Evolving Design Modifiers

Simon Hickinbotham*, Rahul Dubey*, Imelda Friel†, Andrew Colligan†, Mark Price†, and Andy Tyrrell*
*Intelligent Systems & Robotics Research Group, School of Physics, Engineering and Technology
University of York, UK
†School of Mechanical & Aerospace Engineering, Queen's University Belfast
Email: *simon.hickinbotham, rahul.dubey, andy.tyrrell(@york.ac.uk), †I.Friel, A.Colligan, M.Price(@qub.ac.uk)

***Abstract*—Evolutionary Developmental biology (EvoDevo) is a process of directed growth whose mechanisms could be used in an evolutionary algorithm for engineering applications. Engineering design can be thought of as a search through a high-dimensional design space for a small number of solutions that are optimal by various metrics. Configuring this search within an EvoDevo algorithm may allow developmental processes to provide a facility to give more immediate, localised feedback to the system as it grows into its final optimal configuration (form). This approach would augment current design practices. The main components needed to run EvoDevo for engineering design are set out in this paper, and these are developed into an algorithm for initial investigations, resulting in evolved neural network-based structural design modifying operators that optimise the structure of a planar truss in an iterative, decentralized manner against multiple objectives. Preliminary results are presented which show that the two levels feedback at the Evo and Devo stages drive the system to ultimately produce feasible solutions.**

***Index Terms*—genetic algorithms, neural networks, evodevo, generative design, structural engineering**

I. Introduction

The design of engineering solutions seeks to select a small number of near-optimal designs from a huge but finite space of possible designs. The traditional approach is manual (often with the assistance of complex simulators) and reliant on individual or team expertise, resulting in a time consuming and highly involved process. Recent advances in computation allow more complex algorithms to be developed which can explore the vast search space in support of designers to expedite this process and improve design outcomes. Although there have been some attempts to use evolutionary algorithms to search these design spaces [1]–[5], these approaches tend to be reductive, gradually restricting the design space through exploration of topology, shape and size in turn. They do this with good reason: the total search space is vast, and feedback from generation to generation via a fitness function (even a multi-objective fitness function) may not be sufficiently detailed to direct the search. The issue with this reductive approach is that once the domain is defined and a small set of viable structural topologies is chosen, any further optimisation of shape and size is limited to perturbations on this original set. Generative design systems build solutions using a bottom up approach where there is more opportunity to explore a design space, and critically, to allow that space to expand as more is learned about the design. Price et al. [6] show how a biological development analogy can be usefully applied in a generative design system for the creation of a single component. There is significant potential to combine such development processes with evolutionary algorithms to offer new potential in engineering design.

The model application domain of the work reported here is that of optimising the design of planar trusses. Deb and Gulati [1] approach the design problem from the perspective of canonical genetic algorithms with fixed-length encoding. To achieve this, only a small number of joint positions is permitted for any given design problem, and the algorithm searches for connections between this set of joints. Whilst good solutions are found within this framework, the design space is too constrained to find a true optimum. In addition, the single-objective optimisation incorporates several penalty constraints which must be manually tuned to obtain good performance. More recently, Tejani and co-workers [3] explored a range of metaheuristic algorithms, using a single objective function and optimising node position and member thickness to evolve structures with the least weight that do not violate other performance constraints. These metaheuristics augment the core evolutionary algorithms with other techniques, but a development stage is not tested.

This contribution describes a new approach to engineering design, inspired by concepts originating in the study of evolutionary development (EvoDevo). In biological EvoDevo, the development of an organism from a single cell to its final (adult) form is governed by a gene regulatory network (GRN), which turns genes ON and OFF during different stages of development in a co-ordinated manner, without reference to a global controller [7]. Algorithms inspired by EvoDevo aim to augment an evolution component (Evo) with a development component (Devo). For a good overview of the issues in EvoDevo see [8] and for a review of the relation between generative design and evolution see [9]. Wolpert's french flag model (FFM) [10], [11] was amongst the first Devo algorithms: the task was to colour in a rectangular field of cells of arbitrary size with red, white and blue in equal proportions, as seen in the national flag of France. In the absence of central control, the algorithm does not contain explicit instructions regarding what colour a cell should be. Instead, the decision about colouration of a particular cell is made by reference to a gradient of signal strengths emanating from either end of the tricolore. The FFM is informative in terms of the principles of using development towards achieving a particular

This work was funded by EPSRC, UK, grant reference EP/V007335/1

design goal, but it makes no reference to external physical conditions and it does not specify how the GRN can evolve. These aspects are key when applying the methodology to real engineering design problems, because evolution is needed to discover and fine tune the appropriate developmental strategy on the growing design as it interacts with its environment. Navarro et al discuss the challenges of implementing EvoDevo for design problems [12]. They use the Devo concept to drive the production of a wide range of coloured shapes, but whilst the process is efficient, an evolutionary component has not yet been implemented. Walker et al [13] try a similar idea with robots, but the growth routine is fixed and evolution is only applied to the weights of the neural network controller: there is no mechanism to update the topology of the network.

