A Voxel-Based Representation for the Evolutionary Shape Optimisation of a Simplified Beam: A Case-Study of a Problem-Centred Approach to Genetic Operator Design

P. J. Baron[1], **R. B. Fisher**[1], **F. Mill**[2], **A. Sherlock**[2], **A. L. Tuson**[1]

[1]*Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, {peterba, rbf, andrewt}@dai.ed.ac.uk*

[2]*Manufacturing Planning Group, Department of Mechanical Engineering University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, {F.Mill, A.Sherlock}@ed.ac.uk*

Keywords: Shape Optimisation, Genetic Algorithms, Voxels, Directed Mutation

Abstract

This paper examines a voxel (N-dimensional pixel) based representation for shape optimisation problems, and shows that although a basic genetic algorithm performed poorly on a simplified beam design problem, the use of three domain specific operators improved performance greatly. Additionally, the use of a 'directed smoothing' operator that preferentially adds material to high stress areas was examined and found to assist evolutionary search. This paper demonstrates how domain knowledge and an understanding of how genetic algorithms work can be used to inform the design of suitable operators.

1. Introduction

Shape optimisation within constraints is a hard problem from the field of Mechanical Engineering. The objective is to design a shape that best satisfies some predetermined goal whilst at the same time maintaining some property of the shape within a constraint, or perhaps even a set of constraints.

Previous work has applied genetic algorithms (GAs) [1] to shape optimisation problems with encouraging results. Using engineering software packages to evaluate the suitability of a given design, successful shapes have been evolved — examples include [2, 3, 4]. However, previous work on evolutionary shape optimisation has primarily been concentrated around parametric representations of structural design and shape optimisation problems which presume a family of solution shapes.

This paper focuses upon the use of a voxel (N-dimensional pixel) based representation instead, within which the shapes being optimised are represented as a series of binary 0's and 1's. This approach has the advantage that it can describe any topology, and makes no assumptions about the form of the final solution [5]. Furthermore, a voxel based representation, by virtue of its directness of representation allows domain knowledge to be easily added, and to the level felt appropriate by the designer; the following examples will help illustrate this point.

First, areas of the voxel representation can be fixed to be permanently on or off; this allows the designer to prohibit material from being placed in locations where it is not desired. A second example lies with the

ease in which existing designs can be utilised by the system — all that is required is for the initial design to be digitised and the bitmap used to initialise the population of the genetic algorithm.

Finally, taking beam design as an example, holes in the shape of the beam are entirely possible and even probable if some of the mass of the beam is occupying a low-stress area — a parametric representation can only create holes where the user is expecting them to be required and has defined the appropriate parameters.

However, [2] argues that using a binary voxel representation leads to the following problems: a long length of the chromosomes (often greater than 1000 bits); the formation of small holes in the shape; no guarantee that the final shape produced will be smooth; and even if the parents represent a valid shape, the children will not necessarily be valid. On the other hand, these objections appear to be offset somewhat by the work in [3], which has used a voxel-based approach, claiming satisfactory results. However, there appears to be little attempt to include domain knowledge into the genetic algorithm, most probably leading to the very long amount of time required to obtain these results; in this case, more than 23 hours.

This paper describes a case-study of how domain knowledge and an understanding of how genetic algorithms work can be used to inform the design of suitable operators for shape optimisation, as well as a test of the suitability of a voxel-based representation for shape optimisation problems.

2. The Optimisation of the Cross Section of a Beam

One of the problems encountered in shape optimisation, no matter what approach is taken, is interfacing the optimiser with a suitable evaluation package. For example, in wing optimisation, the wing shape has to be smooth, else the CFD (Computational Fluid Dynamics) package will act strangely: either returning negative drag, or resulting in the program crashing. Problems involving Finite Element Analysis (FEA) evaluation packages are somewhat better behaved, but interfacing is still not trivial, and the calculations do take some time.

The shape optimisation of a beam cross-section is the problem considered in this study. Evaluation of the candidate cross-sections was made using bending theory for symmetrical beams, considering only normal stresses [6]. This is a greatly oversimplified model, but sufficient to test the operators devised in this paper, as well as making repeated experimentation feasible.

