The Potential of Evolutionary Methods in Architectural Design

Wassim Jabi¹, Barbara Grochal², Adam Richardson³
Welsh School of Architecture, Cardiff University, United Kingdom
http://www.cardiff.ac.uk/archi/
¹jabiw@cardiff.ac.uk, ²sar8bag@cardiff.ac.uk, ³sar8ar@cardiff.ac.uk

Abstract. In this paper we examine the potential of combining 2D shape packing algorithms and evolutionary methods in the design process. We investigate the ways such algorithms can be used in architectural design and how they may influence it. In the first part of this paper we introduce the theoretical framework of packing algorithms and genetic algorithms as well as the traditional design process and the nature of design problems. In the second part of the paper we introduce a software prototype that tests these algorithms in two contexts: the preliminary design of a shading façade pattern and the design of commercial housing layouts. The aim for both experiments was to generate optimal configurations based on user-defined criteria without resorting to exhaustive search. Several lessons were learned that point to the potential of evolutionary methods in architecture as well as the limitations of such methods. We conclude the paper with recommendations for further developing this research project.

Keywords. Evolutionary design; genetic algorithm; packing algorithm; scripting.

INTRODUCTION

In general terms, a genetic algorithm (GA) can be characterised as a highly parallel and adaptive evolutionary search method. GAs are described as parallel searching methods because they search for solutions using the whole population of possible options as opposed to altering a single potential solution (Frazer, 1995). Since the most favourable solutions are obtained by progressive alterations within the same population over time, Frazer also refers to them as adaptive. Due to the mentioned characteristics, GAs are becoming more popular and are being researched and increasingly applied to practical problems.

Shape packing algorithms are optimization methods that attempt to pack shapes together within a set boundary. In one variation of the problem, a shape-packing algorithm is designed to pack as many shapes as possible, without overlapping them, and attempts to achieve a required minimum coverage area to minimize waste (Lodi et al., 2002). In mathematics, circle packing focuses on the geometry and combinatorial character of packing of circles of either equal or arbitrary size (Stephenson, 2005). For circles of equal size, it has been mathematically proved that a hexagonal honeycomb arrangement of circles produces the highest density (Hsiang, 1992). In architecture, shape packing can be used in many pattern-based problems where density, number of packed elements and spatial relationships between elements is important.
The aim of this paper is to study the potential of combining 2D shape packing algorithms and genetic algorithms (GA) in the design process. It investigates the ways such algorithms can be used as tools for aiding architectural design and how these methods may influence the architectural design process itself. This will be done by conducting two experiments based on the constructive design methodology where the two ‘constructs’ tested would be a software prototype that combines a 2D shape-packing algorithm and a genetic algorithm tested in two experiments. This paper discusses some of the advantages as well as limitations of such tools as design aids.

**BASIC STRUCTURE AND SEQUENCE OF GENETIC ALGORITHMS**

Genetic algorithms consist of two separate spaces: the *search space*, containing *genotypes*, and the *solution space*, containing *phenotypes* (Bentley, 1999). The genotypes, which are the coded solutions to the problem, have to be mapped onto the actual solutions i.e. the phenotypes, which are in the solution space (Figure 1). Mapping refers to the process of assigning the genotypes from the search space to corresponding phenotypes in the solution space. This has to happen before the *fitness* of each solution can be evaluated. The fitness of a solution is assessed according to the *fitness function* that assigns scores to all solutions. The more suitable the solution to solve the problem at hand, the higher is the fitness score (Mitchell, 1995). Effectively, the solutions with higher scores will have a greater probability of being selected and reproduced in the next generation (Figure 2).