In the current work, the design challenge is re-cast as one of selecting an appropriate developmental pathway for a growing entity whose structure is iteratively updated until the final design is arrived at. In a developing system, the concept of "growth" covers a number of options which change the structure: through a change (normally an increase) in a key parameter such as cross-sectional area, or material density; via change in the positioning of joints in the structure, which changes the overall size of the structre without adding new structures; or via the addition of new cells to the developing entity. These options emulate growth, spatial positioning, and division of cells in biological systems. As a proof of concept, the work presented here focuses on the positioning of the joints, but the process is readily extendable to both of the other structural changes just described.

In order to apply the EvoDevo model to engineering design, the following components are needed: a configurable model of a 'seed' engineering structure which can be decomposed into one or more 'cells'; a means to generate or simulate forces on the cells in the model via the complete structure; a methodology for converting these forces to updates via a GRN within each cell; a method of formulating a consensus on the GRN outputs from each cell that can be used to update the structure; a means of measuring the fitness of the final form; an evolutionary algorithm that can use fitness values to select and mutate the genome from which the GRN is constructed.

In order to meet these challenges, a novel EvoDevo algorithm for applied engineering is described below. The three key components are a decomposition of an engineering structure into a cellular representation, feedback to the Devo stage via simulation of the current state of the system, and the emulation of GRN function to direct the modifications to the design. The model application domain was chosen such that the cellular representation could be mapped to obvious sub-assemblies in the structure, namely triangular arrangements of members in a truss structure. These structures can be described simply, and simulated in software-based Finite Element Analysis (FEA) to measure forces acting on the structure for feedback to the GRN. The function of the GRN is emulated via a multi-objective implementation of NeuroEvolution of Augmenting Topologies (NEAT) [14]. The latter allows GRNs of sufficient complexity to be evolved to generate appropriate Devo for a given engineering task.

The advantage of incorporating a Devo stage into the process is that it provides a facility to calculate and incorporate simulated physical forces into the discovery of a design solution. The update process is akin to Lomas's generative art [15], [16], but the way it accesses real physical properties is akin to Adrian Thompson's direct encoding with real signals via Field Programmable Gate Aarrays (FPGAs) [17]. The Devo stage gives the system multiple opportunities to explore its response to local physical conditions during growth. Indeed, in the current implementation, the physical forces on the system act as a proxy for the enzyme signalling gradient used in Wolpert's French Flag mode and the "state" of the cell/GRN subunit.

The cells in biological organisms all share the same genome, but are able to express different genes depending on the current state of the cell. It is unclear how best to implement this in the prototype system for design. In this initial configuration, experiments were devised to verify whether the local conditions pertaining to each cell in the structure drive the evolution of appropriate non-linearities in the NEAT network to emulate switching in GRNs and thus generate appropriate updates to the truss design.

The main novel contribution highlighted in this paper is an algorithm that drives the evolution of an appropriate gene regulatory response to physical simulation, which delivers improvements to the design of the structure without manual supervision. The resulting algorithm evolves neural network-based structural design modifying operators that can adjust the structure of a 2D planar truss in an iterative, decentralized manner, inspired by structural/physiological adaptation in a biological organism.

II. Methods

The EvoDevo technique discussed here is best approached from the perspective of an individual design specification, through its development into its final form, and then how populations of such forms are evolved into the final designs. The way the components of this process are combined into a novel algorithm is described at the end of this section, along with details of the experimental design.

A. Fitness in Static Truss Design

In order to evaluate the approach, a 'Goldilocks' application framework was devised: not so simple that the algorithm has limited scope for improvement, but not so complex that the results cannot be interpreted straightforwardly. It was also important that the design problem had physical parameters that can be easily simulated and understood. For these reasons, a classic engineering design problem was selected: load-bearing static trusses, a well-known and proven example of which is the Warren Truss shown in figure 1.

Our experimental goal is to improve the design of a simple seven-segment truss, where a segment is an equilateral triangle. The truss is fixed and pinned at either end, and a load of 17000 Newtons is applied in the centre of the structure.