Each candidate solution can thus be represented as a 2-D grid of voxels, with the optimisation objective being to minimise the mass m of the beam whilst ensuring that at all points (ie. at all voxels) in the beam the normal stress does not exceed a maximum stress (σ_{max}), which is a constant that is determined by the material to be used. This maximum stress constraint is given by Equation (1):

$$\sigma_{max} \geq \frac{-My}{I} \; for\ all\ voxels \tag{1}$$

where M is the bending moment, y the perpendicular distance of the voxel from the neutral axis, and I is the second moment of area of the cross-section. The neutral axis of a shape is defined as a line which passes through the centroid of mass of the shape. As the voxels are of uniform size and density, the centroid can be found by taking the average of the positions of all the occupied voxels. For a symmetric beam the neutral axis would be horizontal.

The bending moment M is a constant determined by the loading on the beam. The mass m of the beam is proportional to the area of the cross-section and hence the number of voxels turned on. The second moment of area is given by Equation (2):

$$I = \sum_{i \in \{all\ on\ voxels\}} y_i^2 \, dA \tag{2}$$

where y_i is the distance of the ith voxel from the neutral axis and dA is the area of a voxel (as we are dealing with a cross-section).

The optimum cross-section for a beam using this evaluation can be deduced to being a flange at top and bottom of the design domain. For the experiments described here, the following values of the constants were used: $M = 2 \times 10^6$ Nm, $\sigma_{max} = 100$ MPa, and a beam of dimensions 320×640mm was considered.

In practice, this would correspond to an I-beam, but that also requires a web to connect the two plates of the beam together. In a full calculation with shear stresses, the web would arise so to counteract this additional stress. However as shear stress is not represented in this simplified problem, a connectivity requirement in the form of a repair step was added, whereby all voxels must be connected (by a 4 connect rule) to a seed voxel in the centre top edge of the beam. In addition, a straight web was enforced before the connectivity repair step. This was found, in formative experiments, to prevent the formation of a crooked web (as the "sheer-less" physics model used does not prevent this), and improve the results obtained slightly.

3. A Basic Genetic Algorithm Implementation

This study builds upon the following basic implementation of a genetic algorithm. The encoding digitises the shape as a 32×64 grid and represents this as a 1-dimensional string of 2048 bits. Standard two-point crossover (applied with probability 0.3), and bit-flip mutation (applied with bitwise probability 0.001) were used to operate on this encoding. After the application of each operator, pixels that were not connected to the seed voxel were set to zero. An unstructured, generational population model of size 20 with rank-based selection with selection pressure 1.7 was used.

Strings are initialised to random binary bit strings with a preselected 'density' percentage; the higher this value, the more voxels are initially turned 'on'; for this study, this was set to 70%. All initial population strings must pass the validity checks used by the evaluation function: the number of active voxels must be non-zero; the second moment of area must be greater than 1×10^{-12} (goes to 0 if insufficient active voxels); and the fitness must be greater than 0.0001. The rationale behind this was to ensure that there were few instances where the lack of connected voxels would lead to blank, or sparsely filled initial solutions after the connectivity repair procedure was applied.

NOTE: as the aim of this work was to investigate whether a voxel-based representation was a feasible approach, some possible problem simplifications were not considered. One of these was that the symmetry of the beam was not exploited. Of course, full advantage of such problem features would have been taken in a more realistic and computationally expensive problem,

3.1. The Fitness Function

The fitness function was designed to minimise the area of the beam (number of active voxels) within the maximum allowed stress, with a penalty value being applied to any solution which broke that constraint. A small additional factor, $\frac{1}{S}$, was included in the fitness calculation based upon the maximum stress point in the beam. This had the effect of causing any valid solutions to continue to evolve towards better solutions (in this case a beam which minimises area and minimises the maximum amount of stress present). The fitness, F which is to be maximised, is given by Equation (3) below:

$$F = \frac{1}{V - \frac{1}{S} + k \times max\{(S - \sigma_{max}), 0\}} \tag{3}$$

where V is the number of active voxels (ie. beam mass), S the highest stress (calculated using Equations (1) and (2)) felt by any of the pixels, σ_{max} the maximum allowed stress, and k a constant which can be adjusted to vary the weight of the penalty associated with the maximum stress constraint (in this study it was set to $k = 5.0 \times 10^{-5}$).