**THE TRADITIONAL DESIGN APPROACH AND ‘WICKED PROBLEMS’**

In the second half of the 20th century researchers brought the design process to the focus of scientific study (Cross, 2007). They investigated it and outlined the basic sequence of actions involved in it, mainly in order to introduce new aiding tools and regulate it. This research has proven that systemising the design process is not an easy task mainly due to the fact that design problems are classified as ‘wicked’ (Rittel and Webber, 1973). This term refers specifically to the disciplines of social planning, politics and design. Firstly, in most of the cases the design problems cannot be comprehensively formulated. This is due to the fact that nowadays the design process of a specific building involves collaboration between different parties, which hinders arriving at specific requirements early in the project development. Usually, the design problems appear and become clearer as the process proceeds. Secondly, since design is a collaborative effort between different parties, it has to unite what are sometimes radically contradictory interests. It has to take into account a number
of other factors such as moral and social aspects, aesthetic impact and sustainability. Thus, the design solutions cannot be rationalised since design is not merely a pragmatic problem-solving or simple optimisation leading to one right solution. Even though usually one solution is sought, the possibilities for arriving at it are limitless. Thirdly, because there is no linear sequence for a design process (Lawson, 2005), there is also no apparent beginning or end to it. The information needed to make decisions is never fully complete and thus the state of the design problem is constantly evolving. Furthermore, in modern design thinking, problems and solutions are deemed to emerge together during the design process. That is, finding design solutions may cause other, “higher-level” problems somewhere else. Therefore the design process involves finding a balance between solving some problems in one place and causing undesirable effects somewhere else. Rittel and Webber emphasize that these ‘wicked’ design issues, unlike science, depend heavily on the designer’s subjective value judgements. These main characteristics are obstacles when working with algorithms and computer programs that need specific requirements and clearly defined rules in order to perform their tasks.

THE CONSTRUCTIVE METHODOLOGY
This part of the paper will describe two algorithmic experiments employing custom software created by the authors using the MAXSCRIPT scripting language for Autodesk 3ds Max. The software integrates a genetic algorithm with a shape-packing algorithm that operates on any 2D boundary. The first experiment explains the basic logic and functioning of the GA based on façade pattern design. This supplied valuable data for the discussion of the advantages and limitations of this tool. The second experiment concerns a more realistic case of housing layout design based on a real-life master plan. It has to be pointed out here that both of the cases present an integrated approach towards evolutionary design put into practice. The experiments will focus on first designing a pattern or layout and then optimising it based on the design criteria set by the designer.

EXPERIMENT 1: FAÇADE PATTERN DESIGN
The design of the first experiment involves the creation of a panel façade of size 10,000x10,000 mm. The main design goal is to achieve 40-50% of the area coverage of the designed pattern in order to provide the required shading. The second design aim is to design the pattern with 2000 circles of various radii. Both of the design goals have to be achieved following basic criteria set up at the beginning of the process using the custom-designed software (Figure 3). These are as follows:

- Min. Radius: 50 mm
- Max. Radius: 8000 mm
- Buffer: 20 mm

The range for the minimum and maximum radii was decided bearing in mind that the wider the range the more variety will be sustained in the population. These two values could also represent the radii of the smallest and biggest drills used for making the pattern. The reasons for choosing the radii range can be very different. The ‘buffer’ parameter refers to the area between circles where no other elements are allowed.

Genotype and phenotype
Since the design aims at creating a façade pattern made out of circles of various sizes the radius is the only information contained in the genotype (Figure 4). As shown, the genotype is the number falling within the specified range, where the phenotype
is the assigned representation of the genotype – in this case it is a circle of that particular radius.

**Fitness function**

In order to encourage variety within the panel the fitness function favours the circles with radii as close to the minimum or maximum radius values as possible. This will secure more diversity within the population and will create more interesting patterns. Thus, the fitness function is defined as follows:

\[
F = \frac{|R_i - R_{av}|}{R_{av}}
\]

where \(F\) is the fitness of the individual, \(R_i\) is the radius of the individual circle, and \(R_{av}\) is the average of the specified minimum and maximum radii.

