, which can then be used to automatically generate a variety of solutions. Instead of designing a single solution, the designer can now think of designing a multi-dimensional 'space' of design. Each dimension of this design space represents one of the critical parameters exposed by the parametric model, and each individual design variation can be found somewhere within this hyper-dimensional space.

1.2 Beyond parametric
While the parametric approach has broadened the possibilities of design and pushed the boundaries of human-computer

interaction in the design process, the exploration of the design space is still limited by the abilities of the human designer. Although some parameters may be set by the constraints found explicitly in the design problem, for the most part the human designer must investigate different design options by manually varying individual parameters and evaluating each option using their own criteria and intuition in a way not much different than with traditional design methods.

The concept of *generative design*, as described in this paper, addresses this limitation by tasking a computer with exploring the design space semi-autonomously, and then reporting back to the designer which options it considers promising for further analysis. Because a computer can process information much quicker than a human, such a system allows a much deeper exploration of complex design spaces. Traditionally, such an approach has been used to optimize a given model to achieve maximum possible performance based on concrete objectives [8]. With a model of sufficient complexity, however, a generative design system can also be used to reveal interesting parts of the design space and discover novel design solutions that would otherwise be hidden to the human designer.

To take advantage of the possibilities of generative design, the basic parametric model must be extended in two ways. First, the model must include concrete metrics by which each design option can be evaluated. Since the computer does not have any inherent intuition about design, the human designer must explicitly describe to the computer how to determine which designs perform better than others. Second, the model needs to be connected to a search algorithm that can control the input parameters of the model, get feedback from the metrics, and intelligently tune the parameters to find high performing designs while also exploring the full possibilities of the design space. One of the most promising of these algorithms is the multi-objective genetic algorithm (MOGA), which uses principles of evolution to create sequential generations of designs and evolve them to contain higher performing designs over time [9].

The remainder of this paper describes our development of a custom workflow for generative design specifically geared towards architectural space planning, and our application of this workflow to the design of a new office space.

2 RELATED WORK

The application of multi-objective optimization towards solving complex mechanical design problems is well-known in the field of engineering. Marler and Arora [8] provide a good overview of various applications. However, being constrained to the goals of engineering problems, these applications are limited to using only structural performance as optimization criteria.

Liggett [7] provides a thorough historical overview of automated methods for space planning in architecture, including the use of genetic search algorithms. Derix [2], Keough and Benjamin [6], Chronis et al. [1] and Gerber et al. [3] have applied similar optimization methods to a variety of architectural problems. However, their optimization criteria are similarly constrained to well-known and easily simulated physical objectives such as structural and environmental performance. In contrast, we propose a more flexible workflow that can accommodate a diversity of optimization criteria, including those dealing directly with how space is used and experienced at the occupant level

The quantification of spatial experience has also been explored by a variety of authors. Hillier, et al. [4] proposed a variety of analytical tools for studying spatial configurations which they called ‘space syntax’. Peponis, et al. [11] extend this work by proposing a universal method for understanding plan topology through linear representation. Turner, et al. [12] propose a view-based ray tracing technique for understanding and analyzing spatial configurations. While the proposed methods can help the designer derive quantitative data about their designs, they are only offered as tools to aid a traditional design process. In contrast, we extend these methods and show how they can be used as measures of spatial performance to guide an automated optimization process.

3 METHODOLOGY

Our proposed workflow of generative design for architecture is organized into four steps: (1) the design of a geometric model which can create many design variations, (2) the design of a series of performance metrics which can be used to measure the performance of a single design, (3) the exploration of the model’s design space through a MOGA, and (4) the investigation of the resulting design data through statistical analysis. Furthermore, we propose this method as only one component within a broader design process. Thus, there are several steps that must be taken both ‘before generative design’ in order to establish a design concept to drive the geometric model and collect necessary data for the performance metrics. Similarly, there are a variety of steps that must be taken ‘after generative design’ in order to achieve other criteria and develop the selected design solution to the level of a final constructible design.

