Evolving Novel Gene Regulatory Networks for Structural Engineering Designs

Rahul Dubey
Missouri State University
Department of Computer Science
University of York
School of Physics, Engineering, and Technology
rahuldubey@missouristate.edu

Simon Hickinbotham
University of York
School of Physics, Engineering, and Technology

Andrew Colligan
Queen's University Belfast
School of Mechanical and Aerospace Engineering

Imelda Friel
Queen's University Belfast
School of Mechanical and Aerospace Engineering

Edgar Buchanan
University of York
School of Physics, Engineering, and Technology

Mark Price
Queen's University Belfast
School of Mechanical and Aerospace Engineering

Andy M. Tyrrell
University of York
School of Physics, Engineering, and Technology

Abstract Engineering design optimization poses a significant challenge, usually requiring human expertise to discover superior solutions. Although various search techniques have been employed to generate diverse designs, their effectiveness is often limited by problem-specific parameter tuning, making them less generalizable and scalable. This article introduces a framework inspired by evolutionary and developmental (evo-devo) concepts, aiming to automate the evolution of structural engineering designs. In biological systems, evo-devo governs the growth of single-cell organisms into multicellular organisms through the use of gene regulatory networks (GRNs). GRNs are inherently complex and highly nonlinear, and this article explores the use of neural networks and genetic programming as artificial representations of GRNs to emulate such behaviors. To evolve a wide range of Pareto fronts for artificial GRNs, this article introduces a new technique, a real value–encoded neuroevolutionary method termed *real-encoded NEAT* (RNEAT). The performance of RNEAT is compared with that of two well-known evolutionary search techniques across different 2-D and 3-D problems. The experimental results demonstrate two key findings. First, the proposed framework effectively generates a population of GRNs that can produce diverse structures for both 2-D and 3-D problems. Second, the proposed RNEAT algorithm outperforms its competitors on more than 50% of the problems examined. These results validate the *proof of concept* underlying the proposed evo-devo-based engineering design evolution.

Keywords
Evolutionary search, gene regulatory networks, NEAT, CGP, design optimization

1 Introduction

Evolutionary developmental (evo-devo) biology is the part of biology that tries to explain the evolution of growth patterns in organisms, which determine how they develop from a single cell to adulthood (Hall, 2012). Every cell of an organism contains the same DNA, yet each can function in different ways to generate different organs, such as the heart or eyes (Olson, 2006). These

differences in functionality are caused by gene regulation that turns on and off different parts of the genome depending on local environmental factors, endowing cells with *state*. The DNA/genome of a cell consists of a set of genes, but only a fraction of these are activated (turned ON) to create a specific organ. The cohort of gene regulations happening in parallel can then generate a complex multiorgan organism. These different molecular gene regulators combined together create a gene regulatory network (GRN) (Cussat-Blanc et al., 2019). The GRN is the central and most crucial component of the evolutionary developmental cycle.

This article takes inspiration from the evo-devo concept to generate different engineering designs by evolving an artificial GRN. In the field of engineering, structural design optimization has been a topic of study for several decades, involving the use of domain expert knowledge (Christensen & Klarbring, 2008) and computational intelligence techniques (Chi et al., 2021). This optimization process encompasses various aspects, such as size, shape, and topology optimization, either individually or in combination. Conventional approaches to design optimization focus on finding solutions for specific problems by directly encoding the structural representation of the genome. This approach often results in a large search space that needs to be explored through automated evolution. Direct encoding implies that these approaches can only be problem specific and so face limitations in both scalability and generalizability.

To address these challenges, a framework based on evo-devo principles to generate and evolve diverse designs using artificial GRNs is introduced here. The evolutionary component offers a direct encoding of GRNs, while the developmental aspect utilizes these GRNs to update the structures. The primary objective of this approach is to evolve GRNs that are both generalizable and scalable, enabling them to effectively control the growth of engineering designs to optimize the structure for given performance targets (e.g., structural loads). In the existing literature, various computational models of GRNs have been proposed, ranging from low complexity with better explainability to high complexity with lower explainability (Karlebach & Shamir, 2008). However, there is currently no consensus regarding the most suitable method or methods for their applicability. GRNs are inherently nonlinear, and thus different nonlinear models have been studied to mimic the behavior of GRNs.

In this article, an artificial GRN is represented by two different models: neural networks (NN) and computer programs encoded in graphs where problems are formulated as multiobjective optimization problems. The nonlinearity of both NNs and graphs can be varied by changing their hyperparameters. Because the problems are multiobjective in nature, two different evolutionary search methods have been employed: Multiobjective Cartesian genetic programming (CGP) (Miller & Harding, 2008) and multiobjective neuroevolution of augmenting topologies (NEAT) (Stanley & Miikkulainen, 2002) is used to evolve complex computer programs and NNs, respectively. However, extending NEAT to evolve for multiobjective problems while maintaining speciation poses a significant challenge (Van Willigen et al., 2013). To evolve multiobjective NN topologies, this article introduces a CGP-style encoding to encode a neural architecture, where the number of hidden layers and nodes in each hidden layer can be defined. CGP-style network encoding allows tuning of the connections between nodes, weights, and biases. The entire architecture can be encoded into a real-value chromosome and thus is referred to as *real-encoded NEAT* (RNEAT).

By employing the proposed evo-devo-based approach, a range of experiments were carried out on diverse 2-D and 3-D engineering structural design problems. In these experiments, a GRN receives inputs derived from the local geometry of the structure, including the cross-sectional (CS) area of members and node locations, as well as information obtained through finite element analysis (FEA), such as member force or strain energy. The GRN, encoded as either a NN or a computer program, subsequently generates small changes to update (i.e., grow) the physical structure, aiming to minimize both the volume and maximum deflection. The experimental results obtained under various settings demonstrate the viability of the evo-devo-based approach for controlling the growth of structures while achieving the desired objectives. Furthermore, the results reveal that RNEAT-evolved solutions outperformed competitors on more than 50% of the problems, indicating their superiority in terms of effectiveness and performance.