**The GA sequence**

The initial population is created randomly, covering the entire range of possible solutions (search space). In case of the script used for this experiment the new individuals were created using a circle-packing algorithm until the maximum number of attempts for fitting more individuals has been reached (in this experiment it was set at 50,000 attempts). In such a case usually the initial population does not reach the maximum number of individuals (in this case 2000 individuals). After the initial population has been generated the fitness of each individual is calculated. The obtained fitness scores are then used for selecting the fittest individuals and placing them in the mating pool. We specified a constant 50% survival rate and a 1% mutation rate throughout the experiment and implemented a "roulette wheel" selection method to select the fittest candidates while maintaining a similar diversity to the one found in natural selection. After the individuals for the mating pool have been selected the process of reproduction begins using crossover and mutation of their genotypes. The process of generating populations continues until a termination condition is met. Termination takes place either when the population target is met or when the algorithm reaches the maximum number of attempts to fit the individuals (50,000). The section below describes the results of the four tests created based on the rules described above.

**Results**

We conducted four tests in order to meet the design requirements and solve the stated design problem: Achieving area coverage of 40-50% with 2000 circles. In each test, we iterated through four generations (Table 1).

**Discussion and Comments**

As the results show, meeting both of the design goals where the fitness function is awarding the radii from the extremes of the range of 5-800 cm, is rather unlikely to be achieved in the span of 4 generations even if the number of attempts is 50,000. The outcomes might have been different if the number of attempts was increased to 100,000 or more. This is, however, an area for further research that lies outside of the scope of this experiment.

Compared to the non-optimised first generation of packed circles, it is evident from conducting only four tests, that applying the GA dramatically increases the number of circles to meet the goal to pack 2000 individuals within the prescribed boundary (Figure 5), but that has two side-effects: 1) The average radius of circles decreases, and 2) the over-
all coverage area of these circles decreases as well (Figure 6). Also, it can be concluded that the bigger the coverage area, the smaller the number of elements. In all of the tests the maximum number of circles was achieved when the coverage area was consistently below 40%. Based on that, the main design goals had to be revised. Because the GA proved that both design goals couldn’t be achieved simultaneously, the designer has to decide which is a priority – the coverage area or the number of packed elements. Since the main aim of the experiment was based on creating the required shading pattern, the coverage area took precedence. Therefore the façade pattern with the coverage area within the range and achieved with the biggest number of circles was chosen as the proposed design solution. In the four conducted tests, this was achieved in the second generation of the third test with 1449 packed individuals and 49.36% coverage area (Figure 7).

It is clearly visible from both the data and the visual graphs (Figure 6) that even after the maximum number of packed elements has been achieved the GA was still breeding a population of increasingly smaller circles. This occurred due to the fact that the fitness function was awarding both extremes – the smallest and the biggest circles. The mating pool quickly biased itself towards smaller circles after the first generation because at the point when the maximum number of elements was reached there were far more circles with radii closer to the minimum than those closer to the maximum radius. That is, since there were a larger number of smaller circles and because they were considered just as fit for breeding as large circles, there was a higher probability of choosing them for breeding the next population. This strength in numbers phenomenon initiated a vicious cycle of breeding smaller and smaller circles while larger circles quickly became extinct. The solution seems to approach a plateau after the third generation. An interesting contradiction is that the overall results did not improve with the subsequent generations even though the individuals’ fitness was increasing. From an interesting perspective, this result supports a case for diversity where even if individual fitness is high, the overall performance of the population is unsatisfactory due to a lack of diversity.

Table 1
Results of running four tests each with four generations.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Generation No.</th>
<th>Packed individuals</th>
<th>Area Coverage (%)</th>
<th>Avg. Fitness</th>
<th>Avg. Radius (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>195</td>
<td>60.26</td>
<td>0.98</td>
<td>193.42</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1489</td>
<td>44.28</td>
<td>0.98</td>
<td>86.55</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2000</td>
<td>34.73</td>
<td>0.98</td>
<td>70.47</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2000</td>
<td>27.82</td>
<td>0.98</td>
<td>64.84</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>212</td>
<td>55.78</td>
<td>0.95</td>
<td>190.03</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1401</td>
<td>45.42</td>
<td>0.98</td>
<td>90.24</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2000</td>
<td>34.62</td>
<td>0.98</td>
<td>70.15</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2000</td>
<td>26.72</td>
<td>0.98</td>
<td>63.20</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>217</td>
<td>50.66</td>
<td>0.95</td>
<td>186.06</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1449</td>
<td>49.36</td>
<td>0.98</td>
<td>83.92</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2000</td>
<td>31.34</td>
<td>0.98</td>
<td>66.57</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2000</td>
<td>25.09</td>
<td>0.98</td>
<td>61.68</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>204</td>
<td>57.93</td>
<td>0.95</td>
<td>192.08</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1479</td>
<td>45.90</td>
<td>0.98</td>
<td>86.58</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2000</td>
<td>34.20</td>
<td>0.98</td>
<td>69.44</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2000</td>
<td>27.52</td>
<td>0.98</td>
<td>64.19</td>
</tr>
</tbody>
</table>
A visual representation of the test results. The blue circles represent the parents chosen for breeding the next generation.
EXPERIMENT 2: HOUSING LAYOUT DESIGN