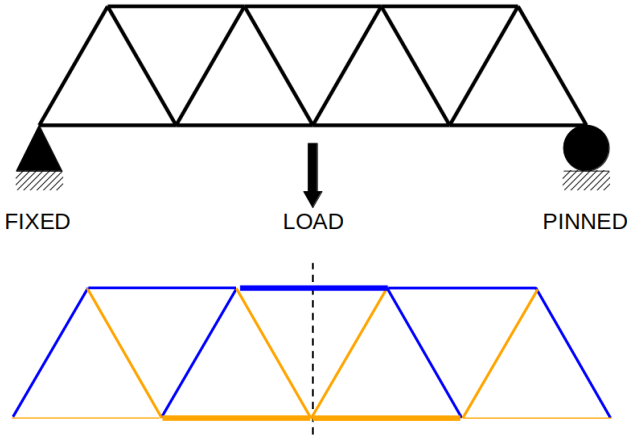

Fig. 1. Engineering description of the seven-segment truss. The top figure shows the arrangement of loads and supports. The resulting loading is shown in the bottom figure. Members coloured in orange are in tension, those in blue are in compression. The thickness of a member is proportional to the force acting on it.

This structure is simulated in Calculix [18] as a planar static truss in 3D. The material is aluminium, with a circular cross sectional area of 1m^2. The equilateral triangles forming the truss have a side length of 25m. Thus the total volume is 375m^3. Simulation returns forces on the truss members, from which other properties can be calculated, e.g. stress, strain energy, buckling of the members, or indeed properties of the entire structure, e.g. total strain energy.

The fitness measures are used in the process of selection. Multi-objective measures were used to drive the system to generate a range of design solutions. Here, the two objectives were to minimise the total volume of material V, and the total strain energy U. The total volume V of the structure is the sum of the volume of each member:

$$V = \sum_{m \in M} A_m L_m \tag{1}$$

where m is a single member of the structure in the set M, A_m is the cross-sectional area of m and L_m is the length of m. The strain energy is a measure of the work done to deform the structure under load, and is calculated using:

$$U = \sum_{m \in M} \frac{F_m^2 L_m}{2 E_m A_m} \tag{2}$$

where F_m is the force acting on member m (derived via simulation), and E_m is Young's modulus of elasticity for the material of which m is made.

B. Gene Regulatory Network (GRN)

The development process aims to improve on this basic design by iteratively moving the joints in the truss to redistribute the forces around the structure. Physical properties of the members and joints are used as inputs to the GRN, which produces a set of "output deltas" (δ) – small changes to the X, Y position of each joint forming the same triangle. This approach is directly extendable to any 2D truss structure, and also to 3D structures where the 'cell' would be formed from tetrahedra instead of triangles.

Algorithm 1: EvoDevo for Generative Design. The algorithm is an iteration over generations and population as in a common genetic algorithm. The Devo stage performs iterative growth on the individual using feedback derived from forces on the structure, and calculates the fitness of the final form of the structure, which is used in a selection process to evolve the next generation.

Input : G = number of generations
P = population size
D = number of development steps
S_0 = initial seed structure

Genomes $g_{p \in P}$ = RandomNEAT()
for g *in* $1..G$ **do**
  for p *in* $1..P$ **do**
    GRN γ_p = CreateGRNFromNEATGenome(g_p)
    Structure S_d = InitialiseStructure(S_0,γ_p)
    for d *in* $1..D$ **do**
      Forces F = CalculateForces(S_d)
      Cells C = CellsInStructure(S_d)
      for c *in* C **do**
        f_c = LocalForces(c,F)
        δ_c = RunGRN(f_c)
      end
      S_d = UpdateStructure($\forall \delta_c \in C$)
      (delta fitness $q_{d,f}$ = StructureFitness(S_d))
    end
    fitness p_f = StructureFitness(S_D)
  end
  Pareto ϕ = ParetoFront($\forall p_f \in P$)
  P_{g+1} = evolveNEATPopulation(ϕ,P)
end
Return: ϕ

Since it was necessary to be able to evolve the topology of the GRN, an evolvable variable-length encoding was needed which could emulate the complexities of gene regulation and be amenable to the evolutionary operators of crossover and mutation. In the absence of an appropriate GRN model, it was decided to emulate its function via a readily-available neural network system with the desired properties: NEAT [14]. Using NEAT, it was possible to specify the inputs and outputs of the GRN, and then initialise an evolutionary run with a population of randomly-configured NEAT networks to test the concept that this could drive the development of appropriate truss structures via selection. The inputs to the GRN were the strain energy values for each member in the triangle and the X and Y positions of the vertices of the triangle. (Additional inputs and outputs on the Z axis were implemented as required by the Calculix software, but updates on the Z axis were always zero-valued in this experiment to restrict the design