The two major contributions of this article are as follows: (a) The article introduces a generalizable and scalable approach based on the evo-devo concept to evolve diverse Pareto front engineering designs, and (b) the article presents RNEAT by taking inspiration from NEAT and CGP to evolve multiobjective NNs. The rest of the article is structured as follows. Section 2 discusses the literature on evo-devo approaches, GRNs, and evolutionary search in engineering design. Section 3 presents the proposed evo-devo framework, and section 4 outlines different GRN representations and the methodology of the proposed RNEAT algorithm. The experimental setup and resulting data are presented and discussed in section 5, while section 6 provides the conclusions and suggests areas for future research.

2 Related Work

Numerous difficulties emerge when employing evolutionary search techniques to evolve solutions for intricate real-world problems, including the challenges of selecting relevant parameters for tuning and formulating the fitness function (Goldberg, 2002; Osaba et al., 2021). The complexity of the search space further exacerbates the situation in engineering design problems, as these spaces are characterized by a nonlinear relationship between parameters that generate phenotypes (Zou et al., 2022). This article introduces an evo-devo-based approach to address these challenges in evolving GRNs for controlling the growth of structural designs.

Approaches based on evolutionary development have been studied to improve the performance of both hardware and software on different domain-specific tasks that include designs (Richards et al., 2012), digital architecture (Navarro-Mateu & Cocho-Bermejo, 2019), music (Albarracín-Molina et al., 2016), and color-based pattern generation (Navarro-Mateu & Cocho-Bermejo, 2020). For instance, Vujovic et al. (2017) introduced an evo-devo strategy to evolve the morphology of physical robots. Their proposed approach enabled robots to grow their leg size to simulate ontogenetic morphological changes in three developmental steps. In each step, either the length of robot legs changes or the length and thickness both change.

Wu et al. (2022) proposed an evolutionary developmental framework to facilitate robotic Chinese stroke writing. Here a genome encodes stroke trajectory points, and the fitness of the genome is computed using a developmental learning algorithm. However, the genome encoding is not scalable, and evolved solutions are problem specific. Mccormack and Gambardella (2022) presented "growing and evolving 3D printable designs" where a covariance matrix adaptation evolutionary strategies algorithm tunes five genetic parameters for optimizing 3-D printable structures. Bidlo and Dobeš (2020) presented an evolutionary developmental method for the design of arbitrarily growing sorting networks. The proposed method basically evolves a grammar, an alphabetically encoded genome, to generate complex strings, and these strings are later converted into comparator structures that are the building blocks of sorting networks.

These evo-devo approaches use evolutionary algorithms to tune numerical or alphabetical parameters to evolve solutions and are limited in terms of scalability and generalizability. Unlike these, the evo-devo approach proposed in this article relies on GRNs to govern the growth of design in developmental steps. In the field of engineering design, evo-devo-inspired approaches have not been significantly investigated to evolve designs.

The GRN plays a crucial role in organism development, and different computational techniques that act as GRN representations have been studied in the literature (e.g., Cussat-Blanc et al., 2019; Delgado & Gómez-Vela, 2019; Schlitt & Brazma, 2007; Karlebach & Shamir, 2008). These techniques range from simple logical functions, such as Boolean functions, to complex nonlinear functions represented by NNs. In the literature, various evolutionary and swarm algorithms have been used to evolve GRNs. In 2005, Swain et al. used evolutionary algorithms to generate computational models of GRNs using data obtained from observations. Xu et al. (2007) used differential evolution (DE), partial swarm optimization (PSO), and a hybrid of DE and PSO to optimize the hyperparameters of recurrent NNs to model the behavior of a GRN on a time series dataset, and Cussat-Blanc et al. (2015) used NEAT to evolve NN topologies. In a manner similar to NNs, computer graphs

(evolving using genetic programming) have also been studied as a representation for GRN (e.g., Streichert et al., 2004). In this article, NNs and genetic programming are considered proxies for artificial GRNs that govern the growth of structural designs and are evolved using CGP and NEAT. CGP was chosen to evolve computer programs because it has been shown in the literature that other types of genetic programming suffer from bloating, whereas CGP does not (Turner & Miller, 2014).

Evolving multiobjective neural architectures has been recognized as a challenging problem. Van Willigen et al. (2013) modified standard NEAT using strength Pareto evolutionary algorithms to evolve multiobjective network topologies. However, to preserve speciation in the population, fitness-based domination is used to convert multiobjective fitness into single-objective fitness and thus is not a true multiobjective NEAT. Schrum and Miikkulainen (2008) used NSGA-II (Deb et al., 2002) to modify NEAT and evolved different topologies. The concept of speciation is not used here because fitness sharing in multiobjectives is difficult, as there is more than one objective value. This article presents a hybrid of CGP and NEAT, where encoding is inspired by CGP and information flow between layers is inspired by NEAT.

Apart from evo-devo-based approaches, in the literature, different bio-inspired algorithms have been used to evolve design (Balamurugan et al., 2008; Perez & Behdinan, 2007; Yildiz, 2013), where problems are categorized into topology, size, and shape optimization. In topology optimization, a bit-array representation scheme has been frequently used to formulate the design problem where the design space is discretized into a grid where rectangular blocks are filled with materials to generate different topologies (Wang et al., 2006). The optimal solution in this representation is dependent on the resolution of the grid, where the smaller the resolution is, the better is the chance of searching for optimal solutions. As the grid resolution increases, the search space increases, and thus these approaches are neither scalable nor generalizable. The next section describes in detail the new algorithm and framework that form the foundational work of this article.

3 Methodology

3.1 Biological to Structural Cell Analogy

As biological cells are the basic building blocks of any organ (Kaldis, 2016), this article considers a triangularly shaped cell for 2-D problems and a tetrahedrally shaped cell for 3-D problems as building blocks to create engineering designs. Figure 1(a) illustrates biological cell growth, where the size of the cell changes and the cell multiplies (by dividing a cell) to generate multiple cells. Cell growth and division mechanisms are controlled by a GRN or a set of GRNs based on local environmental factors.

Figure 1(b) shows an artificial cell, its growth, and multiplication mechanisms. An artificial 2-D triangularly shaped cell[1] consists of three nodes (n_a, n_b, and n_c) and three edges (e_a, e_b, and e_c), where the cell grows by changing the properties of nodes and edges. For consistency, herein cell division is also referred to as a type of growth mechanism. A cell grows by edge growth ($\mathcal{G}_1$), node growth ($\mathcal{G}_2$), or cell division ($\mathcal{G}_3$), as shown in Figure 1(b). A GRN takes the state information of a cell (such as the CS area of edges, locations of nodes, and other properties computed using FEA, such as edge force and strain energy) and generates a response to growing the cell.

