Multi-material Compositional Pattern-Producing Networks for Form Optimisation

Ralph Evins[1,2](✉), Ravi Vaidyanathan[3], and Stuart Burgess[4]

[1] Empa, Swiss Federal Laboratories for Materials Science and Technology, Überlandstrasse 129, 8600 Dübendorf, Switzerland
ralph.evins@empa.ch

[2] Chair of Building Physics, Swiss Federal Institute of Technology ETH Zürich, ETH-Hönggerberg, 8093 Zürich, Switzerland

[3] Imperial College London, South Kensington Campus, London, UK SW7 2AZ

[4] University of Bristol, Tyndall Avenue, Bristol, UK BS8 1TH

Abstract. CPPN-NEAT (Compositional Pattern Producing Networks and NeuroEvolution for Augmented Topologies) is a representation and optimisation approach that can generate and optimise complex forms without any pre-defined structure by using indirect, implicit representations. CPPN is based on an analogy to embryonic development; NEAT is based on an analogy to neural evolution. We present new developments that extend the approach to include multi-material objects, where the material distribution must be optimised in parallel with the form.

Results are given for a simple problem concerning PV panels to validate the method. This approach is applicable to a large number of problems concerning the design of complex forms. There are many such problems in the field of energy saving and generation, particularly those areas concerned with solar gain. This work forms a first step in exploring the potential of this approach.

Keywords: CPPN · NEAT · Form · Multi-material

1 Introduction

1.1 Engineering Form Optimisation

Form is used here to refer to the physical shape of an object, and form optimisation refers to the process of finding optimal or high-performing forms for engineered objects, measured against some metric. Optimisation of form is more challenging than optimising specific design parameters, as form may be represented in many ways, making the design space almost infinite. This paper presents new developments to a systematic way of automatically generating and evolving forms to find areas of optimal performance.

1.2 Form Representation

Form representations can be divided into two approaches. Direct representations relate the set of variables being optimised (the genotype) to the associated form (the phenotype) in a way that is constant throughout the optimisation process. Classes and types of form are broadly defined at the beginning of the process through the choice of a particular representation (shapes or mathematical functions) with a fixed number of parameters (also called degrees-of-freedom). The representation used implicitly affects the forms generated: some forms will be completely unobtainable, and others will be complicated to describe and therefore difficult to discover. Direct representations are appropriate for simple optimisations where the design space is limited to an easily-describable set of forms. Imam [4] discussed a variety of direct representation types, including independent nodes, design elements, super curves and superposition of shapes.

Indirect representations, by contrast, use a mapping from genotype to phenotype that changes as part of the form-finding process. Indirect representations have no predefined set of parameters. Instead they generate a means of representing a form in parallel with the parameters that define it. Types of indirect representation include generative (generating functions that map parameters to forms) and ontogenic (based on iterative mapping transformations). For generative representations, Bentley and Kumar [2] used the term embryogeny: the process of growth that defines how a genotype is mapped onto a phenotype. They discuss three types of generative encoding: external (pre-stated, a form of direct representation), explicit (inherent in the data structure, like a list of instructions) and implicit (interactive, dynamic rules that depend on context). Bentley and Kumar found that for the problem they selected, implicit embryogenies performed best.

Another approach to indirect form representation is the related field of topology optimisation. This is concerned with broad classes of shape (e.g. number of sides, number of holes). It uses a discrete selection field over a fixed domain, analogous to the discrete voxels used in this work. An objective function is minimised over this selection field using a variety of methods, for example the Evolutionary Structural Optimisation approach [8] progressively eliminates low-stress material from the structure. The approaches used in topological optimisation are tightly linked to structural engineering issues, and are not easily adaptable to problems in other fields, especially if analytical objective functions are not available (i.e. when using black-box simulations).

The indirect representation used in this work is Compositional Pattern Producing Networks (CPPN) with the optimisation method NeuroEvolution for Augmented Topologies (NEAT), which are explained in detail in the following section. CPPNs were proposed by Stanley [6]; NEAT was originally developed by Stanley and Miikkulainen [7]. This work builds upon that of Clune and Lipson [3], who developed a 3-dimensional formulation of CPPN-NEAT. CPPN-NEAT has been used on few real problems: to interactively generate artwork, as demonstrated in the website picbreeder [5], and to evolve forms for simulated robots [1].