3.1 Before generative design

As with any architectural design project, the process begins by studying the design problem, understanding its goals and constraints, and formulating a vision and concept for the design. The vision of the project was to create a dynamic and highly functional new office space for Autodesk in Toronto. Some of the constraints included:

1. The outline of the three floors of an existing new building where the office would be located
2. The programmatic requirements, including specific numbers of shared amenities such as meeting rooms
3. Occupation by up to 300 workers
4. Diversity of different departments, project teams, and workstyles that the office needed to accommodate

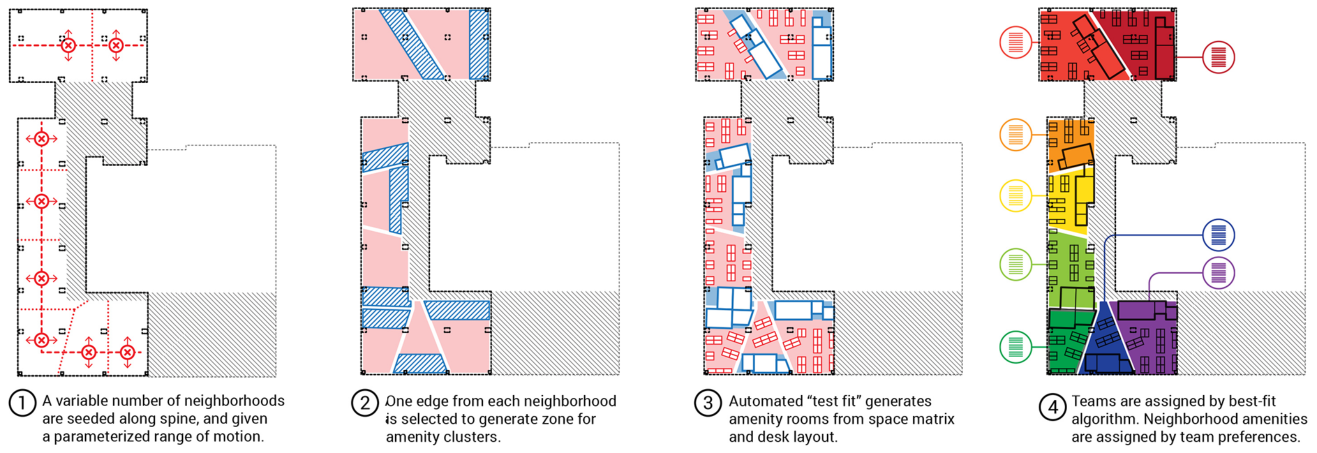
1 A variable number of neighborhoods are seeded along spine, and given a parameterized range of motion.
2 One edge from each neighborhood is selected to generate zone for amenity clusters.
3 Automated "test fit" generates amenity rooms from space matrix and desk layout.
4 Teams are assigned by best-fit algorithm. Neighborhood amenities are assigned by team preferences.

Figure 1. Description of specification of geometric model

Based on the vision of the project and these constraints (the goals of the project were established in a subsequent step), we developed an architectural concept around breaking up the floorplan into a series of individual 'neighborhoods'. In this concept each neighborhood is a work-area for an individual department or project team. The neighborhoods are divided by shared amenity spaces, which are contained within standalone rooms. These rooms create visual variation within the office space as a whole, while providing a degree of privacy and uniqueness to each neighborhood.

Once this basic concept was established, the design problem became the arrangement of neighborhoods within the building floorplan, the location of shared amenity spaces, and the assignment and placement of teams and individual workers in the neighborhoods. In architecture, this type of problem is known as *space planning*, and deals with the optimal arrangement of programs and spaces within a fixed plan. Because there are so many possible variations, this type of problem is traditionally difficult to solve for a human designer, and typically relies heavily on intuition and rules of thumb, along with iterative design and testing of a large variety of solutions before finally choosing the best one. Due to the complexity of this problem, it was actually the subject of one of the first applications of computing to architectural design [5]. For us it was the perfect problem to test the possibilities of the generative design process previously described.

