EFFICIENCY VERSUS EFFECTIVENESS

A Study on Constraint Handling for Architectural Evolutionary Design

LIKAI WANG[1], PATRICK JANSSEN[2] and GUOHUA JI[3]
[1,3]*School of Architecture and Urban Planning Nanjing University*
[1]*dg1436002@smail.nju.edu.cn* [3]*jgh@nju.edu.cn*
[2]*School of Design and Environment National University of Singapore*
[2]*patrick@janssen.name*

Abstract. This paper investigates the impacts of constraint handling on the evolutionary designs in terms of time efficiency and evolutionary effectiveness. To analyse this issue systematically, three generative models with different constraint handling strategies were constructed. The locality of the models and the associated positive and negative impacts on evolutionary designs were analysed.

Keywords. Constraint handling; locality; evolutionary design; time efficiency; evolutionary effectiveness.

1. Introduction

In the last decade, the use of evolutionary design has been gaining popularity as a strategy for architects to improve building performance. By defining generative models (GM) and evaluative models, evolutionary algorithms can be used to explore complex design spaces and to discover creative design alternatives for different objectives.

Along with the instrumental development of various evolutionary design tools, GMs have become an important area of research. Form-finding approaches have been theoretically introduced by early pioneers such as Frazer (1995) and Bentley and Kumar (1999). Following these pioneers, other researchers and designers have attempted to find ways to construct GMs with wide-ranging formal diversity. Different modelling approaches have been experimented with, including surface and solid modelling approaches using NURBS and Boolean operations, and rule-based modelling approaches such as cellular automata and agent-based modelling.

When the evolutionary design is applied to real-world problems, it is often difficult to find viable solutions within the short time-frames and deadlines set by practice. In such cases, the overall progress made by the search process in the short term is often much important than the ability to find the true optimal solutions in the long term.

We describe the search process using two qualities: efficiency versus effectiveness. We use the term *efficiency* to refer to how quickly the search is able to find reasonable solutions. We use the term *effectiveness* to refer to how consistently the search is able to find improved solutions. Aside from the impacts of the evolutionary algorithm, the way that GMs are implemented also affects these two qualities significantly.

As GMs become complex, these two qualities come into conflict with one another. For example, some search processes are efficient but not effective: in the short term, they quickly discover some reasonable solutions but are then rather poor at improving on those solutions in the long term. Other search processes are effective but not efficient: in the long term, they may discover excellent solutions through consistent incremental steps, but they are too slow for use in the short term.

Efficiency and effectiveness are strongly affected by the way that GMs handle *constraints*. In order to control design search spaces, GMs can incorporate constraint handling techniques that exclude infeasible solutions by using explicit or implicit rules (Bentley and Kumar, 1999) in the genotype-phenotype mapping or phenotypic representing (Eiben and Smith 2004).

Embedding constraints in a GM can significantly reduce the size of the search space (Janssen et al. 2014). In general, a smaller search space will require fewer computational resources, thereby improving the overall efficiency. However, such additional constraints also have a negative impact in that they weaken the *landscape locality* of the search space. Landscape locality is a concept that describes how well neighbourhood is preserved in the genotype-fitness mapping (Galván-López et al. 2011). If the neighbourhood is well preserved, then a small change to a genotype will result in a small change to the phenotype, which will, in turn, result in a small change in fitness. In general, the weakened locality will make it harder for the search process to consistently improve performance, thereby reducing overall search process effectiveness.

Thus, when embedding constraints into a GM, a trade-off needs to be considered (Figure 1 left). One the one hand, additional constraints will compress the search space and improve efficiency in the short term. On the other hand, additional constraints will weaken locality and reduce effectiveness in the long term (Figure 1 right).