In the first type of growth ($\mathcal{G}_1$), a GRN governs/updates the thickness of the edge/member's CS area, as shown in Figure 1(b), where the red line shows an increment in CS area and the green line indicates a reduction in CS area. By changing the CS area of members, the size of the structure can be optimized. In a similar fashion, the second type of growth mechanism ($\mathcal{G}_2$) updates the locations of the nodes, as shown in Figure 1(b). Again, by changing the node locations, the size of the structure can be optimized. Finally, the topology of the structure can be modified by adding nodes to the structure, the third type of growth mechanism ($\mathcal{G}_3$), where a single cell is divided into three cells

1 This type of cell representation has been chosen because the aim here is to evolve 2-D and 3-D truss-like structural designs. This cell definition can be easily changed as per problem requirements.

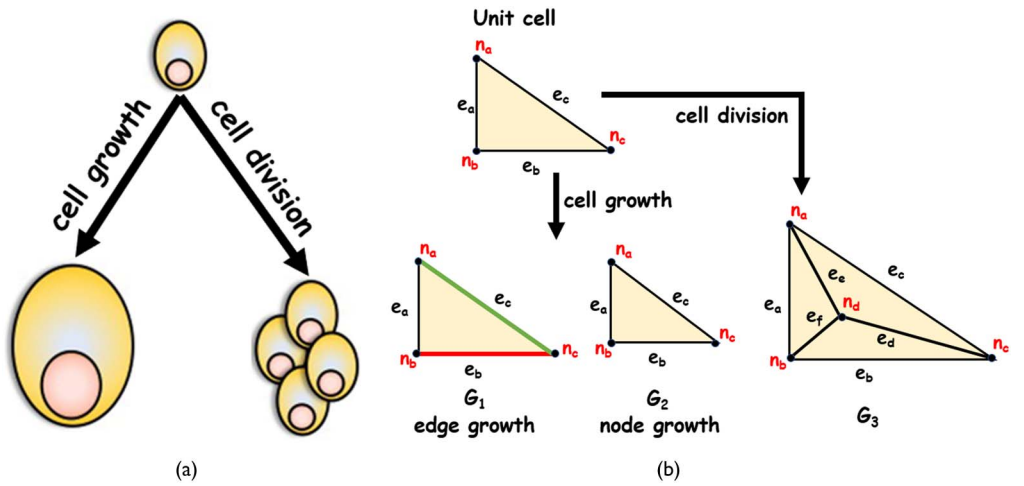

Figure 1. Biological and structural cell analogy. (a) An example biological unit cell, its growth, and its division (Kaldis, 2016). (b) Examples of cell growth in three different types of growth mechanism: $\mathcal{G}_1$, edge growth; $\mathcal{G}_2$, node growth; and $\mathcal{G}_3$, cell division.

by placing a node (n_d) at the centroid of the parent cell (Figure 1(b)). In the proposed evo-devo framework, a node can be added at the centroid of a cell. The same types of growth mechanisms are also applicable to 3-D structures where a cell is tetrahedral.

3.2 Structure Initialization

The previous subsection discussed artificial cell representation and different types of growth mechanisms. These cells are the basic building blocks of design. To produce designs, an initial structure is generated within an environment that represents the design context, for example including support points, load points, load magnitude with direction, and material properties. One such example of a 2-D initial structure is shown in Figure 2, where Figure 2(a) shows the location of points and Figure 2(b) shows the resulting geometry. The initial topology is called the *seedling*, and it is from this basis that all growth steps proceed. In the case of the simple frame structures being used here, a Delaunay triangulation (DT) can be used to create the topology of the seedling/structure. The initialized structure is made up of four cells and each cell has three nodes and three edges/members.

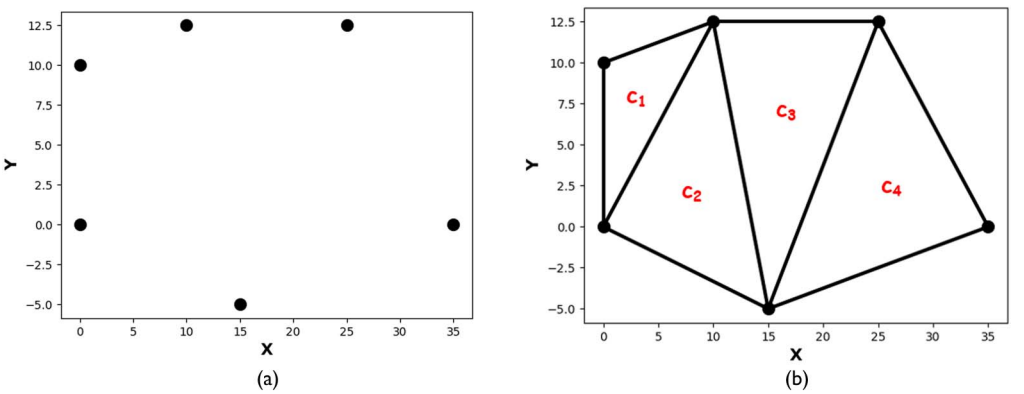

Figure 2. Example of the initial seedling/structure generated using a set of points. (a) A set of points that are locations of supports and loads. (b) Resulting geometry with four cells (c_1, c_2, c_3, c_4) using Delaunay triangulation. (a) Nodes, (b) Structure with four cells.