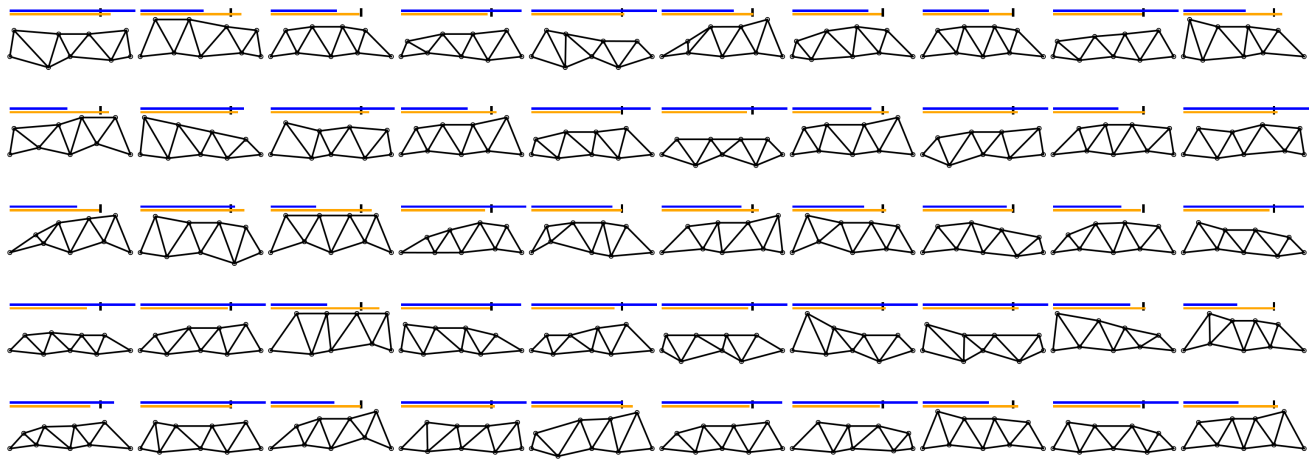

Fig. 2. Initial population of developed designs. Each design is achieved by passing position and strain energy data from each triangle through a randomly generated NEAT GRN. The output of the GRN consists of update δ values for X and Y of each point around the cell. These δ values are consolidated into updates to the structure. The process is repeated $D = 10$ times. The small graphic above each truss image shows the relative values of U (top, blue) and V (bottom, orange) normalised by the respective values for the initial structure in figure 1 ($V = 375.0, U = 0.641$), shown as a tick mark.

to 2D). The outputs from the GRN are the δ values in X and Y of each vertex in the triangle. Where vertices were shared between triangles, the update to the position of the vertex is the mean of all δ values pertaining to that point. Note that these point movements encode the morphological change in the design that are analogous to the concept of growth in biological EvoDevo, but the structure does not necessarily increase in size – especially since one of the fitness measures seeks a reduction in the material volume of the structure.

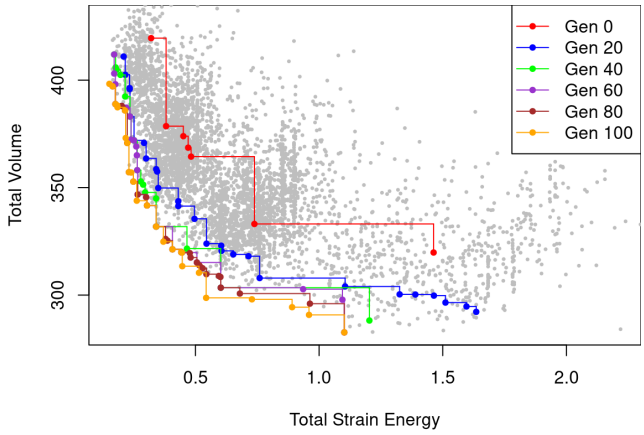

Fig. 3. Change in the Pareto front over 100 generations. Every individual from every generation is shown in grey. Overlaid are the Pareto fronts from every 20 generations. The goal is to minimise both the total strain energy and the total volume of the structure after growth. The final Pareto front (shown in gold) yields a set of designs that are improved in both objectives over the initial, Pareto front (shown in red).

C. Devo Stage

The structural components of the Devo stage described above now need to be assembled into an iterative algorithm. This forms the inner two loops shown in lines 4-16 of Algorithm 1. There are three stages to Devo, which are executed for every individual in each generation of the evolutionary run:

1) Initialisation: Every individual in these experiments starts with the same structure, composed of equilateral triangles as described in section II-A. However, the GRN of each individual is unique, and is set up from the NEAT genotype. The initialisation stage sets up the initial structure for simulation, defines the cells within that structure, and organises the manner in which the inputs to the GRN are calculated and arranged. The initial structure illustrated in figure 1 consists of 7 equilateral triangles, so 7 cells were needed in the inner Devo loop for this experiment. Each cell accesses individual structure's associated GRN, which drives the growth stage. Each cell sends its data to the GRN and receives output values from the GRN, emulating the biological situation where each cell contains its own physical copy of the GRN.

