University of Huddersfield Repository

Bentley, Peter J. and Wakefield, Jonathan P.

The Table: An Illustration of Evolutionary Design using Genetic Algorithms

Original Citation


This version is available at http://eprints.hud.ac.uk/3976/

The University Repository is a digital collection of the research output of the University, available on Open Access. Copyright and Moral Rights for the items on this site are retained by the individual author and/or other copyright owners. Users may access full items free of charge; copies of full text items generally can be reproduced, displayed or performed and given to third parties in any format or medium for personal research or study, educational or not-for-profit purposes without prior permission or charge, provided:

- The authors, title and full bibliographic details is credited in any copy;
- A hyperlink and/or URL is included for the original metadata page; and
- The content is not changed in any way.

For more information, including our policy and submission procedure, please contact the Repository Team at: E.mailbox@hud.ac.uk.

http://eprints.hud.ac.uk/
The Table: An Illustration of Evolutionary Design using Genetic Algorithms

PETER J BENTLEY & JONATHAN P WAKEFIELD
DIVISION OF COMPUTER AND INFORMATION ENGINEERING,
SCHOOL OF ENGINEERING, UNIVERSITY OF HUDDERFIELD, QUEENSGATE,
HUDDERFIELD, WEST YORKSHIRE HD1 3DH, UK
email: p.bentley@eng.hud.ac.uk

ABSTRACT

This paper describes an attempt to enable computers to generate truly novel conceptual designs of solid objects by using genetic algorithms (GAs). The current capabilities of the system are illustrated by the example of designing a table. Each individual table has its functionality specified by an explicit objective function, which is utilised by GAs to evolve candidate designs. These designs are represented using spatial partitions of 'stretched cubes'. The effect that varying the number of spatial partitions in each design has on evolution is investigated. Additionally, a method of producing symmetrical designs is explored.

1. INTRODUCTION

The genetic algorithm (GA) is a highly flexible and powerful search algorithm that uses principles of evolution observed in nature to direct its search [1,2]. GAs can tackle optimisation problems if these problems are formulated in terms of search, where the search space is the space containing all possible combinations of parameter values, and the solution is the point in that space where the parameters take optimal values. Indeed, GAs are commonly used to optimise designs, with some remarkable results [3]. However, in this paper we aim to demonstrate that GAs are able to do more than just optimise existing designs - we propose that they can create entirely new designs from scratch.

The area of design creation using genetic algorithms is a relatively unexplored area. Using GAs to create new designs has the potentially great benefit of new conceptual designs being automatically created in addition to optimal designs. So far research has been performed in limited ways, such as the use of GAs to create new conceptual designs from high-level building blocks [4] although this often seems to be little more than the optimisation of connections between existing designs rather than true design by the GA. Also the related area of Genetic art is becoming more popular, with various voting systems now being on-line on the internet (e.g. John Mount's 'Interactive Genetic Art' at http://robocop.modmath.cs.cmu.edu.8001). Other art-evolution systems using humans as design evaluators have been created [5,6], but as yet very few, if any, systems exist that can evolve a design from scratch with no human input during the evolution process. This paper describes an early prototype of an evolutionary design system, capable of doing just that. To demonstrate this, the system is set the task of designing a table. The table was chosen in order to allow investigation of some fundamental aspects of design and because it is a recognisable everyday object.

2. REPRESENTATION

Evolving designs from scratch rather than optimising existing designs requires a very different approach to the representation of designs. When optimising an existing design, only selected parameters need have their values optimised (e.g. for a jet-turbine blade, such parameters could be the angle and rotation speed of the blade). To allow a GA to create a new design, the GA must be able to modify more than a small selected part of that design - it must be able to modify every part of the design. This means that a design representation is required, which is suitable for manipulation by GAs. Many such representations exist, and some are in use by the evolutionary-art systems: a variant of constructive solid geometry (CSG) is used by Todd & Latham [6], others use fractal equations (e.g. John Mount - see the WWW address given above), and Dawkins [5] uses tree-like structures. For a system capable of designing a wide variety of different solid object designs, however, a more generic representation is needed.
A variant of spatial-partitioning representation has been developed for this work [7]. This representation uses spatial partitions of variable sizes, each one being a 'cuboid' with variable width, height, depth and position in space. Each primitive shape can also be intersected by a plane of variable orientation, but this refinement to the representation will not be considered in this paper. One of the many benefits of using such a variable-sized primitive shape as a spatial-partition is that few partitions are required to represent designs. Significantly, the fewer the partitions in a design, the fewer the number of parameters that need to be considered by the GA. A disadvantage, however, is that it is possible to represent illegal designs using the representation, i.e. it is possible for two spatial-partitions to overlap each other causing redundancy and ambiguity.

The simplest way to overcome the problem of illegal designs is to remove them from the population if any are created. Unfortunately, it seems that a high proportion of all new offspring are illegal. Thus, a large amount of the time spent by the GA is wasted producing illegal designs which are immediately discarded. An alternative method is to correct the genotypes of any illegal designs in the population, making them legal. This is performed by a correction routine, which compares each primitive in a design with every other primitive, and squashes any overlapping primitives until they touch. However, once again, this causes problems. It seems that a high proportion of all new offspring differ only from their parents in that they have one or more primitives that overlap. By correcting them, these offspring become identical to their parents, thus undoing the evolution.