1.3 Application to the Energy Field

It has been established by Clune and Lipson [3] that CPPN-NEAT can generate a diverse range of interesting forms. The methods developed here could be applied to any problem which seeks to optimise abstract forms for objectives evaluated using black-box simulations. There are many such problems in the field of energy research; one particularly relevant area is energy use in buildings, where architectural desires closely interact with engineering requirements.

This paper applied the method developed to a problem concerning a photovoltaic (PV) collector. It is a very simple validation problem, which seeks to establish whether the breadth and diversity of solutions produced by CPPN-NEAT can produce reasonable answers to a specific problem. However, if combined with other constraints, for example the problem of building-integrated PV, this approach could provide a way to find high-performing, highly diverse forms that solve a real design problem.

2 Form Generation Method

2.1 Compositional Pattern Producing Networks (CPPNs)

Compositional Pattern Producing Networks (CPPNs), proposed by Stanley [6], are based upon the biological processes that guide embryonic development: chemical gradients provide information to new cells regarding their position in the overall structure, which influences how they develop. Stanley [6] details the following desirable properties obtainable via such developmental processes: repetition; repetition with variation; symmetry; imperfect symmetry; elaborated regularity; preservation of regularity.

The steps in the CPPN process is given below. The predefined coordinate system is discretised at a chosen resolution over a chosen domain, and the value of a function is calculated for every point x, y, z. The presence or absence of material at a given location is determined by whether the output of that function is above or below a threshold. For an x, y, z coordinate system, the result is a set of voxels (3 dimensional pixels). Further processing may then be conducted to obtain a smooth form from the rectilinear cubic voxels. In this work, an isosurface was generated surrounding the voxel set: each point where a voxel is present has a value of 1, and points where no voxel is present have a value of 0; the isosurface was formed for the value 0.5. The resulting surface consists of triangular planar faces.[1]

[1] It is necessary to threshold the output of the CPPN, which is a continuum across the complete x, y, z domain, in order to produce a binary distinction between solid and void. Because the function must be evaluated at discrete points, this results in a set of voxels whose dimensions correspond to the sampling interval. These must then be smoothed using an appropriate method (here the MatLab isosurface algorithm) to obtain planar faces. The impact of threshold value, sampling interval and smoothing process is an interesting topic for future investigation.

- For n points in the discrete domain:
  - Evaluate network using coordinates x, y, z as values of input nodes.
  - If result is greater than threshold, assign solid voxel to set V.
- (Apply smoothing algorithm to set of voxels V to get surface of polygons P.)
- Evaluate objective function $f(V)$ (or $f(P)$).

This process of form generation depends on a functional representation that takes a set of coordinate values as an input, and produces an output that governs the form produced. Neural networks are an ideal means of representing such a function. Each coordinate dimension is an input node to the network, along with a bias node that is set to 1. Each link in the network has a weight by which its value is multiplied. Each intermediate node has a functional transformation associated with it, selected from a set of available functions (linear, sine, cosine, square...). If there are multiple links into a node, their values are summed. The output node of the network then produces the numerical output of the function.

An illustration of the process is given in Figure 1, which extends into 3D the example used by Stanley [6] and Clune and Lipson [3], describing by means of a CPPN an insect body with several bulbous sections. For simplicity, each dimension is used by one function only, and the results are then summed. The square function is used on the dimensions x and z, thus giving a circular cross section when these are summed (since a circle in that plane has an equation of the form $x^2 + z^2 = r^2$). The cosine function is applied to the y dimension, causing periodic repetition along the long axis.

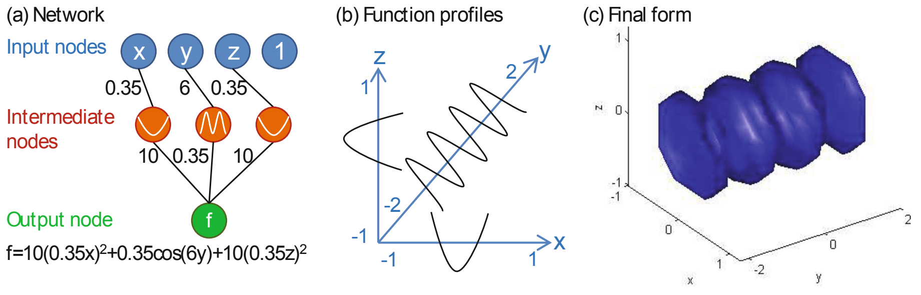

Fig. 1. Example of the CPPN process. (a) Network of nodes, connections, functions and weights. (b) Profiles obtained for each dimension based on the functions used in the network. (c) The final form produced by the network is an isosurface fitted to the set of voxels defined by $f(x, y, z) > threshold$.