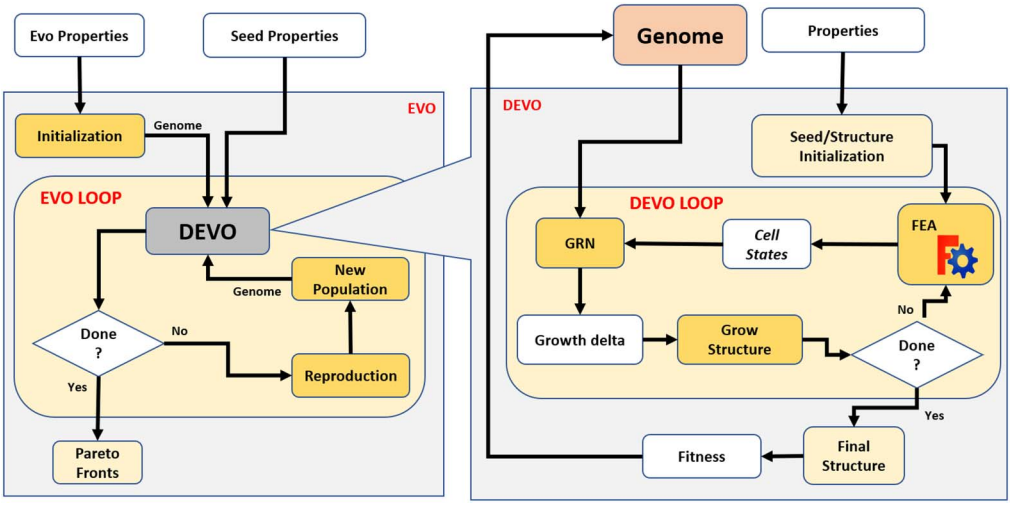

Figure 3. Block diagram representation of the evo-devo framework for growing structural design where evo evolves solutions/GRNs and devo grows the structure for a predefined number of developmental steps for each GRN. A GRN controls the growth, and at the end, the physical properties of the structure are assigned as the fitness to the GRN.

Once the initial structure/seed is generated, this structure can grow depending on local environmental factors in the evo-devo phase.

The block diagram in Figure 3 illustrates the evo-devo framework for the evolutionary development of engineering designs. The framework consists of two main components: evo and devo. The evo component focuses on the representation and evolution of GRNs, with devo serving as the evaluator of the GRN's quality, measured by its ability to generate Pareto-optimal designs.

3.3 Evo: Evolving GRNs

The evo stage randomly initializes a population of solutions/genomes where each solution encodes an artificial GRN. Here the encoding differs depending on the type of GRN representation. To evaluate the quality/fitness of each genome, the devo component is initialized where the GRN controls the growth of a structure for d developmental steps. At the conclusion of the devo process, the physical properties (e.g., volume and deflection) of the updated structure are assigned as fitness to the genome. On the basis of the fitness of GRNs, the selection, crossover, and mutation operators evolve GRNs for a predefined number of generations within the evo-loop. In this work, NNs and computer programs are used as GRNs representation where different multiobjective evolutionary search techniques evolve a Pareto front of GRNs that generate diverse designs. Section 4 discusses in detail GRN representation and the proposed hybrid evolutionary search algorithm.

3.4 Devo: Growing Structure

In the devo component, a structure undergoes a growth phase through various mechanisms, such as altering CS areas, relocating nodes, adding new nodes, or combinations thereof. Within the evo-devo framework, evo provides an indirect encoding of growth rules, whereas devo carries out the actual growth process based on local environmental factors. Figure 3 (right) is a block diagram representation of the devo component. Here, first, a structure is generated using predefined structural properties, and a GRN is generated by decoding the genome. In artificial evolution, the introduction of a development step allows a structure to be updated through different growth mechanisms. In the literature, the principles of devo have been demonstrated with simple engineering design problems, such as brackets (Price et al., 2022) and trusses (Hickinbotham et al., 2022).

Algorithm 1. Developmental phase.

```
Input : GRN, d, 𝒮
for i in d do
    cells_p = FEA(𝒮)
    cells_g = getGeometry(𝒮)
    cells_norm = normState(cells_p, cells_g)
    δ = []
    for cell in cells_norm do
        δ_cell = GRN(cell)
        δ.append(δ_cell)
    end
    𝒮 = GrowStructure(𝒮, δ)
end
```

3.4.1 Growth Phase

In devo, after the initial structure/seedling is generated, for example, as shown in Figure 2(b), the structure grows for a predefined number of developmental steps, where the growth is regulated by a GRN. Algorithm 1 presents the developmental algorithm that takes a genome from evo, initialized structure ($\mathcal{S}$), and the maximum number of developmental cycles (d).

During each step, local physical states of the structure are recorded, such as member forces, stress, and strain energy (cells_p), using an open source FEA tool, CalculiX (Dhondt, 2017), and node locations from structure geometry (cells_g). As discussed earlier, artificial cells are the basic building blocks of a structure, thus physical properties of the structure are divided into a number of cells. The cell properties are normalized in each developmental step. Each cell's state information, iteratively, is then fed to the GRN, which generates delta (δ). Depending on the δ generated by GRNs, the structure grows by changing the CS area of members in each cell, by moving nodes, or by adding new nodes in cells. At the end of d cycles, the fitness of the final modified structure is computed and assigned to the genome.

In the field of engineering design, these growth mechanisms are referred to as size, shape, and topology optimization (Dhondt, 2017), respectively. Although there are several other challenges involved in growing a seedling/structure, such as the representation of the structure, utilization of the global structure state information, and more, the scope of this study is limited to the search for a controller or regulator using evolutionary search algorithms.

3.4.2 Fitness and Constraint Formulation

At the end of the developmental steps, the fitness of the modified structure is calculated and assigned to the GRN. The experiments outlined in this article aim to minimize the total volume and the maximum deflection recorded at nodes. Volume is selected alongside deflection, as this represents a stiffness-to-weight optimization problem that is commonplace in engineering design, especially in structural design. When evolving designs, several constraints must be taken into consideration. These constraints are volume and maximum deflection to restrict the evolved structure from being too thin to manufacture or too heavy:

$$\begin{aligned} \min\ (f_1, f_2) &= \frac{\sum_{i=0}^{m} A_m L_m}{V_{d0}}, \frac{max[nd_0, \ldots, nd_n]}{MD_0} \\ \text{s.t. } C1 &: 1 - f_1 \leq 0 \\ C2 &: 1 - f_2 \leq 0 \end{aligned} \tag{1}$$

Equation 1 shows the two objectives (f_1, f_2) subjected to two inequality constraints ($C1$, $C2$). We assume that the structure has m members, where A_i and L_i are the CS area and length of the ith member, respectively. Assuming there are n nodes in the structure, each node's deflection (nd) is computed using FEA, and the maximum is taken as the objective. Depending on material type, loading conditions, and size of the structure, the objective values can have different ranges. Thus objectives are normalized by dividing V_{d0} and MD_{d0}, which are the volume and maximum deflection of the initial structure/seedling, respectively.