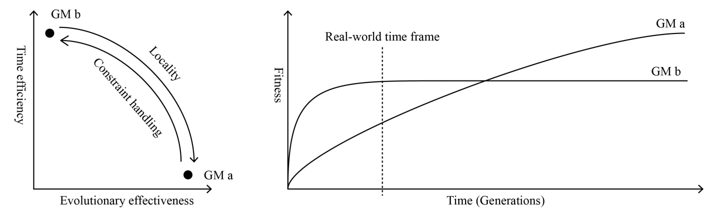

Figure 1. The relationship between efficiency and effectiveness of the search process.

When designers implement GMs, the issue of finding an appropriate balance between efficiency and effectiveness should be considered. In most cases, designers may prefer to use simple GMs with few constraints despite the fact that this will result in a larger search space. However, such large search space will make the search process difficult to find reasonable solutions in the short term. This may undermine the practical value of using any such algorithms in practice. The development of GMs that achieve a more balanced trade-off between efficiency and effectiveness is therefore desirable.

Taking this as the point of departure, this study mainly focuses on the impact that different constraint handling techniques have on the balance between efficiency versus effectiveness. To investigate this issue systematically, three GMs are constructed using different constraint handling techniques. Each GM is then analysed from the perspectives of the trade-off between efficiency and effectiveness. Finally, the advantages and disadvantages of the constraint handling techniques are analysed.

2. Case Study

A high-rise 40 storey office design with an atrium and vertical gardens is introduced as a case study. Such atriums and vertical gardens can be used to improving environmental performance in many regions, from tropical to temperate climate zones (Wood and Salib, 2013). However, finding an appropriate trade-off between economic performance and environmental performance requires atriums and vertical gardens to be carefully controlled and configured within the building volume.

For the case study, a fixed structural frame is used, consisting of a rectangular plan office floor with an open atrium in the centre rising up through the whole building, flanked by two structural cores on either side.

All three GMs use the same general mechanism for inserting vertical gardens: first dividing the tower volume into 3D cells, and then switching cells from solid to void, thereby creating complex patterns of interlocking indoor and outdoor spaces. For the subdivision of the tower into cells, floors are first grouped into ranges of 2-to-5 floors, and each group is then divided in plan into 11 cells (see Figure 2). (Applying floors ranges is not only for reducing the number of parameters but also for the reason that single floor vertical gardens are impractical.)

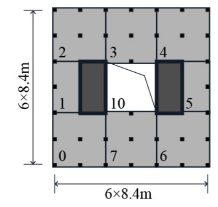

Figure 2. The structural frame.

Based on this structural frame and generative mechanism, three GMs are implemented, called the *naïve* GM, the *constrained* GM and the *constrained-repaired* GM, each inserting vertical gardens with progressively more constraints.

2.1. THE NAIVE GM

The naïve GM has the fewest constraints. In this GM, the on-off condition of every cell is defined by a binary switch externally. Such simple control structures are easy to construct and often applied in these types of optimisation problems.

The genotype defines the layout for ten floor groups. For each floor group, the genotype contains two parameters. The first parameter is an integer between 2 and 5, defining the number of floors in that group. The second parameter is a string containing 11 binary switches, defining the solid and void pattern for the 11 cells in that floor group.

The ten groups will each have variable floors, which may result in either too many or too few floors. Some simple rules are therefore applied in order to ensure that the correct number of floors is achieved. If the total floors are less than 40, then the topmost floor layout will be taken to fill the rest floors. If the total floors are greater than 40, then extra floors will be culled.

The simple genotype-phenotype mapping ensures that the GM has good locality, which should result in an effective evolutionary process. However, the simple mapping also results in a very large genotype and phenotype search space with large numbers of naïve solutions. Such a large genotype and phenotype space can severely hinder the evolutionary process. As a result, constraints may be needed in order to ensure that time taken to discover reasonable solutions is acceptable.

2.2. THE CONSTRAINED GM

In practice, certain basic architectural design rules for atriums and vertical gardens can be defined, which can then be implemented as constraints in the GM. First, the number of vertical gardens should be limited to one vertical garden per floor. Second, the size of a vertical garden should be controlled and should not be significantly larger than that of the indoor space. Third, vertical gardens should be connected to the atrium to facilitate natural ventilation.