2.2 NeuroEvolution of Augmenting Topologies (NEAT)

Since CPPN describes a form by means of a particular network (nodes, connections, functions and weights), in order to evolve object forms, it is necessary to evolve network representations. The method used here is NeuroEvolution for

Augmented Topologies (NEAT), originally developed by Stanley and Miikkulainen [7] for evolving neural networks but also commonly applied to CPPN problems. The steps are given below

- Initialise network (random connections and weights).
- Assign species (used in selection process).
- Evaluate CPPN (see above).
- For each generation:
  - Check for stagnation or refocus.
  - For each space in new population:
  - Select parents (based on shared fitness of species).
  - Generate new individual by crossover (splice networks) and mutation (Add node, add connection, change function, perturb weight).
  - For each individual in new population:
    - Assign species.
    - Evaluate CPPN.
  - Select individuals to continue.

The method begins with a very simple network (just input nodes, bias node and output node, directly connected) and increases its complexity by adding connections and nodes, mutating connection weights and node function types, and crossing over network segments. The number of input nodes is equal to the number of dimensions of the CPPN. This may be 2D, 3D or include other possibilities like distance from centre. There is only one output node. Recurrence in networks is not used at all in this work (it was found by Clune and Lipson [3] to produce highly fractal forms). The process of crossover is complicated in variable-structure representations: it is necessary to know which segments of two individuals can be interchanged without breaking connectivity or introducing spurious deformity. This is achieved in NEAT by means of historical innovation tracking: each alteration to a network is recorded, and this historical information allows only correctly-aligned network segments to be exchanged (a process termed artificial synapsis, see [7]). The parameters used are given in Table 1.

2.3 Implementation

The NEAT code used in this work is loosely based on the MatLab implementation by Christian Mayr[2], which was based on the original C++ code of Kenneth Stanley[3]. An improvement here is to construct an explicit function string that is evaluated very easily for each set of inputs. The function string was constructed iteratively by substituting placeholders for upstream nodes, working backwards from the output node.

This approach to form optimisation require a very large number of function evaluations (up to 150,000 evaluations per run). The code was run on the University of Bristol Advanced Computing Research Centre machine BlueCrystal.

[2] http://nn.cs.utexas.edu/?neatmatlab
[3] http://nn.cs.utexas.edu/?neat_original

Table 1. Parameters used for NEAT algorithm

Maximum generations		*10000*	[3]	Refocus	Threshold	*10*	
Population size		*15*	[3]		# generations	*100*	
Selection	Pressure	*1.1*		Mutation	Add node	*0.25*	[3]
	Kill percentage	*0.2*	[7]		Add connection	*0.3*	[6]
	Number kill	*5*	[7]		Change function	*0.1*	
	Number copy	*1*	[7]		Perturb weight	*0.9*	[6]
Speciation	Threshold	*4*			Gene re-enabled	*0.25*	[6]
	C1	*1*	[6]		Weight cap	*-100*	
	C2	*1*	[6]		Weight range	*-10*	
	C3	*0.4*	[6]	Crossover	Overall	*0.75*	[6]
	C4	*2*			Interspecies	*0.001*	[6]
Stagnation	Threshold	*10*			Multipoint	*0.75*	[6]
	# generations	*15*	[7]				

In order to minimize the need for parallel-specific coding, each optimisation run was split across 8 local cores using the Matlab parfor syntax for parallel loops. Separate processors were used for each repeat of a run.