4 GRN Representations and Evolution

GRNs are complex and highly nonlinear functions; thus GRNs are represented by nonlinear models. In this work, NNs and symbolic programs are used as proxies for GRNs, where multiobjective NEAT and CGP evolve Pareto-optimal solutions.

4.1 NEAT and CGP-Based GRN Representations

In the literature, both NEAT and CGP have been used to evolve complex nonlinear systems. NEAT evolves NN architecture where, during the evolutionary phase, it randomly adds or removes nodes to the networks, and because of this, the nonlinear response of the networks can change. These random changes often result in a reduction in fitness, and so to keep diverse NN architectures in the population, speciation was introduced in NEAT for single-objective problems (Stanley & Miikkulainen, 2002). However, when NEAT is modified with NSGA-II to evolve Pareto fronts of NN architectures, the concept of speciation is not used—instead, Pareto ranking and the crowding distance-based matric maintain the diversity in the population. This diversity is based purely on the fitness values and not on the network topologies (Schrum & Miikkulainen, 2008). Deeper inspection reveals that when Pareto ranking is combined with standard NEAT, the evolved networks are not complex (i.e., evolved networks have fewer hidden nodes) and thus are not suitable for complex nonlinear problems. This is one of the drawbacks of the NEAT algorithm in a multiobjective setting.

Similar to NEAT, genetic programming has also been employed for searching nonlinear functions. However, when evolving functions or programs for intricate problems using standard genetic programming, bloating becomes a significant concern, leading to uncontrolled growth during evolutionary searches. To mitigate this issue, Cartesian genetic programming was introduced, representing a type of GP where the tree depth is predefined. When utilizing CGP, users are required to define the numbers of rows and columns based on the numbers of inputs, the number of outputs, and desired tree depth (Miller & Harding, 2008). In this setting, nodes at lower depths can receive inputs from any higher tree nodes, offering flexibility in generating various functions. However, CGP does not scale well with complex problems (O'Neill et al., 2010).

For multiobjective problems, NNs can scale well as problem complexity increases, but the NEAT algorithm is unable to evolve complex networks, whereas CGP provides flexible tree structure representation but does not scale with problem complexity. This article combines the useful features from NEAT and CGP and presents a new algorithm, as described in the next section. Figure 4 shows a NN representation using this hybrid approach, which has two inputs (I_1, I_2), one output node (O_1), and two hidden layers with two nodes in each layer. The output node can receive

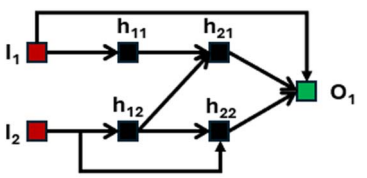

Figure 4. Example of a two-input and one-output NN encoding in CGP-style genotype representation, where a 2 × 2 grid is defined as hidden layers.

Algorithm 2. RNEAT.

Input : *popSize*, *maxItr*, *numH*, *numN*, $\mathcal{S}$
1 *numInput*, *numOutput* = *Size*($\mathcal{S}$)
2 C_l = *Chromosome Size*(*numInput*, *numOutput*, *numH*, *numN*)
3 P_0 = *Initialize Population*(*popSize*, C_l)
4 Evaluation(P_0, $\mathcal{S}$, *numInput*, *numOutput*, *numH*, *numN*)
5 **for** *t in maxItr* **do**
6 P_c = *Reproduction*(P_t)
7 *Evaluation*(P_c, $\mathcal{S}$, *numInput*, *numOutput*, *numH*, *numN*)
8 *Fronts* = *Ranking*(P_t, P_c)
9 *cwd* = *Crowding Distance*(*Fronts*)
10 P_{t+1} ← *NextGenIndividuals*(*Fronts*, *cwd*)
11 **end**
12 return *Best Front*

inputs from the input nodes or any other hidden layer nodes, and hidden layer nodes can receive inputs from any node in previous layers, similar to CGP.

4.2 RNEAT

To evolve multiobjective NN topologies in a controlled manner, a hybrid representation based on NEAT and CGP is presented, where connections, weights, and biases can be tuned using existing evolutionary search algorithms. The primary motivation of RNEAT is to evolve NNs of different architectures for multiobjective problems (Schrum & Miikkulainen, 2008; Stanley & Miikkulainen, 2002).

Algorithm 2 shows the proposed RNEAT algorithm that takes population size (popSize), maximum number of generations (maxItr), number of maximum hidden layers (numH), maximum number of nodes in a hidden layer (numN), and initial seedling/structure ($\mathcal{S}$) as inputs and returns the best-evolved Pareto solutions. To generate the initial population, first, the numbers of inputs and outputs are computed using the initial seedling/structure, then the length of the chromosome is calculated. The length depends on the type of experiment being performed. Table 1 gives the number of inputs and outputs for different experimental setups. Figure 5 shows the genome encoding of the network shown in Figure 4. Note that here each node (hidden and output) has two different types of genes: connection genes and weight genes. For example, node h_{21} can take inputs from four nodes (I_1, I_2, h_{11}, h_{12}) and thus has four connection and weight genes, which can be evolved in a multiobjective setting. Once the initial population is generated Algorithm 3 evaluates each individual's fitness. The rest of Algorithm 2 implements tournament selection, simulated binary crossover, polynomial mutation, and Pareto ranking operators, as in standard NSGA-II.

Algorithm 3 takes genomes, initial structure, number of inputs, number of outputs, hidden layers, and the number of nodes in a hidden layer and computes the fitness of each genome. Here a genome encodes the network architecture and weights between nodes. Note that, similar to CGP, the dimension of the network is predefined, whereas a real value–encoded chromosome defines the connectivity and associated weights. Line 1 of Algorithm 3 creates a fixed-size network (which is termed a GRN), then connections and weights are decoded from each genome. The network's architecture follows feed-forward information propagation, enabling nodes from previous layers to connect to the current-layer nodes. This decoded NN is the GRN that controls the growth of the initial structure in d developmental steps, as presented in Algorithm 1. The performance of the full-grown structure is computed using Equation 1.