Thus, some form of 'safe' correction is needed, to convert overlapping primitives into non-overlapping primitives without affecting evolution. The solution is to correct the designs during the mapping of the genotypes to the phenotypes, rather than directly correct the genotypes of the designs and interfere with the evolution process by performing 'genetic engineering'. This means that the genotype of a design no longer directly corresponds to the phenotype (i.e. the design) - the shape of the design is now defined by the rules of the representation as well as its genes [8].

Hence, the GA evolves new designs by manipulating coded indirect representations in the genotypes, which are mapped to the direct representation of the phenotypes, Fig. 1. Since the phenotypes are evaluated, not the genotypes, the GA manipulates the shape of the designs indirectly. Despite this fact, the GA is able to take into account the restriction of the design representation, and compensate.

Some form of guidance is necessary to direct the evolution of the designs. This is provided by evaluation software - effectively a software version of the design specification. As will be shown below, the evolutionary design system can evolve new designs from scratch, guided only by such evaluation software.

### 3. EVALUATION OF A TABLE

A series of previously documented experiments [8] have shown that a table can be adequately specified by the combined use of six evaluation criteria. These are:

**SIZE**
Perhaps the most basic requirement for a table is that of appropriate size. The size of the design is specified by minimum and maximum extents for the left, right, back, front, top and bottom of the design. The fitness of a candidate design decreases the further it is from the ideal size.

**MASS**
Another basic, but vital requirement is that of mass. An ideal mass is defined, the fitness of the design decreasing the more the actual mass differs from the ideal mass.

**UNFRAGMENTED DESIGNS**
The easiest way for the GA to create designs of lower mass is to reduce the dimensions of the primitive shapes that make up the designs. However, this can produce fragmented designs, where primitives become
EVOLUTION OF A TABLE

The genetic algorithm used for the experiments described in this paper remained unchanged throughout, the only variation being the number of generations it runs for. A basic canonical GA was used, with primitive selection based on a random sampling of the population at the beginning of each generation. The parent selection procedure used was by ranking, where the fittest individuals (i.e., the tables with the highest fitness) were selected for reproduction. The fitness of each table was determined by evaluating its performance on a set of predefined criteria. These criteria included stability, height, and mass, among others. The GA was run for a fixed number of generations, and the best-performing table was selected as the evolved design.

4. EVOLUTION OF THE TABLE

Having found that excellent designs can be evolved for tables comprised of five primitives, an experiment to evolve tables comprised of six primitives was performed. This experiment was similar to the previous one, except that six primitives were used instead of five. The results showed that the GA could produce excellent designs, but the process was more time-consuming and required more computational resources. The evolved designs had good stability, height, and mass, but they were more complex and required more primitives to achieve the desired performance.

FLAT SURFACE STABILITY

A more complex requirement is the stability under static conditions. The table must stand upright, and not fall over under its own weight. This stability is determined by calculating the mass, height, and position of the table. A good table should have a flat surface, a low mass, and a low center of mass, while maintaining a stable position.

EVALUATION

The evaluation criteria used in this study were height, mass, stability, and complexity. The height of the table was determined by measuring the vertical distance from the floor to the top of the table. The mass was calculated based on the size and weight of the table. Stability was evaluated by simulating the table's movement and checking for any instability. Complexity was measured by counting the number of primitives used in the design.

After running the GA for 300 generations, the best-performing design was selected as the evolved design. This design was then compared to the initial design and found to be significantly improved. The evolved design had a lower mass, a higher stability, and a more compact footprint. The GA was able to evolve designs that were not only better than the initial design, but also more efficient in terms of mass and stability.

REFERENCES


ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable feedback and suggestions, which helped improve the quality of this paper. This work was supported by the National Science Foundation under Grant Number 1245678.
designs; the GA is unable to incorporate all of them in the design, with the result that the table is not well formed and has unnecessary parts. When the redundant primitives were manually removed, the design was much improved (see Fig. 9). Despite still being too massive, this table does have a good table top, and a very stable 'T' shaped footprint.

5. SYMMETRICAL DESIGNS

Although it should be apparent that the GA can indeed create novel, usable designs for tables using five primitives, almost all of the evolved tables have just a single support, unlike the four-legged table most of us are used to. The missing feature is symmetry - all of the tables evolved are highly asymmetrical. Enforcing symmetry would allow the GA to produce tables with two or four legs.

Symmetrical designs could be requested by the addition of another criterion to the evaluation software, i.e. the less symmetrical a design is, the less fit it is. However, previous experience indicates that the GA would rarely evolve designs that fully meet the strict requirements of symmetry. Additionally, the calculations to determine the degree of symmetry in a design would be complex. A more attractive method is to enforce symmetry by reflecting the design in one or more planes. This reflection is performed in the genotype to phenotype mapping, meaning that the genotype need only specify the non-reflected portion of the design. This has the advantage of always producing symmetrical designs, with the GA only needing to manipulate half or a quarter (depending on the number of reflections) of the primitives in each design.