Besides exploring many design options, another advantage of the generative design approach is that we can evaluate designs at a much higher level of detail than possible with traditional approaches, including evaluating some aspects of the design which are often ignored or abstracted in typical space planning projects. In this case we wanted to judge each design not only on global architectural goals such as maximizing the amount of light in the space, but also on local goals having to do with the individual preferences of each of the office's future occupants.

To get information about these preferences we distributed surveys to all individuals and teams in the office, asking their preferences in terms of which amenities they want to be close to, which other teams or individuals they often work with, and the office conditions they prefer. Based on this information, we were ready to construct the generative design model that could generate unique design solutions and evaluate each one based on specific performance metrics.

3.2 Geometric model

The first step was to create a geometric model that could define a set of neighborhoods within the two main floors of the office building, position shared amenity zones between neighborhoods, and then locate specific programs within the amenity zones and individual workers within the neighborhoods. To create each individual design, our geometric model applies the following algorithm (see Figure 1):

1. Locate a seed point for every neighborhood
2. Draw neighborhood boundaries based on edges equidistant from the neighborhood seeds (similar to a voronoi diagram)
3. For each neighborhood, choose one of the edges along which to place a shared amenity zone
4. Place shared programs within amenity zones based on a greedy fill algorithm
5. Assign teams to neighborhoods, also based on a greedy fill algorithm
6. Assign people to specific desks in neighborhood based on list order.

To establish the neighborhood seeds, a linear spine is drawn over the plan and the seeds are distributed evenly along this spine. Then, each seed's exact location is refined by two individual parameters – the first defines the distance to move along the spine from the initial point, and the second defines the distance to move away from the spine in the perpendicular direction. A third unique parameter chooses the edge along which to place the amenity zone by specifying its normalized distance along the neighborhood boundary. The placement of individual amenity programs, teams, and individuals is not parameterized, but is instead directly determined according to the geometry of the neighborhood boundaries.

With 15 neighborhoods controlled by 3 unique parameters each, the model is completely described by 45 unique parameters. Currently, there are no theories or rules for how many individual parameters a model should contain to ensure that a robust search of the design space is both feasible and complex enough to create a wide variety of design options. In general, the current best practice is to make this number as small as possible, while ensuring that each critical aspect of the design is controlled by a *unique, continuous variable*. The uniqueness of each parameter is important so that the algorithm can directly control each aspect of the design independently while searching for the best combinations. The continuity of each parameter is important because the algorithm should be able to fine-tune the parameter settings by predicting future results based on past experiences. If each setting of a parameter yields completely different results, it will be far more difficult for the algorithm to search through the design space.

Finally, in order to take advantage of learning within the automated search process, the entire model needs to be completely deterministic, relying only on the input parameters exposed to the algorithm to generate each design. No noise or random parameters should be utilized in the geometric model.

3.3 Design metrics

To allow the search algorithm to automatically measure the performance of each design generated, we also defined a set of unique goals, or metrics, which rate the relative performance of each design along a set of criteria. These metrics form the set of output values that the search algorithm can use to evaluate how well each design option performs, and to guide its search of the design space toward discovering higher performing designs.

One apparent limitation of the generative design process is that all performance criteria for a given design system must be exposed to the search algorithm as a numeric quantity. Thus, any performance metric that we want the algorithm to consider must be both quantifiable and computable in a reliable and efficient way for all solutions within the design space.

In engineering applications where similar optimization workflows have been explored for a number of years, the metrics are relatively straight forward. For example, the strength of a structural component is easy to compute using standard finite element analysis (FEA) software. An architectural design problem, however, often has many competing and complex goals, some of which are difficult if not impossible to quantify such as beauty, fairness, quality of space, elegance, and novelty. To deal with this potential difficulty, we divide the set of all possible architectural performance metrics into three groups:

- Those that can be easily quantified and calculated using existing tools (e.g. daylight analysis)
- Those that can theoretically be quantified but cannot be computed using existing tools, for which new computation tools must be developed (e.g. employee work style preference and activity hotspots)
- Those that cannot be quantified and must be addressed through other means outside of generative design (e.g. beauty)