Because RNEAT draws inspiration from NEAT and CGP, they share some common features but also have some differences. For example, NEAT relies on complexification to add hidden layer nodes and uses speciation to preserve diversity in the population, whereas RNEAT defines a fixed

Table 1. Problem specifications: structure support and load points, number of inputs, and outputs under different types of growth/update mechanisms.

	Problem			
	WT	CB	MBB	TL
Support points	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
	(200, 0, 0)	(0, 60, 0)	(200, 0, 0)	(0, 0, 20)
				(0, 20, 0)
				(0, 20, 20)
Load points	(100, 50, 0)	(180, 30, 0)	(100, 40, 0)	(20, 0, 0)
				(20, 0, 20)
				(20, 20, 0)
				(20, 20, 20)
$\mathcal{G}_1$: in		3 m_{se}, 3 m_{cs}		6 m_{se}, 6 m_{cs}
$\mathcal{G}_1$: out		3 δ_{cs}		6 δ_{cs}
$\mathcal{G}_1$, $\mathcal{G}_2$: in		3 m_{se}, 3 m_{cs}, 9 n_l		6 m_{se}, 6 m_{cs}, 12 n_l
$\mathcal{G}_1$, $\mathcal{G}_2$: out		3 δ_{cs}, 9 δ_n		6 δ_{cs}, 12 δ_n
$\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$: in		3 m_{se}, 3 m_{cs}, 9 n_l		6 m_{se}, 6 m_{cs}, 12 n_l
$\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$: out		3 δ_{cs}, 9 δ_n, 1 δ_{cd}		6 δ_{cs}, 12 δ_n, 1 δ_{cd}

Note. WT = Warren truss, CB = cantilever beam, MBB = Messerschmitt-Bolkow-Blohm, TL = trabecular lattice in a 3-D space.

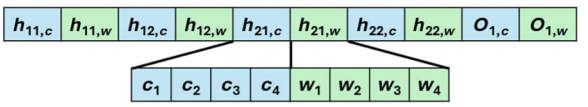

Figure 5. Genome encoding of the example network as shown in Figure 4, where each hidden and output node has two types of gene: connection genes and weight genes. The connection genes (c) are represented in blue, weight genes (w) in green. Hidden and output nodes can take inputs from all previous nodes. For example, hidden node h_{21} has four connection (c_1, c_2, c_3, c_4) and four weight (w_1, w_2, w_3, w_4) genes.

architecture, similar to CGP. NEAT focuses more on topological evolution through crossover and mutation, whereas weights and biases are only mutated. In contrast, when using RNEAT, both topology and weights get a fair chance of being searched through crossover and mutation.

5 Experimental Results and Discussion

This section presents and analyzes experimental results pertaining to studies formulated as size, shape, and topology optimization problems. Note that the growth of a structure can occur in three ways: by altering the CS area of members ($\mathcal{G}_1$), by relocating nodes ($\mathcal{G}_2$), or by adding nodes ($\mathcal{G}_3$). The objective is to evolve a GRN that utilizes local structural state information to govern the growth of the structure, with the aim of minimizing both volume and maximum deflection. To facilitate a comparative analysis, experiments on four distinct problems were conducted, employing three

Algorithm 3. Evaluation.

```
Input : genomes, S, numInput, numOutput, numH, numN
GRN = Network Architecture(numInput, numOutput, numH, numN)
for g in genomes do
    /*** Decode neural network architecture from the genome ***/
    for node in GRN.nodes() do
        node.weights = weights(g, node)
        node.connection = Connections(g, node)
    end
    /*** Growth in Developmental Phase ***/
    S = Developmental Phase(GRN, d, S)
    g_f, g_c = computePerformance(S) (using eq.1)
end
```

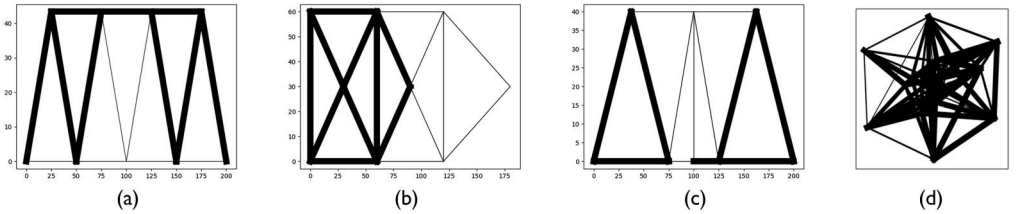

Figure 6. Initial seedling of (a) a Warren truss with seven cells, (b) a cantilever with nine cells, (c) an MBB with six cells, and (d) a 3-D trabecular lattice. These initial seedlings are designed to be sub-optimal by having different cross-sectional areas of members and will be modified by GRNs to improve the quality in terms of volume and maximum deflection.

different algorithms[2] RNEAT, NEAT, and CGP. The results obtained from these experiments are examined and discussed.

This article encompasses a total of 11 diverse experiments from three 2-D problems and one 3-D problem, where the performance of the three algorithms is compared. In the first four experimental setups, only one type of growth mechanism, $\mathcal{G}_1$, was used to update the structure design; in the next four experiments, two types of growth rules ($\mathcal{G}_1$, $\mathcal{G}_2$) were applied together, and in the last three experiments, all three growth rules were applied concurrently. Results of these diverse problems show the generalizability and scalability of the proposed approach and the effectiveness of the proposed RNEAT in evolving Pareto-optimal/near-optimal solutions or GRNs.

5.1 Experimental Setup

The experiments encompassed the evaluation of GRNs on a Warren truss (WT), a cantilever beam (CB), Messerschmitt–Bolkow–Blohm (MBB) beam structures in a 2-D space, and a trabecular lattice (TL) in a 3-D space, as shown in Figure 6. In literature, these sample structures are commonly used to test the effectiveness of new algorithms and approaches. Note that structures are made of cells, where the WT has seven cells, the CB nine cells, and the MBB six cells. Experiments were conducted considering different types and combinations of growth mechanisms, such as $\mathcal{G}_1$, $\mathcal{G}_1$, and $\mathcal{G}_2$ and $\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$. Table 1 gives support and load locations for these problems and local structure state information as input to the GRN, in addition to output deltas suggested by the GRNs. In each experiment, the loads act in the direction of the negative *y* axis.