3 Multi-material Formulation

3.1 New Development

The new development presented in this work is the extension of CPPN-NEAT to multi-material forms. Engineered objects usually consist of more than one material, and the interactions between them can have a significant effect on performance. This makes it difficult to determine the optimal placement of each material independently; they must be developed in harmony to take advantage of synergies between them. It is highly desirable that the two materials together should make up the whole form (no holes) and nothing else (no dislocations) so as not to affect the performance of the form-finding process. The CPPN-NEAT method has been extended to allow the evolution of separate material placements in parallel with the overall form-finding process. This has been applied to thin shelled forms, assuming a hollow object consisting of triangular planar panels.

Material placement could be optimised using a lower level optimisation process, i.e. for each proposed form, a second-level optimisation would be performed to determine the optimal placement of the materials. However, this would be computationally much more demanding: if the form optimisation is order O, performing an optimisation of material placement for every evaluated form would be of order O^2. It is much more efficient to optimise both the form and the material division in parallel as part of the same optimisation, this being of order $2O$. This has been achieved by evolving two CPPN representations using a single NEAT loop (separate NEAT processes were used for each CPPN to allow different parameters for each, but both used a common generation iteration). The first network represents the overall form as before; the second network maps the placement of materials onto the form generated by the first network. The second

CPPN is queried only at locations where material is present (as determined by the first CPPN). If the output value from the second CPPN is greater than the threshold it indicates the primary material; if it is less than the threshold it indicates the secondary material. In this way since the network provides a value for all solid locations and no others, holes and dislocations are avoided.

For thin shell objects a mapping has been used that operates on the polygon mesh rather than on the voxel set. This allows the use of local coordinates: the orientation and inclination of each polygon. This permits changes of material between horizontal and vertical, North and South facing and top and bottom sides, independently of position in the overall form. Figure 2(a) shows this mapping: the solid voxels from the form CPPN are used to produce a shell consisting of the set of polygons P, for which the orientation θ and inclination γ values are determined; these are used as the input coordinates to the material CPPN, which provides the primary and secondary material polygons P_1, P_2.

Instead of distinguishing between two discrete material types, some property of the material can be treated as continuously variable. This could correspond to thickness, reinforcement, void ratio, glazing ratio etc. The process for this is very similar to above, but rather than applying a threshold to the output of the material CPPN to give a binary choice, the value is scaled from 0 to 1 to provide the continuous property M for each polygon. This is presented as a second option of the new development (see Figure 2(b)).

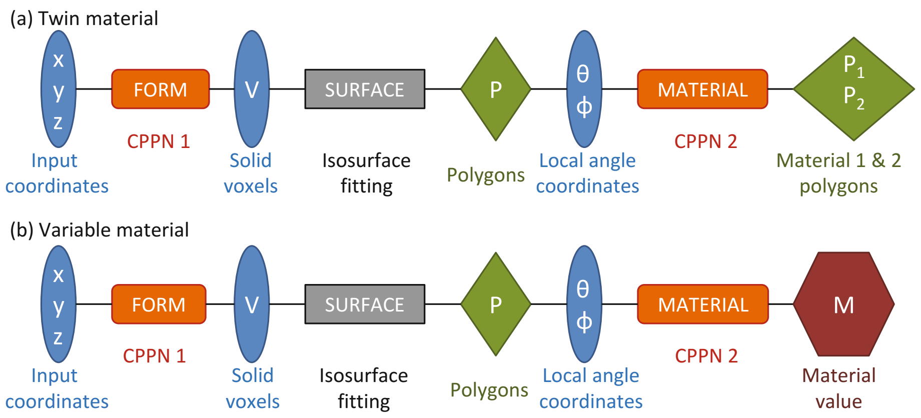

Fig. 2. Process diagrams for using two CPPNs to determine form and material distribution in parallel, for (a) two-material forms and (b) variable-material forms. CPPN 1 produces voxel set V; an isosurface is fitted to V to give polygon set P; CPPN 2 is applied to P to find material division P_1, P_2 or continuous property M.

3.2 Objective Functions

An example problem is used to validate the multi-material formulation in which the second material represents PV panels, and the objective function takes the ratio of energy generated to total cost. Two different options were addressed.