While this classification addresses the current limitations of the generative design workflow, the conclusion of this paper outlines some ideas for future research that suggests machine learning as a way to quantify and evaluate goals that are challenging to compute using direct calculation. In our case, our analysis of the project goals along with discussions with the managers and individual workers yielded six discrete design metrics to evaluate each design (see Figure 2):

1. *Adjacency preference*, which measures the travel distance from each employee to their preferred neighbors and amenities
2. *Work style preference*, which measures the suitability of an assigned neighborhood's daylight and distraction measurements to the assigned team's surveyed preferences
3. *Buzz*, which measures the amount and distribution of high-activity zones
4. *Productivity,* which measures concentration levels at individual desks based on sight lines to other desks and other noise sources
5. *Daylight,* which measures the total amount of natural daylight entering the space throughout the year.
6. *Views to outside,* which measures the ratio of workspaces with an unobstructed view to the exterior glass façade

One of these – daylight – is well understood and can be calculated using existing analysis tools. The other five were either novel or highly specific to our design goals. For these we developed our own custom analysis tools which we built directly into the generative design model.

Each new design project potentially brings with it a unique set of goals and performance requirements, which will never be fully described in any given design software. Thus, part of the responsibility of the designer in the generative design workflow is to be able to use computational tools such as parametric modeling and custom scripting to describe their

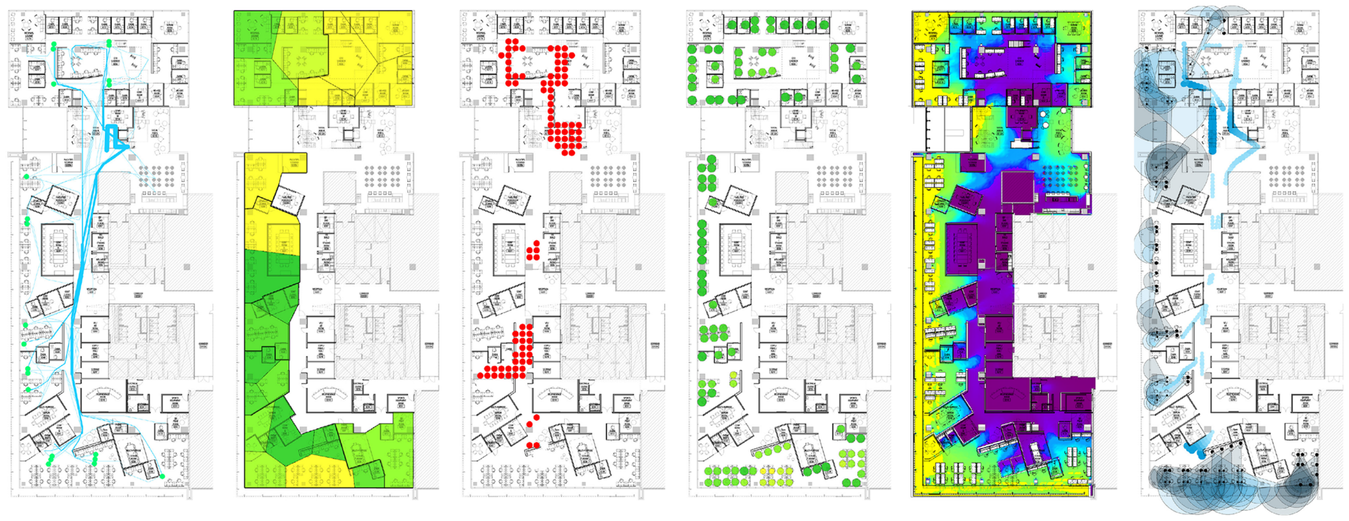

Figure 2. Design metrics (from left to right: adjacency preference, work style preference, buzz, productivity, daylight, and views to outside)

unique design goals to the computer. Although this sometimes makes the design task more difficult, it also has the potential to expand the role of the human designer while opening up new opportunities for design though enhanced human-computer interaction.