2) Iteration: Having set up the initial structure, the algorithm iterates over a preset number of development steps. The first part of this process involves setting up and running a simulation of the loading environment shown in figure 1 to determine the properties of each member. From here, each cell in the structure is considered independently by passing the properties through the NEAT GRN in order to obtain local δ values for repositioning the joints around each cell. Once all the cells have been processed in this manner, the final Devo step is to consolidate the δ values pertaining to each joint in the structure to obtain a new position.

3) Interrogation: Having completed growth, the final step in the Devo stage is to determine the fitness of the resulting structure. Note that the fitness measures do not need to have any particular relationship to the inputs and outputs of the GRN. The algorithm has two levels of feedback: one at the cell level via the GRN; and one at the complete design level via the fitness measures.

D. Evolutionary Stage

The heart of this algorithm is the evolvable GRN, which is mostly handled using NEAT-Python [19]. The main change to this algorithm has been to extend it to handle multi-objective fitness functions. This has been achieved in a similar manner to the functioning of the NSGA-II algorithm [20], whereby fitness ranking of an individual is determined by its Pareto rank and its crowding distance measure. Note that the crowding distance and speciation both aim to improve the diversity of the population. Multi-objective optimisation is highly desirable in generative design (indeed in any real-world design). In the example used here, minimising (or maximising) a single objective such as volume can drive the system to select for unrealistic designs. The Pareto front is intended to capture a range of design options ranging from high-volume solutions which are strong but expensive to low-volume solutions which are relatively cheap but also relatively weak. In between these extrema lies a 'sweet spot' where improvements on the initial structure in more than one fitness objective can be found.

For the experiment reported here, a population size of 50 evolved over 100 generations, using 10 Devo steps to improve the initial structure.

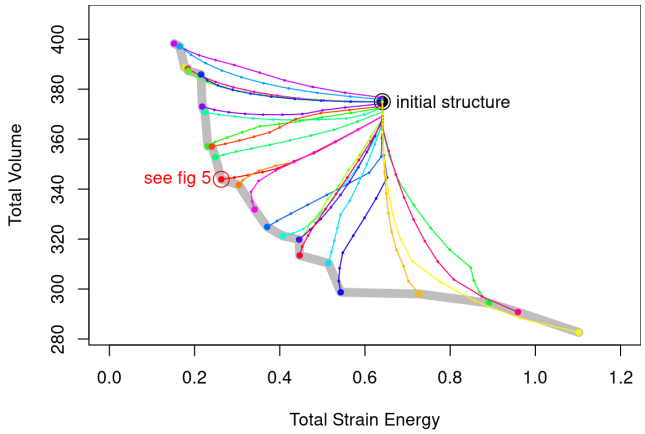

Fig. 4. Trajectories of development in fitness space for all individuals on the final Pareto front (shown in grey). Fitness of the initial structure is shown as a black point and each individual (coloured lines) grows from here to its final position on the Pareto front. The points along each coloured line show the way growth changes the fitness of each individual. The individual whose growth is further described in figure 5 is shown in red, with a circle around its final position on the Pareto front.

E. Algorithm

Algorithm 1 gives pseudocode which summarises how the operations described above are combined programatically. The two outer loops, which iterate over the population for each generation handles the evolution of the GRN. The two inner loops iterate over the cells in the structure for each Devo growth step to generate the final structure from its initial state[1].

[1]Source code for the python implementation of this algorithm will be made available on publication

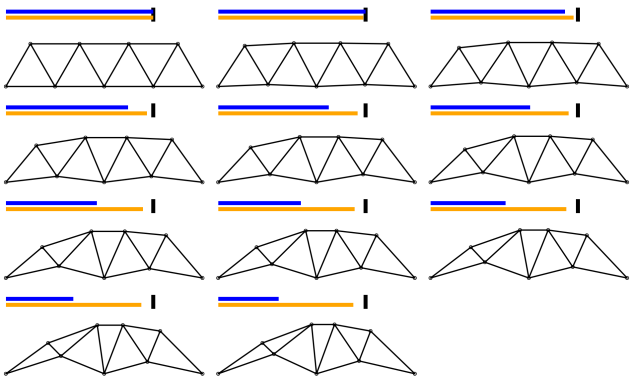

Fig. 5. Development of the design on the midpoint of the Pareto front, indicated by the red ring in figure 4. Steps are ordered left to right, top to bottom. Values of V and U are shown graphically, as described for figure 2. Each Devo step shows a reduction in the volume V of the structure, but also a decrease in the total strain energy U, indicating that the GRN guides the growth of the structure to a state which better distributes the load.