2 We also compared CPPN (compositional pattern-producing network), but results of NEAT and CPPN were almost the same.

These experiments aimed to evolve the Pareto front of solutions using different parameters tailored to the complexity of each problem, allowing for meaningful comparisons. To assess and compare the Pareto fronts obtained through various algorithms, the hypervolume (HV) (Guerreiro et al., 2021) performance indicator was employed. The selection of HV was based on its Pareto-agnostic nature, meaning it does not rely on a true Pareto front, thus enabling effective comparisons across different algorithms.

5.2 Evolving GRNs for Size Optimization

In size optimization, the CS area of members is modified to minimize material usage. Note that a structure is made up of cells, and a GRN takes input from each cell and generates deltas to update the state of each cell iteratively. In this case, the GRN takes into account the strain energy (m_{se}) and the CS area (m_{cs}) of each member of a cell as input and provides a delta change in the CS area (δ_{cs}) for a predefined number of developmental steps. For 2-D size optimization problems, a GRN takes six inputs and provides three outputs, as shown in Table 1, while the $\mathcal{G}_1$ type of growth mechanism happens, whereas for the 3-D size optimization problem, a GRN has 12 inputs and 6 outputs.

These initial structures, shown in Figure 6, are suboptimal designs, and the aim here is to evolve a Pareto front of GRNs that can generate a diverse set of optimal or near-optimal structures. Figure 7(a) shows the Pareto fronts obtained using the three algorithms on the WT problem over 10 runs, where red dots are RNEAT-evolved solutions and blue and aqua represent NEAT and CGP solutions, respectively. Here a solution represents a GRN (either a NN or a computer program). The black dot is the initial structure fitness, and the green dot is the reference point to compute the HV performance indicator. The initial structure objectives are (1, 1) because each objective is normalized, where the initial structure objectives are taken as a reference.

Figure 7(a) shows that evolved GRNs were able to grow the structure such that both volume and maximum deflection reduced. Note that a GRN grows the structure for d developmental steps, and at the end, the fitness of the final modified structure is computed and assigned as the fitness to the GRN. Figure 7(b) shows the fitness trajectory in different devo steps, indicating that the GRN, chosen from a Pareto front, learned to minimize both objectives in the initial few steps, then compromised between the two objectives. This behavior is expected because reducing the volume often tends to increase the deflection at nodes. Incorporating each devo step fitness into the final fitness might be helpful in evolving even better quality Pareto solutions, but this is part of the future endeavors of this work. Similar experiments were conducted on the other three problems: CB, MBB, and TL.

Experimental results are presented in Figure 8, where Figures 8(a)–8(d) compare the average hypervolume obtained in each generation over 10 runs using the three algorithms. Figures 8(e)–8(h)

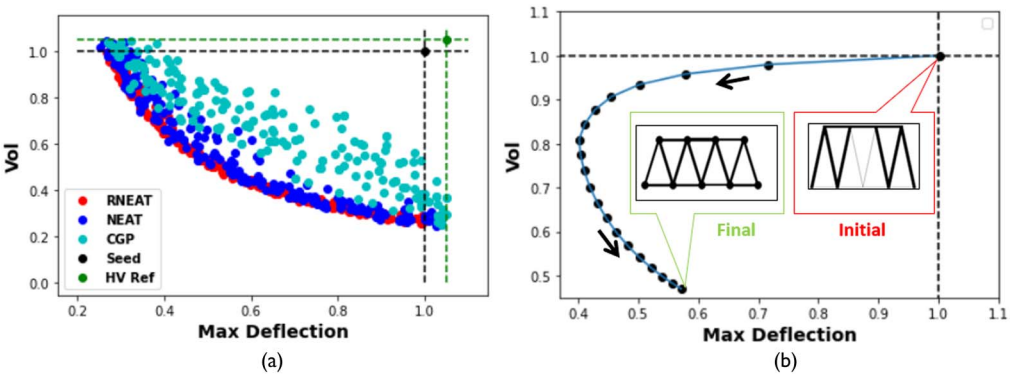

Figure 7. (a) Pareto fronts obtained using the three algorithms in 10 runs on the WT problem. (b) Example of fitness movement in different developmental steps, where the last developmental step fitness is recorded as the fitness of the genome. (a) Last generation Pareto front of 10 runs, (b) Fitness trajectory in devo steps.

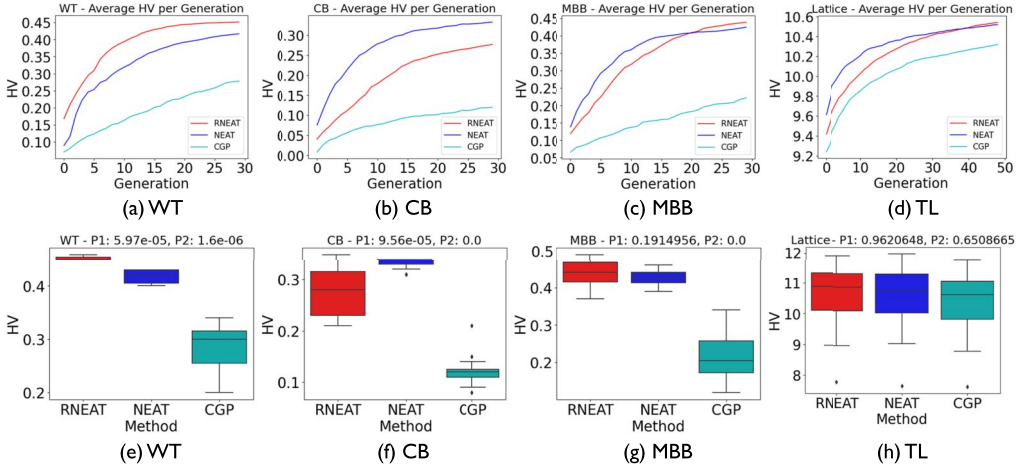

Figure 8. Size optimization. (a–d) Comparisons of the average value HV per generation obtained using different algorithms over 10 runs. (e–h) Comparisons of the distribution of HV from the last generation over 10 runs.

compare the distribution of HV values of the last-generation Pareto fronts. It is important to note that a higher HV value indicates a better-quality Pareto front (Guerreiro et al., 2021).