Along with the geometric model, the design metrics constitute the second half of the full *generative design model*. This model is a closed system that (1) takes in a discrete set of input parameters, (2) creates a unique design solution based on those parameters, (3) evaluates the design along a set of unique metrics, and (4) outputs those metrics as a set of discrete values. When this system is connected to a search algorithm, it can be automatically explored for good design solutions. However, although the algorithm can explore many more designs than possible through traditional manual means, it can only evaluate them based on the specified metrics output by the model. Thus it is crucial that the chosen metrics sufficiently capture the priorities of the design problem, and accurately describe the relative performance of each design according to those metrics.

3.4 Design evolution

Once we have defined the generative design model, we can use a search algorithm to automatically explore the space of possible designs and discover novel and high performing design options. A search algorithm is a subset of a general optimization algorithm, which is concerned with discovering optimal settings of input parameters of a function which maximizes the value of one or more outputs. Although many search algorithms exist, the one of particular interest to us is the multi-objective genetic algorithm (MOGA).

This algorithm generates designs in groups called generations. The first generation is composed of a set of initial designs either randomly or evenly sampled from the design space. Subsequent generations are then produced by either directly taking high performing designs from the previous generation (a process called *elitism*), or randomly mixing the parameters of two high performing designs to create a single new design (a process called *cross-breeding*). Each new design's input parameters may also be slightly modified before it enters the population (a process called *mutation*). This process is then repeated for multiple generations, either until the target number of generations is reached, or performance fails to improve for a certain number of generations. In this way, a MOGA uses concepts found in natural evolution to generate new designs based on the input parameters (genome) of previous high performing designs, thus gradually promoting the best options (survival of the fittest) and 'evolving' higher performing designs over time.

This type of algorithm has many advantages in the context of generative design. As the name implies, the MOGA can optimize designs along any number of output metrics. Furthermore, the user does not need to prioritize or weight the individual metrics beforehand. This is because the MOGA determines relative performance based on the idea of *dominance* rather than the absolute difference in metric values. A design is considered better performing than another if it dominates or performs better in one or more of the metrics. Thus the algorithm will continue to produce designs that are dominant in as many of the metrics as possible, and the user can later decide how to prioritize the metrics.

Another advantage of the MOGA is that it works stochastically by sampling designs from the design space, and trying to learn optimal configurations of the input parameters through experimentation. Other optimization algorithms such as gradient descent rely on computing gradients for each objective with respect to each input parameter. This is not possible with most parametric design models, which are

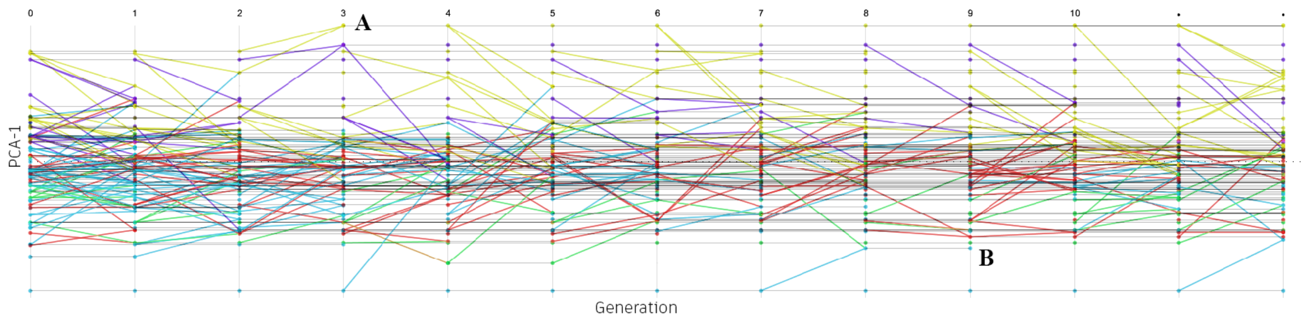

Figure 3. Time plot showing lineages of designs through generations (color indicates design cluster)

defined by a large number of geometric functions, none of which can be easily differentiated. Thus, such model can only by optimized through a stochastic experimental process.

Finally, genetic algorithms have also been shown to be exceptionally good at finding the overall best performing designs within a design space (the global optimum) while avoiding locally high-performing areas that may not be the best overall. By recombining high-performing designs from different areas of the design space, and slightly mutating designs over time, genetic algorithms can avoid local optimums more effectively than simpler, more deterministic algorithms such as gradient descent.