For this GM, the genotype still defines the layout for ten floor groups. However, in order to constrain the GM to the above rules, certain modifications were introduced into the control structure. For each floor group, the genotype now contains three parameters. The first parameter is the same as the naïve GM, and defined the number of floors in that group.

The second and third parameters replace the binary string. Instead of simple binary switches, these parameters are used to create voids through an explicit rule-based approach. Since there are only two cells directly connecting the atrium, and vertical garden must include one of these two cells. The second parameter is either 0, 1, or 2. If the value is 0, then it indicates that there will be no void, in which case the third parameter can be ignored. If the value is 1 or 2, then it indicates which one of the two cells adjacent to the atrium will be included in the vertical garden. Finally, the third parameter is an integer that selects a void pattern from a predefined set of patterns. To restrict the size of the vertical garden, the number of cells in each void pattern is limited to a maximum of 5, which results in a total of 14 unique patterns (Figure 3).

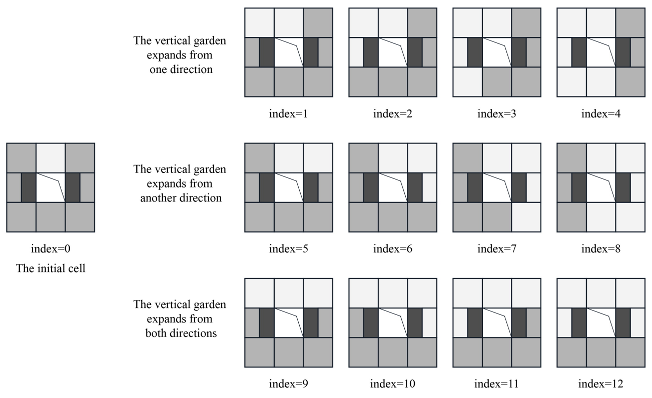
The vertical garden expands from one direction
index=1
index=2
index=3
index=4
The vertical garden expands from another direction
index=5
index=6
index=7
index=8
index=0
The initial cell
The vertical garden expands from both directions
index=9
index=10
index=11
index=12

Figure 3. Different floor layouts from one initial position.

The constrained rules result in the possibility of there being neutral mutations that have no effect on the phenotype. In some circumstances, the GM is indifferent to the value of the third parameter if the second one defines that no vertical gardens are generated. Although this will significantly degrade locality, the constrained rules are able to significantly reduce the size of the search space and also ensure that most of the phenotypes meet basic architectural design rules.

The independence between floors layouts, however, can create certain types of voids that may be problematic. Two key types of problematic voids are identified: oversized voids in cases where two voids meet above one another and become merged, or cross-diagonal voids in cases where two voids meet at a point on the diagonal. Such voids are hard to avoid within the generated designs changing dynamically.

2.3. THE CONSTRAINED-REPAIRED GM

The genotype for the constrained-repaired GM uses the same control structure as the constrained GM. However, additional implicit rules are introduced as repair operations in order to amend the oversized voids and cross-diagonal voids generated by the constrained GM. If an oversized or a cross-diagonal void is generated, then a repair operation will modify a floor from one or more groups and will assign all cells on that floor to be non-void. In the case of the oversized void there being another operation, floors are iteratively removed from the top and the bottom of the void, until a suitable height is reached (*a-a'* in Figure 4). In the case of the cross-diagonal void, all cells on the floor in the middle will be assigned non-void, so that the two voids become disconnected (*b-b'* in Figure 4).

These repair operations may, however, result in additional problematic conditions being generated. In particular, inserting non-void floors in certain groups can result in many single-floor pendulous cells which are hard to rent or construct. Hence, an extra repair operation is needed in order to correct these conditions. This repair operation will identify isolated or pendulous cells and will switch them to the opposite solid-void condition. Due to the fact that additional problematic conditions can continuously emerge after the execution of the first and the second repair operations, these operations are run in a loop until all infeasible conditions have been eliminated.