III. Results

This initial experiment is demonstrates the concept that a single run of the algorithm can deliver improved designs of the truss. Figure 2 shows the developed structures for every member of the population at generation 0. 22 of the 50 structures show a decrease in volume and 24 showed a decrease in strain energy. Only 7 showed a decrease in both volume and strain energy, and only 2 of these 7 points were members of the intial Pareto front. This demonstrates that randomly-configured GRNs are unlikely to generate good designs, and acts as a benchmark against which to measure the effectiveness of the evolutionary process.

The evolution of the Pareto front towards GRNs that are capable of reducing both the material volume and the total strain energy of the structure is shown in figure 3. The GRNs evolve rapidly to generate better solutions from generations 0 to 40, but show relatively little improvement from generation 40 to 100. This is to be expected since there are limits on the absolute value of δ, meaning that each joint in the truss can move a maximum of 10 metres during development. There is also a limit in terms of how small a value of strain energy can be obtained using a seven-segment truss with this topology. Note also that the Pareto front at generation 0 has only 7 points, whereas that for generation 100 has 24, meaning that there are many more non-dominated solutions by the time evolution reaches the final generation.

Although not required by the Evo stage, it is possible to calculate and record the fitness of an individual at each growth iteration during Devo. This action is indicated in brackets on line 14 of algorithm 1. This change in fitness can be recorded to see how growth manifests itself in fitness space. This is shown for all individuals on the final Pareto front in figure 4. The trajectories are generally linear in nature, reflecting the fact that growth is via the iterative firing of the same GRN. Some curvature in these trajectories indicate non-linear changes in the way the GRN fires, which is the sort of