As with any optimization algorithm, the MOGA has hyper-parameters that need to be set before beginning the search process. These hyper-parameters have a significant impact on how the algorithm behaves and thus are an important aspect of generating good results. However, these settings also depend on the nature of the problem, so their tuning is often a product of heuristics and previous experience. The MOGA hyper-parameters include:

- The sampling method or the starting population
- The size of the starting and subsequent populations
- The termination criteria of the process (run for a set number of generations, or continue until no new better designs are found for a number of generations?)
- Cross-over rate, which dictates how many of a generation's designs are created by combining two designs from the previous generations
- Mutation rate, which dictates the rate at which a design's parameters are slightly modified before entering the next generation

In our case, we used generations of 100 designs each and ran the process for 100 generations creating 10,000 designs. The starting population of 100 designs was generated by randomly sampling from the design space. Through experimentation we settled on settings of 95% for cross-over, and 0.2% for mutation. The entire process ran over 5 days on a single MacBook Pro with a 2.60GHz Intel Core i7 processor and 16 GB RAM.

3.5 Data analysis

This process generated a data set containing 10,000 designs, including the input values for each design and its score along the six metrics. One approach at this stage would be to filter the dataset by the metric scores and directly select a few high-performing designs for further analysis. However, depending on the complexity of the design problem such a selection can be challenging for a number of reasons.

First, the various metrics might be directly competing with each other, which means that there is actually no single best design but a range of equally high performing designs along the trade-off between competing metrics. For example, when designing an industrial component there is typically a trade-off between the part's weight and its strength. In this case, unless there is a specific weight or strength target, it would be difficult to select a single 'best' design without first understanding how this trade-off works.

Second, as previously mentioned, the hyper-parameters of the MOGA have a significant effect on how the search works, and proper tuning of these settings depends on the particularities of each generative design model (including how many and what type of input parameters and output metrics are used). Thus, it is rarely enough to run only a single search process, and it is helpful if the results of every search are studied in depth to determine how the hyper-parameters may be tuned for future runs.

Finally, one of the advantages of a learning-based process such as MOGA is that it not only finds high-performing designs but also performs the search in a structured, semi-intelligent manner. By investigating the search process itself, more can be learned about the nature of the problem as a whole. In order to investigate this process and gain a deeper understanding of the design space, we developed a series of data analysis tools to aid the designer in exploring the dataset of designs generated by the MOGA.

Inheritance analysis

In addition to the input and output data for each design, the MOGA also outputs a history of how these designs were generated. Figure 3 shows a plot of this data, with each point representing a design, and each column of points representing a generation of designs. Two colored lines entering a point from the left indicates that the design was formed through cross-breeding of those two designs. A thin black line indicates that the design was carried over directly into the next generation.

In this plot you can see an instance where a newly formed design is high performing and thus is consistently carried over into future generations (A), as well as a case where a new design gets carried over one generation but then dies out, likely due to the fact that it was not as high-performing as others in its generation (B). Studying such plots helps us understand how the algorithm explored the design space, how dominant design lineages are formed, and helps locate potential blind spots in the design space missed by the algorithm.

Input space analysis and clustering

To analyze how the sampled designs are distributed within the design space, we can use principal component analysis (PCA) to transform the 45-dimensional input space into a new 45-dimensional space where the dimensions are now ordered according to the extent to which they describe the variance in the data. Then we can use the first two PCA components to create the best-possible two-dimensional projection of the high-dimensional design space and see how the sampled designs are organized within that space.

To further study the distribution of designs in the design space we can cluster them based on Euclidean distance in the full 45-dimensional design space using the K-means algorithm (see Figure 4). Intuitively, this gives a representation of different design typologies or strategies that share similar input parameters. Once we have assigned the clusters to each design we can study how these design typologies relate to performance in the output metrics. For example we can see if certain design types perform better in some metrics than others. Such tools can help us understand the design problem in general and reveal potential design strategies, rather than simply picking the single best design.