3.2. Results Obtained

The basic genetic algorithm was found to give disappointing results. When allowed to run to convergence (over 2000 generations), the shapes produced, though recognisable as approaching an I-beam shape, were highly irregular and possessed small holes. Figure 1 illustrates this by showing the 4 best solutions from ten genetic algorithm runs. From these results it appears, at least for a simple genetic algorithm implementation, that the potential problems described in [2] do manifest themselves.

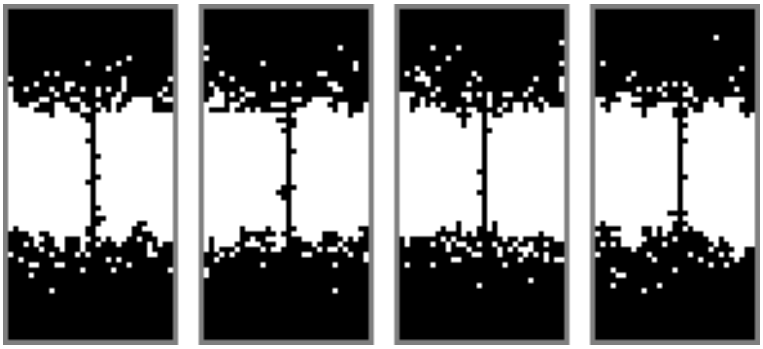

Figure 1. Typical End-of-Run Results Obtained by the Basic GA

4. An Improved Genetic Algorithm Implementation

The performance of the basic genetic algorithm was disappointing, however, many successful applications of the genetic algorithm use domain-specific operators [7]. Knowing the nature of the problems that arise with this approach, would it be possible to design operators to overcome them? With this end in mind, the basic operators were replaced with domain-specific operators in order to improve the following perceived problems:

- Crossover effectiveness;
- Removal of holes and isolated pixels;
- Removal of rough edges.

The operators are: a 2-dimensional crossover operator, a smoothing mutation operator, and a mutation operator that mutates a 2x2 area of the voxel grid. All of these operators also address one weakness in the basic genetic algorithm implementation: the encoding of a 2-dimensional problem as a one-dimensional string. Each of these operators will now be described in turn, and results of experiments to evaluate their effectiveness will be summarised.

4.1. A Mutation Operator for Smoothing

If a human designer was to examine the results in Figure 1, it would be likely that the designer would modify the design to remove the small holes and rough edges, as the designer's intuition would indicate that this would improve the quality of the beam. So one solution to the poor performance of the basic genetic algorithm would be to introduce an operation that embodied this intuition.

A mutation operator for smoothing was therefore devised. An area of x and y sizes ranging from 2×2 pixels to $\frac{1}{4}$ of the dimensions of grid was randomly selected. The most common value for the pixels in the area selected was then found, and then written to all of the pixels in that area, as illustrated by Figure 2.

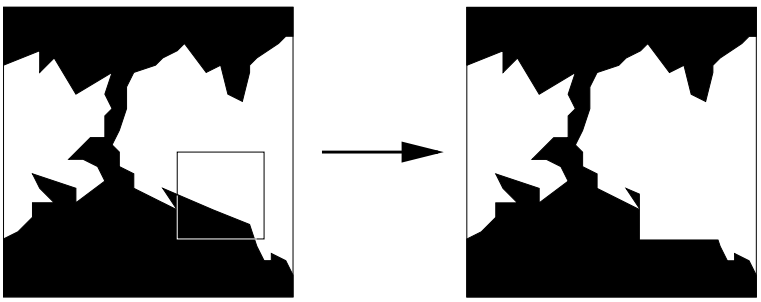

Figure 2. The Smoothing Mutation Operator

Examination of the results obtained, in a few formative runs, confirmed that this operator can remove isolated pixels with great success. With this as the only mutation operator, the shape was optimised to two near perfect horizontal bars at the vertical extremities (an I-beam) after 1000 generations.