The second experiment explores the application of a genetic algorithm to the design of a housing layout. As an early stage test, it ignores the vast number of variable factors that influence the design of such schemes and focuses on achieving the greatest density of housing units. This ‘House Packing’ script enables the user to define a series of areas (representing buildable plots of land), the program then places houses around the perimeter of these areas and orients them to the nearest edge (representing a road). The fitness of each individual is calculated based upon their proximity to their nearest neighbour. This factor of ‘remoteness’ ensures that the density of houses in subsequent generations increases. The program was tested on a case study housing development that has recently been granted planning permission. For the purpose of this research, this provided a realistic framework onto which the program could be applied. The program is able to vary the size of the houses within what has been defined as a realistic range, based upon the house sizes found in the case study. The intention is to achieve a more realistic configuration of houses. The input parameters were as follows:

**Phenotype Parameters:**
- Min. Radius: The minimum separation distance between houses.
- Min./Max. Length: The range of lengths of the houses being generated.
- Min./Max. Width: The range of widths of the houses being generated.
- Dist. To Road: The minimum distance from the road at which houses can be placed.

**Population Parameters:**
- Maximum: Sets the maximum number of houses to be generated.

![Figure 6](image1.png)
Left: Graph representing the decreasing average radius over four generations in all four tests.
Right: Graph representing the decreasing coverage area over four generations in all four tests.

![Figure 7](image2.png)
The proposed façade pattern based on overall results.
• Survival Rate (%): Controls the percentage of the population that will survive and be allowed to breed the next generation.
• Mutation Rate (%): Controls the chances of an individual within the population to mutate.
• Max. Attempts: The maximum number of attempts that the computer is given to place the houses correctly.

**Results**
The two variables that control the effectiveness of the genetic algorithm are the survival rate and the mutation rate. To find optimal values for these, we carried out a series of evaluations that first tested the system at varying survival rates and then varying mutation rates. We tested the program over five generations on the case study layout. For each setting, we recorded the three read-outs: ‘Packed’, ‘Coverage Area’ and ‘Average Fitness’. This gave an indication of how well each rate was performing, though the most telling result was the average fitness score as this is more directly linked to the overall efficacy of the algorithm. It would seem that the rate showing the greatest increase in average fitness over consecutive generations should be selected as the optimal setting. Over five generations this would appear to be the 100% survival rate. However, a survival rate of this magnitude stifles the genetic algorithm by preventing it from removing poor performing individuals. As a result the values for ‘Packed’, and ‘Coverage Area’ tend to peak very early, and more often than not exceed those achieved by lower survival rates, as the program attempts to squeeze more and more houses onto the site. The average fitness score on the other hand will usually remain relatively low, since the proposed solution still contains a number of poor performing individuals. The results recorded in this experiment appear contradictory, as the highest average fitness score is achieved by the 100% survival rate. The problem with such a high survival rate is that the algorithm is relying entirely on the mutation of individuals to increase the average fitness. It is more of a brute force trial and error approach rather than systematically breeding a better solution. This method appears to work for this particular experiment, as the fitness score is very closely linked to the number of individuals placed (more individuals = lower factor of remoteness = higher fitness score). However the purpose of this experiment is to test the potential of the genetic algorithm, and thereby following a method that nullifies part of the breeding process would be contrary to that goal.