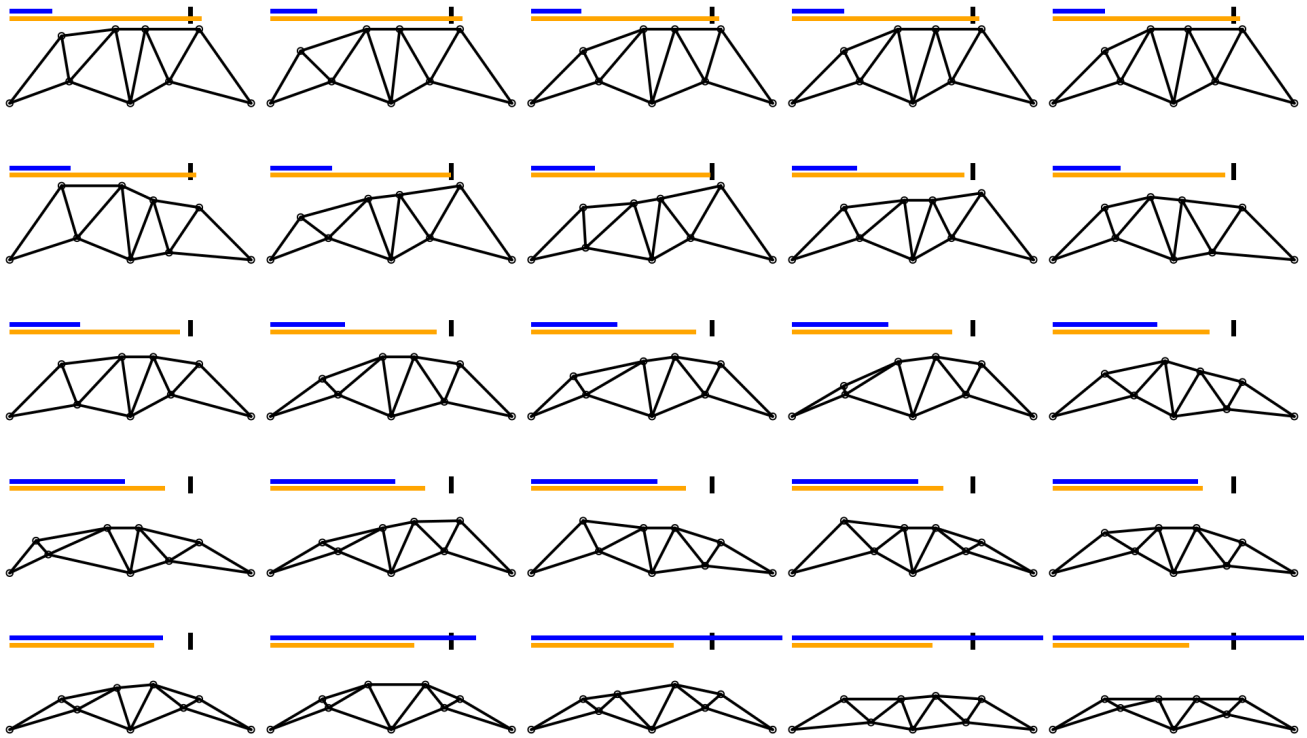

Fig. 6. Final Pareto front of developed designs, ranked by total strain energy. Steps are ordered left to right, top to bottom. Values of V and U are shown graphically, as described for figure 2. The first six designs show an increase in material volume, but have the best reduction in strain energy. The final four designs show the lowest volume, but an increase in strain energy. All other designs show an improvement on the original in structure on both objectives, transitioning from lower to higher strain energy whilst going from higher to lower material volume

behaviour that may be needed with more complex structures. It is striking how some groupings of trajectories have a common early path and then diverge, and some trajectories cross the path of others, again indicating non-linear responses to the range of input vectors generated during growth.

An example of the developmental updates to the structure is shown in figure 5, which corresponds to the red trajectory in figure 4, indicated with a red ring around it's final position along the Pareto front. It is clear that the GRN is guiding the upper horizontal towards a semi-circular shape, which is known to be the most appropriate way of handling compressive loads. Recall that the support and loading points are fixed, but the remaining two points on the lower horizontal tend to move upwards, with the effect of shortening the cross bracing and reducing the overall volume of the structure. The resulting forms are only approximately symmetrical, but this is to be expected because of the way the output nodes of the GRN are interpreted as pertaining to the X, Y positions of each joint in the triangle, clockwise from the leftmost point, so the GRN has no information about the symmetry of the structure.

The final state of each individual design on the evolved Pareto front is shown in figure 6, ranked by the total strain energy of each design. The top and bottom rows of this figure show structures in which only one of the objectives is better than those of the initial structure – in the top row, it is the total strain energy that is minimised, whereas in the bottom row, it is the total material volume that is minimised. The middle rows show evolved designs that are better in both objectives than the initial structure, demonstrating that the algorithm is capable of evolving GRNs that can drive the design of a structure to optimal solutions.

IV. Discussion

In the absence of a designer in biology, the development of a multi-cellular organism from a single cell is governed by gene regulatory networks, which through evolution have gained the capability to grow organisms with a level of sophistication that can be seen in all facets of the living world around us. It has been shown in this paper how this phenomenon can be emulated in an artificial organism, and used to generate appropriate structures in the absence of a design engineer.

It is remarkable that the same network used over each cell and over each Devo step can evolve to generate better designs by generating an appropriate response to the local physical forces. One useful feature of this approach is that the Devo part of the process allows for a broad range of individuals to be created in a range of environments (e.g. load cases) and so introduce more variance in the generations with a corresponding variety in the resulting solutions.

The novelty in this work lies in the combination of physical simulation, EvoDevo and multi-objective NEAT. Of these

components, the NEAT-based GRN is perhaps the best candidate for further experimentation. It is reasonable to expect that other evolvable paradigms could be used as proxies for GRNs such as Genetic Programming [21] or Artificial Chemistries [22].

This approach has the following advantages: firstly, it adds an inner feedback loop (the Devo process) which can be used for local optimisation; secondly, a wide range of inputs and outputs can be used within this framework, meaning the technique can be used on varied design challenges; thirdly, the resulting GRNs have the potential generalise over different families of problems without the need to retrain.

Having successfully proved the overall concept, new research goals arise, central among them being how to add new topology to the structure, how and whether to induce cell differentiation (by introducing more statefulness to the NEAT GRN representation), how easily to extend the work to three-dimensional trusses and how this will fit into an engineering design process. Another option is to examine how successfully evolved networks can be applied to other structures, to explore how well the GRNs generalise. In order to apply this approach to real engineering problems, it will be important to compare evolved design strategies with current design practice.

V. Acknowledgements

This work was supported by EPSRC programme Grant Ref is EP/V007335/1, "RIED: Re-Imagining Engineering Design". The authors thank Kate van Lopik, Paul Goodall and Peter Kilpatrick for comments on an early draft of this manuscript.