The first option assumed a discrete material distribution, using the formulation in Figure 2(a). A highly simplified calculation is used for the energy generation potential, which was taken as proportional to the total area of PV polygons that are within 55 degrees of South and with an inclination of 0 or greater (i.e. not angled downwards). Cost is taken as the sum of the total area of PV panels multiplied by a price factor (here 10, i.e. the PV panel cost is ten times that of the support), plus the total area of both materials (i.e. the support system). Thus the objective function was:

$$\frac{\sum P_2|_{\theta>125,\,\theta<235,\,\gamma\geq 0}}{10\sum P_1 + \sum(P_1 + P_2)} \tag{1}$$

where θ is the angle of orientation of the polygon from north, γ is the angle from horizontal, P_1 is the area of support polygons, and P_2 is the area of PV polygons. For a proper analysis of the energy generated from the PV panels, a more detailed simulation would be necessary. This could be using a table lookup for different angles, if self-shading is ignored, or using a detailed ray tracing simulaton, if self shading is important. Both are beyond the scope of this paper, where the aim is to validate the new material representation with a very simple case.

The second option assumed a variable-material property using the formulation in Figure 2(b), taking the percentage of a polygon surface covered by PV to be a continuous variable. The generation from a polygon was determined by the percentage of PV (the property M) and the cosine of the angle between the polygon normal vector and the optimum alignment vector for the chosen latitude (here taken to be South, 45 degree inclination). Thus the objective function was:

$$\frac{\sum P_2 M \frac{\sqrt{2}}{2}\left(\sin(\theta) - \cos(\gamma)\right)}{10\sum P_1 + \sum(P_1 + P_2)} \tag{2}$$

4 Results

4.1 Two Materials

This case demonstrates two things: that the CPPN-NEAT method can produce forms that respond to the optimisation objective, and that the two-material formulation can also adjust the material distribution in accordance with the objective. Figure 3 shows the final forms from all twenty runs of the two-material case, ordered by fitness value. It is clear that a very wide range of forms can be produced. Nuances of the objective function become apparent, such as the way the isosurface fitting to the voxels affects the range of angles of polygons.

Fitness values ranged from 76.7 to 81.3 with a mean of 79.0 and a standard error of 1.4. The objective can be split into the following components, in rough order of priority: maximise the area of PV panel that is broadly south-facing; minimise the area of PV panel that does not meet the above conditions; minimise the total surface area. There are a number of different approaches evident in the