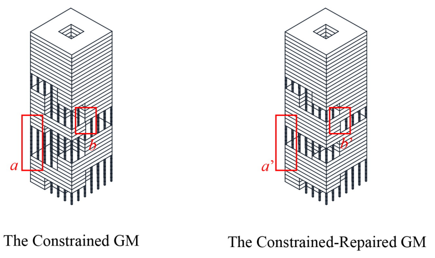

Figure 4. Example of repair operations.

2.4. THE FITNESS FUNCTION

In order to evaluate the generated solutions, a simplified economic index was used. This index has the advantage that it is fast to calculate.

For each floor, the fitness function calculates the the potential profit that can result from the rentable floor area, and subtracts three construction cost factors: the core cost, the slab cost, and the facade cost.

- Potential profit: Rentable floor area multiplied by a factor that gives preference to south facing spaces and spaces on the upper or lower floors (due to the better view or accessibility).
- Core cost: Core area in plan multiplied by a factor that increases with floor number.
- Slab cost: Slab area (excluding core but including outside spaces) multiplied by a factor that increases with floor number.
- Facade cost: Facade area multiplied by a constant cost factor.

Aside from the above index for every single floor, an upper limit of the gross area for the whole building is also defined. A building whose gross floor area surpasses the predefined limit will have its potential profit proportionally scaled back according to the excess area.

Based on the above fitness function, every change in the building will cause a corresponding change in its fitness. This fitness function can drive the evolutionary process towards a valid and feasible solution from the perspective of architectural designs.

3. Results

Based on the external (naïve GM), explicit (constrained GM) or implicit (constrained-repaired GM) rules, different constraint handling strategies applied in the presented GMs have a strong impact on the formal and structural features of generated designs (see Figure 5). It is clear that, by embedding more constraints, the associated rationality of the designs in the population are improved significantly even before the evolving processes. The capability to avoid infeasible solutions from being generated means that it is easier for the evolutionary processes to discover promising areas of the search space, thereby improving the efficiency of the search process.

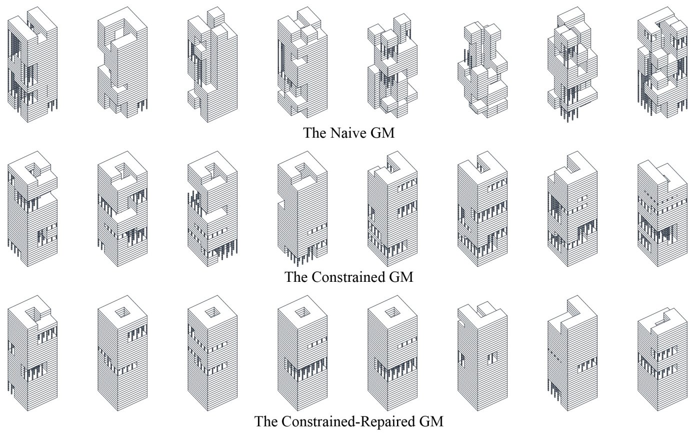

Figure 5. Random sampling based on the presented GMs.

In a first stage, locality and fitness were analysed by randomly generating populations of designs for each GMs, without running any evolutionary search process. However, although the additional constraints would be expected to improve efficiency, they may also reduce effectiveness due to the negative impacts on locality. In order to investigate the impacts of weaker locality, a second stage of the research ran the evolutionary search process based on the presented GMs and analysed both efficiency and effectiveness.

3.1. GM LOCALITY AND FITNESS

When measuring locality, only the phenotypic response to small genotype changes are taken into account, since good respondency between two neighbouring genotypes is critical to ensure that the design can be continuously evolved, especially when the evolutionary process begins to converge and differences between parents and children reduce. There are several approaches to evaluate the locality. For this study, a random sampling approach referred as fitness clouds (Vanneschi et al. 2004) was applied. This approach evaluates locality through the phenotype respondency. The respondency is calculated by the fitness difference between two phenotypes which are generated by a pair of neighboring genotypes. Thus, for calculating the locality of each GM, a Latin hypercube sampling of 100 pairs of neighbouring genotypes were selected, and the associated phenotype respondency was then evaluated.