Metric space analysis

Once we have understood the distribution of designs in the input space, we can study how the designs perform along the six performance metrics. Since there are usually less output metrics than input parameters, the space of outputs is not typically as high-dimensional as the input design space. Nevertheless, if there are more than 3 or 4 metrics it can be difficult to represent the results on a single plot. Our typical approach is to do a pairwise plot of all the output metrics to find combinations of metrics that have an interesting relationship or a clear trade-off. We can then study the tradeoffs in greater detail by plotting them against each other on a scatter plot (see Figure 5).

Once we have studied the performance of the whole set of designs, we can select a subset for further manual analysis. As a baseline the MOGA will provide us with a set of designs which are statistically dominant called the Pareto designs. To narrow it down further we can look for designs that occur at different points along the trade-offs, which can help us to see the effect of those trade-offs on the design solution. We can also use the cluster information generated earlier to identify cases where similar performance was achieved by different typologies of designs.

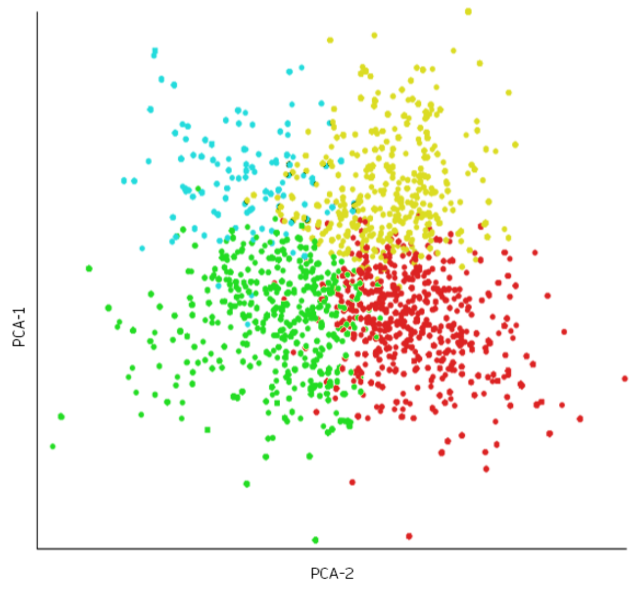

Figure 4. Plot showing clustering in input design space (color indicates design cluster)

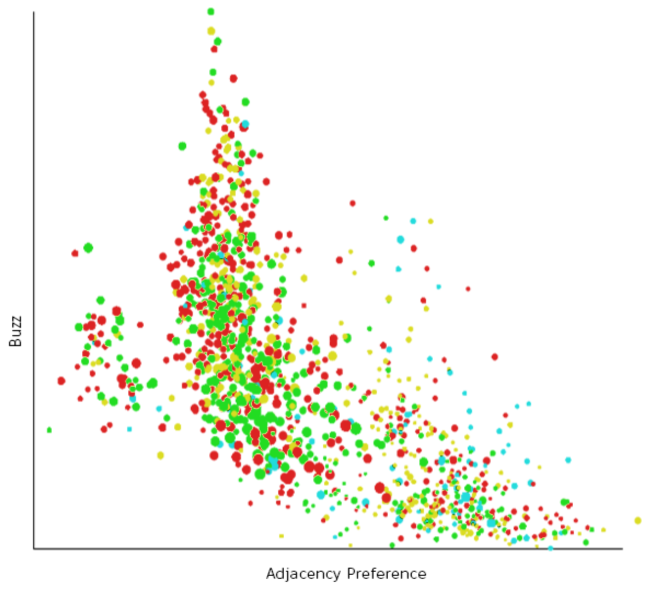

Figure 5. Plot showing tradeoff between two objectives (color indicates design cluster)

3.6 After generative design

Once a set of interesting designs is selected, they can be further analyzed by the human designer, discussed with the stakeholders, and developed into a final design. It is important to note that since the MOGA follows a stochastic process based on sampling a limited number of designs from the design space, the overall optimal design will not necessarily be found through the search process. Furthermore, as discussed previously, not all aspects that are important to an architectural design can necessarily be represented as a metric in the generative design model. Some aspects, such as beauty, cannot be quantified, and thus need to be considered once the generative design process is complete.