4.2. Two-Dimensional Crossover

Another problem encountered in representing what is a 2-dimensional optimisation problem as a one-dimensional chromosome arises with *linkage*. In a 1-D encoding, voxels that correspond to spatially close points on the actual shape can be far apart on the string. Therefore, use of crossover can more easily disrupt building blocks such as one part of the shape being of high fitness, because the bits that correspond to it are spread throughout the string.

Such a situation has been encountered before in the use of a genetic algorithm to solve the source apportionment problem [8, 9]. That investigation found that representing the problem as a 2-D matrix, and devising a crossover operator (UNBLOX) that swapped a 2-dimensional section between solutions led to improved performance (Figure 3). For the purposes of this study, the block size was allowed to vary between a 1×1 grid to $\frac{1}{2}$ the x and y grid dimensions.

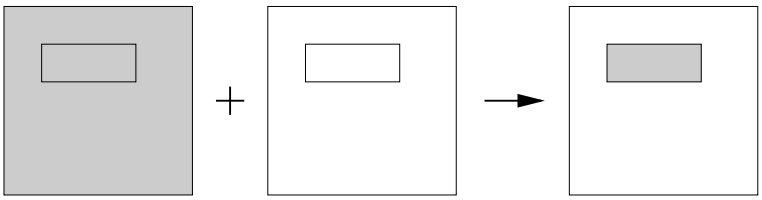

Figure 3. The UNBLOX Crossover Operator

When UNBLOX was implemented for this problem it was able to produce a recognisable I-beam after 500 generations — a large improvement in performance over the one-dimensional two-point crossover operator that shows that linkage is a significant factor in this problem.

4.3. 2×2 Area Mutation Operator

The third operator was devised for three reasons: first, the belief that as we are tackling a 2D problem, the operators should reflect this; second, a 2×2 area was thought sufficient to eliminate many of the loose pixel/wobbly-line problems found with the basic genetic algorithm; third, applying the operator only where at least one pixel is present would give a good chance of a worthwhile modification being found, and would be useful at the end of a genetic algorithm run where only small local changes would be required.

To this end, this operator acts on a 2×2 area of the chromosome array, and only modifies the contents if at least one voxel in the chosen area is turned on and one other voxel is turned off. This is so that this mutation operator would work well on the boundary of the evolved shapes. If so, standard bit-flip mutation is applied to each voxel in the area. Use of this operator increased the rate at which excess material was chopped away, whilst the ability of the genetic algorithm to continue to find improvements after convergence was also improved. In addition, the final form of the I-beam was found to be cleaner.

4.4. Putting it all Together

After the formative evaluations of each of the new operators above, it was necessary to see if these operators would work effectively in combination. Therefore, the genetic algorithm was run with the following settings: $p(UNBLOX) = 0.3$, $p(2 \times 2\ mutation) = 0.125$, and $p(Smoothing\ mutation) = 0.125$ (all probabilities are per-string); where each operator is applied sequentially to the string with these probabilities, and the sizes and locations of the operations were selected with a uniform probability. The probability of the 2×2 mutation operator was increased by 0.0005 per generation to a maximum of 0.4 as this was found to improve the quality of the results obtained slightly. All of the other genetic algorithm settings were left unchanged from the basic genetic algorithm.

Figure 4. Typical End-of-Run Results Obtained by the Improved GA

Figure 4 shows some typical end-of-run results, which are near-perfect I-beams, with no holes — a noticeable improvement over the basic genetic algorithm (Figure 1). Plots of the fitnesses against generation for the basic and improved genetic algorithms are given in Figure 6 (these figures are an average over 10 runs). As can readily be seen, the new operators have dramatically improved the performance of the genetic algorithm, finding a better quality solution more quickly than the basic genetic algorithm, thus vindicating the approach used.