The quantitative criteria defining locality currently remains a disputed issue. One of the arguments is how to define neutral mutations (the fitness difference is 0 between two neighbouring genotypes). Table 1 shows three common methods (Galván-López et al. 2011), but it is also reported that none of these methods can accurately predict the locality in all different scenarios. For this study, since there are considerable numbers of many-to-one genotype-phenotype mappings in the constrained and constrained-repaired GMs, simply ignoring such neutral mutation is inadvisable. As the result, *Def0* and *Def1* were applied for the evaluation of the locality.

Table 1. Definition of different fitness distances.

Fitness Distance	***Def0***	***Def1***	***Def2***
0	Non-local (very small)	Non-local (small)	Local
Small	Non-local (small)	Local	Local
Large	Non-local (large)	Non-local (large)	Non-local

Figure 6 shows the locality calculated using two different methods. The general tendencies for these two methods are similar, with only minor differences in the actual values. This result reveals that the locality can be predicted coherently by two methods, which suggests that the locality of the presented GMs is accurate. The results show that the locality of the naïve GM is markedly better than that of the other two GMs. Between the constrained and constrained-repaired GMs, the additional constraints (repair operations) do not further degrade its locality significantly.

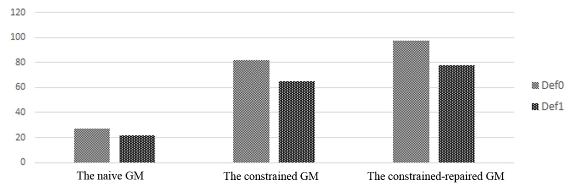

Figure 6. The statistical analysis of locality based on Def0 and Def1.

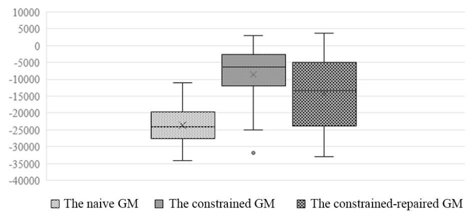

Figure 7. The statistical analysis of the fitness values.

Although the constrained and the constrained-repaired GMs are non-local in general, the constraint handling brings noticeable benefits. Figure 7 shows the box plot of the statistical results from the above samplings.The results show that the median and average fitness value of the constrained and constrained-repaired GMs are better than that of the naïve GM, which can allow the evolving process more likely to find feasible solutions during the early stochastic search phase. However, between the constrained and constrained-repaired GMs, the additional constraints make the average fitness value of constrained-repaired GM drop markedly. This can be explained by the reason that most solutions corrected by the repair operations usually have larger gross area surpassing the upper limit. But, even so, the maximum fitness value of the constrained-repaired GM is still higher than that of the constrained GM.

3.2. EVOLUTIONARY EFFICIENCY AND EFFECTIVENESS

In order to further investigate the impacts of the different constraint handling strategies had on efficiency and effectiveness, the evolutionary search process was run and the results analysed. The evolutionary algorithm was executed using the Rhino-Grasshopper environment, and the standard genetic algorithm in the Galapagos was applied. The population size was set to 100. Due to the large genotype space for some of the presented GMs, the population of the initial generation was raised to 1000. Meanwhile, to avoid the premature convergence, a higher mutation rate and a lower selection pressure were used. (In Galapagos, the settings are 25% for *maintain* and 25% for *inbreeding*). Last but not the least, the evolutionary process was repeated five times in order to reduce the impact of stochastic variation.