Finally, most generative design models including the space-planning model presented in this paper are fairly abstract and oversimplified, providing only rough geometry, boundary, and location information. After a basic space-planning strategy is selected, there is still much refinement and design work to be done, including selecting architectural materials and designing connection details, to get it to the level of a final constructible design.

Therefore, the process does not end with choosing one of the designs found by the algorithm. Instead, a deep analysis of various high performing designs and their trade-offs should suggest potential design strategies that the designer can further explore to achieve a final best design.

4 CONCLUSION

This paper described our development of a generative design workflow for architecture, and our application of it for the design of a new office space for Autodesk in Toronto.

Although the results of this investigation have been very encouraging, the process also has some limitations. Currently, the placement of programs and individual people in the plan depends on the neighborhood geometry, and thus cannot be directly controlled by the MOGA. To get a better and more targeted search we would need to develop methods to directly parameterize this placement and expose those parameters to the algorithm.

Another limitation is that the calculation of each design is still relatively slow – about one minute for each design – which limits the amount of exploration we can do. Automatically analyzing 10,000 designs already dramatically improves the capacity of a human designer, but is relatively small considering it is sampled from a 45-dimensional design space. Distributing the execution of designs within a single generation over several computers in a network would allow many more designs to be evaluated.

Finally, the workflow can be improved by integrating other types of modelling, particularly machine learning, for quantifying aspects of the designs that are difficult or impossible to compute through direct calculation. This is particularly interesting because it might allow the computer to develop knowledge of various design factors such as comfort, beauty, or novelty that are crucial to good design but have traditionally been difficult to relate to a computer.

As these types of workflows continue to develop in the future, it is our hope that they not only allow designers to develop high performing design options, but also help them understand their design problems better through a more collaborative human-machine design interaction. This will allow us to move far beyond the basic automation of tasks evident in early CAD tools, and leverage the full potential of true computer-*aided* design.

REFERENCES

1. Chronis A., Tsigkari M., Giouvanos E., Aish F., Zaki A. A., "Performance driven design and simulation interfaces: A multi-objective parametric optimization process", *Proc. SimAUD* (2012)
2. Derix, Christian, "In-Between Architecture Computation". *International Journal of Architectural Computing 7*, 4 (2009), 565-585.
3. Gerber, D., Lin, S., Pan, B., Solmaz, A.S. "Design optioneering: multi-disciplinary design optimization through parameterization, domain integration and automation of a genetic algorithm," *Proc. SimAUD* (2012)
4. Hillier, Bill, et al. "Space syntax." *Environment and Planning B: Planning and Design 3*, 2 (1976): 147-185.
5. Johnson T., Dietz A. G. H., Weinzapfel G., Drauss R., Morris D., "Space Arrangement," *AD Architectural Design* 9 (1969)
6. Keough I., Benjamin D, "Multi-objective optimization in architectural design," *Proc. 2010 Spring Sim Multi-conference*, Orlando, Florida (2010), 1-8
7. Liggett, Robin S. "Automated facilities layout: past, present and future." *Automation in construction 9*, 2 (2000): 197-215.
8. Marler R.T. and Arora J.S., "Survey of multi-objective optimization methods for engineering," *Structural Multidisciplinary Optimization 26* (2004), 369–395
9. Murata, T. and Ishibuchi, H., "MOGA: multi-objective genetic algorithms," in *Evolutionary Computation, IEEE International Conferenc*e on (1995), Vol. 1, p. 289
10. Negroponte N., "Through a Humanism Through Machines," *AD Architectural Design 9* (1969)
11. Peponis J., Wineman J., Bafna S., Rashid M., Kim S. H., "On the generation of linear representations of spatial configuration," *Environment and Planning B: Planning and Design 25* (1998), 559-576
12. Turner A., Doxa M., O'Sullivan D., Penn A., "From isovists to visibility graphs: a methodology for the analysis of architectural space", *Environment and Planning B: Planning and Design 28* (2001), 103-121