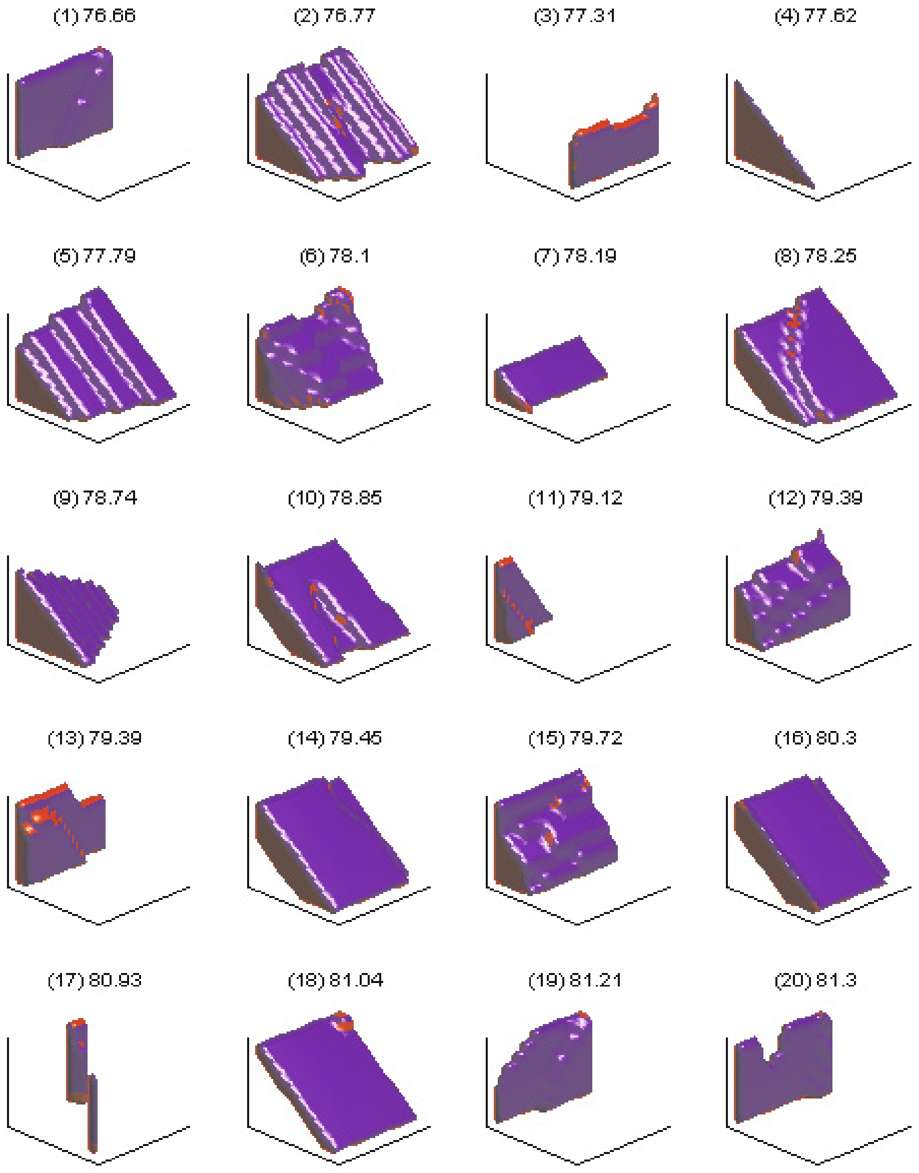

Fig. 3. All final optimised two-material forms with fitness values. Support is red, PV panel material is blue.

solutions found. The greatest total south-facing area is given by an inclined plane, either as a pyramid (solutions 4, 9) or wedge (2, 5, 7); the total surface area can then be reduced by making it thinner (8, 10, 14, 16), culminating in making it as thin as possible (1 voxel) (18). The greatest broadly south-facing area per total surface area is given by a section of a sphere (6); this may be approximated by a section of a cylinder, aligned either horizontally (12, 15) or vertically (17). The simplest form with reasonable performance is a thin obloid (1, 3, 11, 13, 19, 20). Because the PV panels need not face exactly south, there is scope to increase the surface area by adding undulations (2), steps (5, 8, 9, 12) and bulges (1, 10, 19, 20). Similarly the area of the top surface can be increased by including slopes (19) or dips (20). There was no limitation on single-block forms for this problem. However, generally dividing a form adds extra material without increasing south-facing area, and whilst multiple high-performing forms would maintain high fitness they would be likely to require more complex representations than single forms. Only one run resulted in a multiple block solution (17).

Material distribution is clearly adapting to the objective of the optimisation. All forms have the PV panel material predominantly on the south-facing side only none of the forms have any significant PV panel material on the rear or under sides (not shown). There is some variation in how well the PV panel covers the south face, with some obvious gaps (2, 8 10, 15) and missing upper edges (3, 11, 13). In general the material placement errors are low: on average across the 20 solutions, 1.8% by area has ‘missing’ PV (would fit the criteria but not present) and 0.9% incorrect PV (does not meet the criteria). It is interesting to note that the high-fitness solutions (nearer to number 20) do not have notably lower material placement errors (e.g. the highest error of 7.2% is for solution 18), although the errors are more likely to be in missing south-facing area rather than erroneous non-south-facing areas. There is clearly a balance between the performance of the form and the accuracy of the material distribution.

4.2 Continuously-Variable Material

The second option, to optimise a continuously-variable material parameter, is a more challenging and subtle problem. Because the angle of the surface relative to the average sun position is taken into account in calculating energy generated, it is now more important that the PV should face directly south at 45 degree inclination. This eliminated the curved surfaces from the previous case. Figure 4 shows selected forms from the continuously-variable material option. These examples cover all the forms found: there were three low fitness forms like (1), fourteen mid fitness forms like (2), and the unique forms (3, 4, 5). Fitness values ranged from 54.7 to 78.0 with a mean of 68.9 and a standard error of 5.5. The types repeat many those of the previous section: horizontal cylinder (2), wedge (4) and thin angled plan (3, 5); there is also the notably low-performing horizontal plane (1) where the algorithm was unable to progress beyond the simple plane. This occurred in 4 out of 20 runs, whereas there are no such low-performing solutions in the previous case. Additional complexity is introduced