We found that the optimal survival rate was 60% and mutation rate was 40%, this coincides with experiments by other researchers that suggest a mutation rate of approximately 50% (Elezkurtaj and Franck, 1999). Using these settings, we ran the algorithm for 15 generations to discover the effectiveness of the optimisation process. The results show a general positive trend in the average fitness score and number of packed houses, indicating that the optimisation process is functioning correctly (Figure 8). What is interesting is the amount of fluctuation between generations. These results indicate a pattern of 3-4 successive increases followed by a significant decrease, the magnitude of which reduces with each repetition. This is an indicator of how the genetic algorithm works. Over successive generations,

![Figure 8](image-url)
the range of fitness scores will decrease as they all become fitter (and the average fitness increases). The result of which, in terms of the roulette wheel selection method, is that each individual has a much more equal chance of being in the percentage of the population which survives to the next generation (the survival rate). Conversely, they also have a more equal chance of being removed from the population. This is demonstrated in the results where after 3 generations at a 60% survival rate, new individuals with lower fitness scores replace 40% of the relatively high scoring population.

**Discussion and Comments**

It is clear that a genetic algorithm based design aid holds great potential in increasing the efficiency of the commercial housing design process. The ability of this kind of software to act as a catalyst for design ideas whilst simultaneously conforming to a plethora of constraints is something that, as the need for greater efficiency and precision within the design process grows, is going to prove invaluable. The most pressing question raised by this research is the way in which the software should be integrated into the design process.

For this experiment we shared the results with a group of architectural practitioners to gauge their reaction. Unsurprisingly, this expert consultation exposed a desire amongst the designers to have a greater amount of input in the generation of a solution. This gave an interesting insight into the way that they feel about the software. One expert questioned the ability of software to replicate the “human ability to ... make a subjective judgement”, demonstrating a lack of trust in the system to generate a complete design solution. The designers want the software to carry out the time consuming, menial tasks, enabling them to focus their time on the more skilled areas of design, but do not wish the software to shift all decision making from human to machine. The lack of trust also answers the question of the potential marginalisation of the architect’s role through the advancement of digital design tools. The designers do not see the technology as marginalising their role; they feel that “If anything the development of IT in design has given more control back to the designers”. This further reinforces the role of the program as a design aid, not as a complete design solution.

**CONCLUDING REMARKS**

The two experiments we have conducted reinforce the notion that evolutionary methods have many advantages. Most importantly, the experiments illustrated that evolutionary methods are indispensable when dealing with a large potential solution space. Rather than conducting a manual and exhaustive search for the best solution from a large data set, evolutionary methods allow the designer to set target goals and input parameters and rules that act together to search the population for the best possible candidates and use that pool of candidates as an input to breed an even better solution. Genetic algorithms allow designers a more precise method to achieve the desired goal as it faithfully applies the states rules and precisely measures the performance of individuals and the overall population with each generation. Given their parallel search nature, genetic algorithms can help us speed the whole design process when the pool of options is large. Additionally, evolutionary methods are capable of supplying very surprising outcomes aiding the designer’s creativity and suggesting new solutions. Finally, due to their high efficiency, evolutionary methods allow the design more time to focus on the quality of the design, omitting menial tasks such as ensuring that the proposed design is compliant with the stated goals and constraints.

The experiments also exposed some important limitations of these methods. Due to the ‘wicked’ nature of design problems, it is not always feasible to state and code clear design rules and objectives. The design process is highly dynamic and often changes course. Although possible in future iterations of the software, the genetic algorithm code we have developed was not designed to handle changing design goals between generations. The addition of new parameters and fitness score methods between
generations could prove even more difficult to implement. As most computer-based systems, genetic algorithms cannot replace the tacit knowledge, common sense, and intuition of human designers. In particular, these methods cannot replace human judgment since it is difficult to encode in the algorithm, as it is not based on clearly definable rules. Yet, we believe these methods fundamentally change the design process and the role of the designer. We envisage the designer mutating from the role of the supreme creator of the final outcome to the role of the maker of rules. In partnership with sophisticated evolutionary systems, the designer can then explore the plethora of solutions offered by these methods fluidly shifting its rules and input parameters that in turn alter its path of evolution.

REFERENCES