Figure 8 shows the trend lines of fitness improvement during the evolutionary processes. Certain key features of the trend lines seem to confirm the expected impacts of locality. The more local of the GM is, the more smoothly and gently the trend line grows. However, it is also clear that the improved effectiveness has not been able to deliver better designs, as the fitness levels of the naïve GM are much lower than that of the other GMs. Thus, in this case, efficiency seems to outweigh effectiveness.

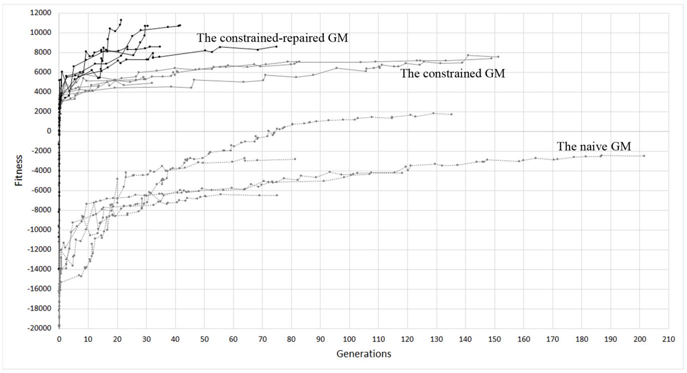

Figure 8. Convergent trend lines.

Even with the poor locality, the constrained and constrained-repaired GMs can significantly improve the time efficiency of the evolutionary designs and, more importantly, the fitness level of the evolved designs. Even though the poor locality causes fitness levels to fluctuate significantly, the associated reduced search space seems to be resulting in a faster convergence of the evolutionary process on reasonable solutions.

4. Discussion

This study has investigated the trade-off between efficiency and effectiveness when applying constraint handling in the GM. The results of the case study support the proposed hypothesis with regards to the efficient-effective trade-off (Figure 1). For the given time frame, reducing search space size was found to be more important than maintaining good locality. Constraint handling, therefore, had a positive impact on both the efficiency as well as the final quality of the evolved results, despite the reduced effectiveness.

In the second analysis, the designs evolved using the naïve GM never managed to surpass the designs evolved using the constrained and constrained-repaired GMs. But the search space defined by the naïve GM is a super-set of the search spaces defined by the constrained and constrained-repaired GMs. Therefore, given enough time, one might conclude that the naïve GM should be able to find the same solutions as the other two GMs, and might even be able to find better solutions (Eiben and Smith 2004). However, when practical limitations are imposed on computational resources, the time taken may simply be too long.

To conclude, when available computational resources are not able to keep up with the ever larger design search spaces, shrinking the design space by incorporating additional constraints can be a beneficial strategy. In particular, such additional constraints can result in significantly improved design being evolved within the time that is available. These benefits become more pertinent when applied to evolutionary designs approaches that require time-consuming performance simulations.

Acknowledgement

This paper was supported by the National Natural Science Foundation of China (51378248) and the China Scholarship Council (201706190203).

References

Bentley, P. and Kumar, S.: 1999, Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem, *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, Orlando, Florida, 35-43.

Eiben, A.E. and Smith, J.E.: 2004, *Introduction to Evolutionary Computing*, Springer, New York.

Frazer, J.: 1995, *An Evolutionary Architecture*, Architectural Association.

Galván-López, E., McDermott, J., O'Neill, M. and Brabazon, A.: 2011, Defining locality as a problem difficulty measure in genetic programming, *Genetic Programming and Evolvable Machines*, **12**, 365-401.

Janssen, P. and Kaushik, V.: 2014, Evolving Lego, *Rethinking Comprehensive Design: Speculative Counterculture, Proceedings of the 19th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2014)*, Kyoto 14-16 May 2014, 523–532.

Vanneschi, L., Clergue, M., Collard, P., Tomassini, M. and Vérel, S.: 2004, Fitness Clouds and Problem Hardness in Genetic Programming, *Genetic and Evolutionary Computation GECCO2004 Part II*, **3103**, 690-701.

A. Wood and R. Salib (eds.): 2013, *Guide to Natural Ventilation in High Rise Office Buildings*, Routledge.