The only caveat that was found was that the size range permitted for the UNBLOX and smoothing operators does interact with the amount of bending moment applied, and this can affect the results of the optimisation somewhat. This is thought to be because the size of the changes produced by these operator should be comparable to the size of the features of the final shape, which is determined in part by the bending moment. For example, if the final shape was to have thin plates at the top and bottom, then using a smoothing operator with a large size range would be very likely to remove a good section of thin plate. Fortunately, in our experience, finding a sensible range proved straightforward so long as a little common-sense was used.

5. Guiding the Search: Directed Smoothing

At this point, the operators are still applied in a somewhat undirected fashion. One manifestation of this was the observation that the smoothing operator occasionally removed material from areas that were of high stress, and added material to areas where it was not needed. Obviously, this goes against an engineer's intuition which would suggest the opposite. Therefore, a directed smoothing operator was designed to

make use of some of the useful information available to the system as a result of the evaluation function's operation. This operator was based upon a 'directed mutation' operator that has been successfully used in timetabling [10].

The evaluation must, as a necessary step to calculating the results, obtain a value for the stress present at every voxel location. The simple physics model we are using here will also calculate a close approximation to the stress present in each of the empty voxel locations as if they were solid. Thus, we can obtain an array of values representing the stress on each voxel and the average stress value for the whole array. In addition, this approach should be extensible to real a FEA package if we use material with a Young's modulus 1×10^{-3} times that of the solid material; this approach has been used successfully in [11].

This information can then be used to guide the smoothing operator in a more intelligent way. The smoothing operator was modified so that after an area has been selected, all voxels in that area with above average stress are switched on, and all voxels with below average stress are turned off. This algorithm will therefore smooth an area of voxels to match the underlying stress values according to the average value for the whole shape.

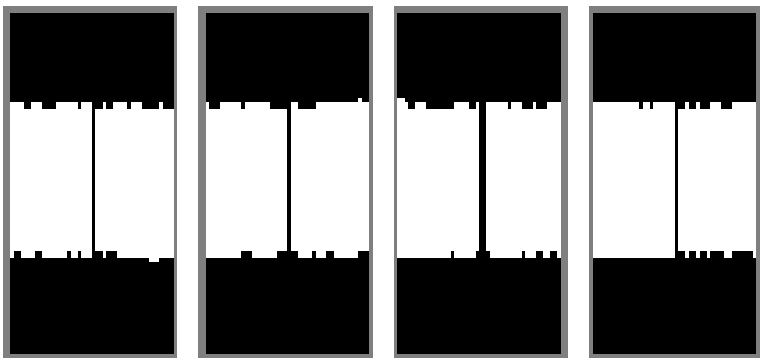

Figure 5. Typical End-of-Run Results Obtained by the GA with Directed Smoothing

The experiments in Section 4.4 were repeated with the directed smoothing operator replacing the smoothing operator. Figure 5 shows some typical end-of-run results, which again are near-perfect I-beams, therefore any fears about this operator misleading the search appear unfounded. A plot of the fitness against generation for the directed genetic algorithm, along with the plots for the basic and improved genetic algorithms for easy comparison, are given in Figure 6 (these figures are an average over 10 runs).

It is apparent that this algorithm has been found to operate extremely well, causing much faster convergence to a near-optimal solution when used in combination with the other 'improved' operators, whilst giving a solution of equivalent quality. This will be of great use when experiments using a full FEA package are undertaken, as these packages are very computationally intensive.

6. Conclusions

Results presented here show that, for the optimisation of a simplified beam cross-section, the use of a 'standard' genetic algorithm with two-point crossover and bit-flip mutation led to disappointing results — the final shape was highly irregular and possessed small holes. Three domain specific operators were then used: a 2-dimensional crossover operator, a smoothing mutation operator, and a mutation operator that mutated a 2×2 area of the voxel grid. These were designed to overcome perceived problems with schema linkage, irregular shapes and small holes.