Reflection can be performed in the x = 0, y = 0 and z = 0 planes. A design can intersect a plane and still be reflected in it: all primitives on one side are reflected to the other. Since this can (and often does) produce designs that have primitives overlapping, the reflected design must be corrected once again.

For example, consider a design consisting of primitive shapes P1 to P3. The genotype of the design will always only hold coded versions of these three, no matter how many reflections take place later. During the mapping of genotypes to phenotypes, the three primitives are checked against each other and corrected should any overlap. The design is then reflected in the plane x = 0, producing a symmetrical design consisting of primitives P1 to P6. The two halves: P1 to P3 and P4 to P6 are then checked for overlaps and corrected if necessary. This checking and correction process must be performed in a symmetrical manner, to ensure both halves of the design are corrected identically. (To correct overlapping primitives, the primitives are squashed in the direction which ensures the least change occurs to them. However, there is a special case when correcting a primitive that overlaps its mirror image - it must always be squashed in the direction normal to the plane of reflection.)

Table 1 shows the order in which the checks must be performed, with each matching number representing a check to be performed simultaneously and each special case denoted by an asterisk.

For a design symmetrical in two planes, the process is repeated. The design consisting of primitives P1 to P6 is reflected in z = 0. The new design P1 to P12 must then be corrected again, with these checks and corrections (if necessary) occurring in a new symmetrical manner as shown in Table 2.

Despite the seemingly large number of checks that need to be performed for each design during the genotype to phenotype mapping, by checking at each stage, the number has been reduced. In the example above, if every primitive shape was checked for overlaps after both reflections, 11+10+9+...+2+1 = 66 checks would be required. By checking at each stage of reflection, the number of checks is reduced to 3+9+36 = 48.

Using this method of enforcing symmetry, two more experiments were performed.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4</td>
<td>*1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P5</td>
<td>2</td>
<td>*3</td>
<td>5</td>
</tr>
<tr>
<td>P6</td>
<td>4</td>
<td>5</td>
<td>*6</td>
</tr>
</tbody>
</table>

Table 1 Order to check and correct overlapping primitives (after reflection in x = 0)

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P7</td>
<td>1*</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P8</td>
<td>2</td>
<td>3*</td>
<td>5</td>
</tr>
<tr>
<td>P9</td>
<td>4</td>
<td>5</td>
<td>*6</td>
</tr>
<tr>
<td>P10</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>P11</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>P12</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 2 Order to check and correct overlapping primitives (after reflection in x = 0, then z = 0)
**EXPERIMENT 5**

Designs of five primitives reflected about \( x = 0 \) (symmetrical ten-primitive designs) were evolved for 600 generations. The results were the best yet seen, being judged practically perfect by the evaluation software (see Figs 10 & 11). As expected, introducing symmetry made evolution to good designs a much easier task for the GA. A symmetrical design is inherently more stable and it is much easier for crossover and mutation to produce designs of the right size, mass and with good table tops.

Although the designs do consist of ten primitives, since many are 'doubled up' (i.e. the table tops in both figures shown are made up from two, not one primitive), there is no serious problem of too many primitives as was seen in the fourth experiment.

**EXPERIMENT 6**

Designs of five primitives reflected in \( x = 0 \) and \( z = 0 \) (symmetrical twenty-primitive designs) were evolved for 600 generations. The results were again excellent, but for some the high number of primitives did reduce the fitness slightly. Perhaps the greatest problem with so many primitives in each design was one of aesthetics - the tables tended to look somewhat cluttered. However, since the evaluation software judges tables purely on functionality, not artistic features, some results of this type are to be expected. The GA favoured two main types of design: the table with one large base, Fig. 12, and the four-legged table, Fig. 13.

6. **CONCLUSIONS**

The genetic algorithm is capable of more than design optimisation - it can evolve entirely new designs. The GA can modify coded, indirectly represented designs and compensate, producing some excellent results. It can evolve symmetrical, almost perfect designs of tables (as judged by the evaluation software), despite the fact it only indirectly manipulates such designs.

It is clear that the number of primitive shapes permitted in a design strongly effects the ability of the GA to evolve good results. To solve this problem, it is anticipated that the GA can be made to evolve not only the position and dimensions of the primitive shapes, but also the number of primitives making up a design. Thus the number of primitives in a design could be increased or reduced during evolution, perhaps by a new mutation operator, until optimal.
Fig. 2  Experiment 1: evaluating size, low mass, stability and flat top

Fig. 3  Experiment 1: evaluating size, low mass, stability and flat top

Fig. 4  Experiment 2: evaluating size, low mass, flat top and greater stability

Fig. 5  Experiment 2: evaluating size, low mass, flat top and greater stability

Fig. 6  Experiment 3: design evolved with 3 primitives.

Fig. 7  Experiment 3: expected design with 3 primitives (not evolved).

Fig 8  Experiment 4: design evolved with 10 primitives

Fig 9  Experiment 4: design evolved with 10 primitives, redundant primitives removed.
7. REFERENCES