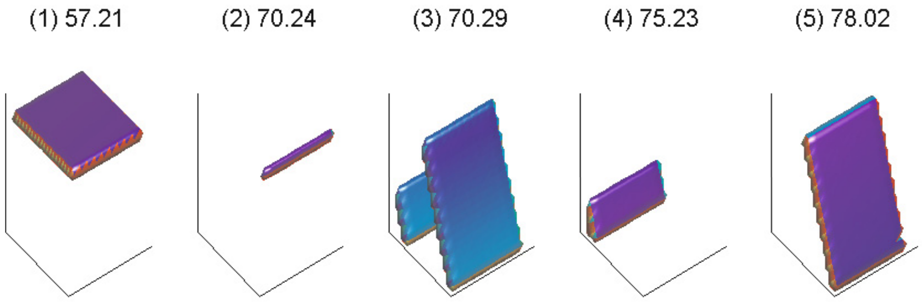

Fig. 4. Selected optimised variable-material shell forms with fitness value

to the problem by the variable-material formulation, which appears to be preventing the algorithm finding good solutions in some cases. The distribution of the PV is sometimes almost binary (1, 2, 4), where the material mapping falls almost entirely to one end of the scale or the other. This is to be expected, as it provides solutions with reasonable performance that have very simple material distributions and are therefore very easy for the algorithm to find. The distribution in form (3) is a gradual progression, with low PV ratio at the bottom and high at the top, demonstrating that the formulation is able to produce this sort of distribution. The highest performing solution (5) uses a precise distribution in which the main face has a high PV ratio (100%), the top edge (at an angle to the sun that is sizeable but less than 90 degrees) has an intermediate ratio, and the rest has a ratio of zero.

5 Conclusions

The multi-material implementation was demonstrated on an example problem concerned with PV panels. The algorithm was successful, generating solutions that combine high-performance forms with appropriate material distributions. For the discrete problem, the diversity of solutions found across 20 runs was large, highlighting the range of shapes and distributions obtainable. This also included solutions that exploited aspects of the process, for example using curved edges to increase surface area. For the continuously-variable problem, there was a greater range of fitness values, showing that the algorithm sometimes fails to find good solutions. It is inevitable that the continuous problem will be harder, but future work could investigate how to overcome this barrier, perhaps by approximating the gradient as bands.

The algorithm is exploring a very large search space, and the great variety of solutions make it useful to compare several runs of the algorithm rather than to take only one solution as indicative. Form optimisation problems are by nature very complicated, and problems may not be solved completely. The algorithms developed can be used to guide the design process, but are unlikely to generate a perfect result.

There are clearly many simpler ways of approaching the problem of form optimisation, especially for the problems examined here. However, the method used offers the potential for breadth and adaptability: the range of forms available is limitless. On that basis, the initial demonstration of the method has been satisfactory: from the vast realm of possible configurations, finding solutions to conceptually simple problems is not trivial. This paper is the first step in developing this approach to indirect form representation and optimisation into a usable method. Future work will extend the application to other cases where complex forms are required, for example the trade-off between winter solar gain, summer solar gain and light availability in passive building design.

Acknowledgments. Funded by the Industrial Doctorate Centre in Systems, Universities of Bristol and Bath, UK (EPSRC Grant EP/G037353/1) and Buro Happold Ltd, UK.

References

1. Auerbach, J.E., Bongard, J.C.: Evolving complete robots with CPPN-NEAT: the utility of recurrent connections. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 1475–1482. ACM, New York (2011)
2. Bentley, P., Kumar, S.: Three ways to grow designs: A comparison of evolved embryogenies for a design problem. In: Genetic and Evolutionary Computation Conference, pp. 35–43. Morgan Kaufmann (1999)
3. Clune, J., Lipson, H.: Evolving 3D objects with a generative encoding inspired by developmental biology. SIGEVOlution **5**(4), 2–12 (2011)
4. Imam, M.H.: Three-dimensional shape optimization. International Journal for Numerical Methods in Engineering **18**(5), 661–673 (1982)
5. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T., Stanley, K.O.: Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation **19**(3), 373–403 (2010)
6. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines **8**(2), 131–162 (2007)
7. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2), 99–127 (2002)
8. Xie, Y.M., Steven, G.P.: A simple evolutionary procedure for structural optimization. Computers & Structures **49**(5), 885–896 (1993)