With these operators, the genetic algorithm was able to produce shapes that were known to be optimal for this problem — thus providing some vindication of this approach and showing, contrary to previous arguments in the literature, that this approach can be used for evolutionary shape optimisation *when used in*

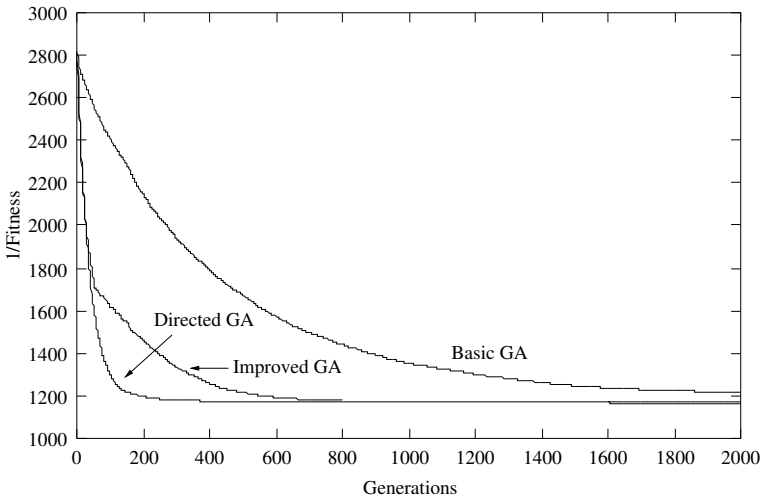

Figure 6. Fitness Plots for the Three Versions of the GA

conjunction with suitably designed operators. This is, of course, with the caveat that the problem studied here is somewhat simplified; work applying this approach to real-world shape optimisation problems is currently underway.

Finally, a 'directed smoothing' operator that adds material to high stress areas was examined to ascertain the degree that this form of domain knowledge can assist evolutionary search, and was shown to speed up search significantly.

In summary, this paper describes a case-study of how domain knowledge and an understanding of how genetic algorithms work can be used to inform the design of suitable operators, as well as demonstrating the suitability of a voxel-based representation for a simple shape optimisation problem.

Acknowledgements

Thanks to the Engineering and Physical Sciences Research Council (EPSRC) for their support of Andrew Sherlock and Andrew Tuson via studentships with references 95303677 and 95306458. The authors would also like to thank the reviewers for their relevant and useful comments.

References

[1] Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

[2] Watabe, H., Okino, N., 1993, A Study on Genetic Shape Design,*Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois, USA, July*, pp. 445-450.

[3] Chapman, C. D., Saitou, K., Jakiela, M., 1994, Genetic Algorithms as an Approach to Configuration and Topology Design, *Journal of Mechanical Design*, **116**, 1005-1012.

[4] Husbands, P., Jeremy, G., McIlhagga, M., Ives, R., 1996, Two Applications of Genetic Algorithms to Component Design, *AISB Workshop on Evolutionary Computing, Sussex University, UK, April*, pp. 50-61.

[5] Smith, R., 1995, *A First Investigation into a Voxel Based Shape Representation*, Manufacturing Planning Group, Department of Mechanical Engineering, University of Edinburgh, UK.

[6] Gere, J. M., Timoshenko, S. P., 1984, *Mechanics of Materials 2e*, Brooks/Cole Engineering.

[7] Davis, L., 1991, *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold.

[8] Cartwright, H. M., Harris, S. P., 1993, Analysis of the distribution of airborne pollution using genetic algorithms, *Atmospheric Environment*, **27**, 1783-1791.

[9] Cartwright, H. M., Harris, S. P., 1993, The Application of the Genetic Algorithm to Two-Dimensional Strings: The Source Apportionment Problem, *Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois, USA, July*, pp. 631.

[10] Ross, P., Corne, D., Fang, H.-L., 1994, Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation, *Parallel Problem-solving from Nature — PPSN III, Jerusalem, Israel, September* pp. 556–565.

[11] Bendsoe, M., Kikuchi N., 1988, Generating Optimal Topologies in Structural Design using a Homogenization Method, *Computer Methods in Applied Mechanics and Engineering*, **71**, 194-224.