References

[1] K. Deb and S. Gulati, "Design of truss-structures for minimum weight using genetic algorithms," *Finite Elem. Anal. Des.*, vol. 37, no. 5, pp. 447–465, May 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168874X00000573

[2] P. Nanakorn and K. Meesomklin, "An adaptive penalty function in genetic algorithms for structural design optimization," *Comput. Struct.*, vol. 79, no. 29, pp. 2527–2539, Nov. 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045794901001377

[3] G. G. Tejani, V. J. Savsani, V. K. Patel, and P. V. Savsani, "Size, shape, and topology optimization of planar and space trusses using mutation-based improved metaheuristics," *Finite Elem. Anal. Des.*, vol. 5, no. 2, pp. 198–214, Apr. 2018. [Online]. Available: https://academic.oup.com/jcde/article-pdf/5/2/198/33135953/j.jcde.2017.10.001.pdf

[4] N. D. Lagaros, M. Papadrakakis, and G. Kokossalakis, "Structural optimization using evolutionary algorithms," *Comput. Struct.*, vol. 80, no. 7, pp. 571–589, Mar. 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045794902000275

[5] J. J. McKeown, "Growing optimal pin-jointed frames," *Structural Optimization*, vol. 15, no. 2, pp. 92–100, Apr. 1998. [Online]. Available: http://link.springer.com/10.1007/BF01278495

[6] M. Price, W. Zhang, I. Friel, T. Robinson, R. McConnell, D. Nolan, P. Kilpatrick, S. Barbhuiya, and S. Kyle, "Generative design for additive manufacturing using a biological development analogy," *Finite Elem. Anal. Des.*, vol. 9, no. 2, pp. 463–479, Mar. 2022. [Online]. Available: https://academic.oup.com/jcde/article-abstract/9/2/463/6547705

[7] Vijesh, Chakrabarti, and Sreekumar, "Modeling of gene regulatory networks: A review," *J. Biomed. Sci.*, 2013. [Online]. Available: https://www.scirp.org/html/28365.html?pagespeed=noscript

[8] R. Doursat, H. Sayama, and O. Michel, "A review of morphogenetic engineering," *Nat. Comput.*, vol. 12, no. 4, pp. 517–535, Dec. 2013. [Online]. Available: https://doi.org/10.1007/s11047-013-9398-1

[9] P. Bentley, "An introduction to evolutionary design by computers," *Evolutionary design by computers*, 1999.

[10] L. Wolpert, "Positional information and the spatial pattern of cellular differentiation," *J. Theor. Biol.*, vol. 25, no. 1, pp. 1–47, Oct. 1969. [Online]. Available: http://dx.doi.org/10.1016/s0022-5193(69)80016-0

[11] J. Sharpe, "Wolpert's french flag: what's the problem?" *Development*, vol. 146, no. 24, Dec. 2019. [Online]. Available: http://dx.doi.org/10.1242/dev.185967

[12] D. Navarro-Mateu and A. Cocho-Bermejo, "Evo-Devo strategies for generative architecture: Colour-Based patterns in polygon meshes," *Biomimetics*, vol. 5, no. 2, May 2020. [Online]. Available: http://dx.doi.org/10.3390/biomimetics5020023

[13] K. Walker, H. Hauser, and S. Risi, "Growing simulated robots with environmental feedback: an eco-evo-devo approach," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21. New York, NY, USA: Association for Computing Machinery, Jul. 2021, pp. 113–114. [Online]. Available: https://doi.org/10.1145/3449726.3459514

[14] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002. [Online]. Available: http://dx.doi.org/10.1162/106365602320169811

[15] A. Lomas, "Species explorer: An interface for artistic exploration of multi-dimensional parameter spaces," *Electronic Visualisation and the Arts*, pp. 95–102, 2016. [Online]. Available: https://www.scienceopen.com/hosted-document?doi=10.14236/ewic/EVA2016.23

[16] ——, "Cellular forms: an artistic exploration of morphogenesis," in *ACM SIGGRAPH 2014 Studio*, ser. SIGGRAPH '14, no. Article 1. New York, NY, USA: Association for Computing Machinery, Jul. 2014, p. 1.

[17] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined with physics," in *Evolvable Systems: From Biology to Hardware*. Springer Berlin Heidelberg, 1997, pp. 390–405.

[18] Dhondt, "Calculix crunchix user's manual version 2.12," *Munich, Germany, accessed Sept*, 2017. [Online]. Available: http://www.dhondt.de/ccx_2.19.pdf

[19] McIntyre, Alan and Kallada, Matt and Miguel, Cesar G. and Feher de Silva, Carolina and Netto, Marcio Lobo, "neat-python." [Online]. Available: https://github.com/CodeReclaimers/neat-python

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[21] M. T. Ahvanooey, Q. Li, M. Wu, and S. Wang, "A survey of genetic programming and its applications," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 4, pp. 1765–1794, 2019. [Online]. Available: https://www.koreascience.or.kr/article/JAKO201919761177651.page

[22] W. Banzhaf and L. Yamamoto, *Artificial Chemistries*. MIT Press, Jul. 2015. [Online]. Available: https://play.google.com/store/books/details?id=oI4jCgAAQBAJ