From the complexity perspective, the size optimization problem is the least complex compared to other experiments conducted in this article. Figures 8(a) and 8(c) demonstrate that RNEAT achieves higher HV values compared to the other two algorithms on the WT and MBB problems, indicating superior performance. However, on the CB problem, RNEAT shows inferior results compared to NEAT, but better results than CGP. The distribution figures (Figure 8(e)–8(h)), also reveal that RNEAT's performance is better than NEAT on WT and MBB, comparable on TL, and better than CGP on all four problems.

Statistical analysis has been conducted from the data obtained using three methods and is indicated in Figures 8(e)–8(h), where P1 refers to the p value calculated using the distribution of HV obtained using RNEAT and NEAT and P2 refers to the p value computed using the HV distribution obtained using RNEAT and CGP. Experimental results show that RNEAT is statistically significantly better than NEAT on the WT problem with a p value of less than 0.05 and better than CGP on WT, CB, and MBB problems, again with p values less than 0.05. Table 2 gives the best, worst, and median HVs from the last-generation Pareto front over 10 runs, indicating that on three out of four problems, the median performance of RNEAT is better than others.

Note that the three 2-D problems have different search space sizes, where WT, CB, and MBB has 15, 17, and 13 members, respectively. As the number of members in the structure increases,

Table 2. Comparing the best, median, and worst values of HV obtained from the last-generation Pareto front on four problems over 10 runs for the size optimization problem.

	WT			CB			MBB			TL		
	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
Best	**0.46**	0.43	0.34	**0.35**	0.34	0.21	**0.49**	0.46	0.34	11.91	**11.97**	11.78
Median	**0.45**	0.42	0.3	0.28	**0.33**	0.12	**0.44**	0.425	0.20	**10.87**	10.795	10.64
Worst	0.45	0.4	0.2	0.21	0.31	0.08	0.37	0.39	0.12	7.78	7.66	7.63

Note. Boldface number indicates the best result obtained using different methods.

the complexity also increases, and it would be difficult for standard GAs to manage the problems' complexity. However, the proposed evo-devo approach grows cells of the structure iteratively and is thus applicable to these complex problems. This shows the scalability property of the proposed approach. These results provide encouraging evidence that the evo-devo-based approach can effectively evolve a diverse set of Pareto-optimal GRNs that have learned how to control the growth of initial structures, resulting in simultaneous minimization of both volume and deflection.

5.3 Evolving GRNs for Topology Optimization

In a similar manner to size, the topology of the structure can be optimized with minimal modifications to the GRN's inputs and outputs. In the case of topology optimization, the GRN governs both member CS area and node movement. In this scenario, the GRN incorporates node location (n_l) information, along with member thickness (m_{cs}) and strain energy (m_{se}), to generate deltas for both node location (δ_n) and member thickness(δ_{cs}); thus two types of growth mechanisms happen together, $\mathcal{G}_1$ and $\mathcal{G}_2$. The number of inputs and outputs for topology optimization is 15 and 12, respectively, for 2-D problems and 24 and 18, respectively, for 3-D problems, as given in Table 1.

As different types of growth mechanisms are combined to update/grow a given structure, the complexity of the problem increases; however, the same GRN representations can be used to evolve diverse designs. This shows the generalizability characteristics of the proposed evo-devo approach.

Similar experiments have been conducted, and comparative results are presented. The average HV values obtained in each generation using different techniques, along with the distribution of HVs from the last-generation Pareto fronts over 10 runs, are illustrated in Figure 9. Figure 9(a) shows that on WT, RNEAT's HV is higher than the other two, and Figure 9(e) shows the distribution of last-generation HV values over 10 runs. Figure 9 also shows p values obtained using the distribution of RNEAT and NEAT HV values (P1) and RNEAT and CGP HV values (P2). Both P1 and P2 are less than 0.05, indicating statistical significance, on the WT problem. Table 3 compares the best, worst, and median HV values from the last-generation Pareto front in 10 runs. From this table, it is clear that on the WT, RNEAT is best, and median HVs are better (higher) than the others.

A similar performance comparison is made on CB, MBB, and TL. On CB and MBB, CGP is the best-performing algorithm. On TL, the performance of RNEAT is comparable to that of CGP and better than that of NEAT. However, Table 3 shows that the best and the median HV values obtained using RNEAT are higher than those for both NEAT and CGP. Statistical analysis shows that RNEAT is better than NEAT on WT, CB, and MBB problems.

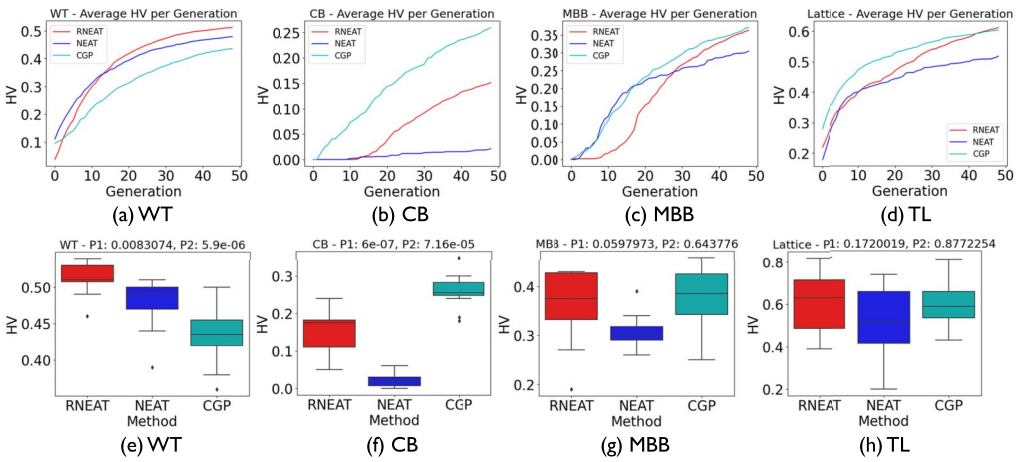

Figure 9. Results for topology optimization, comparing the average HV and distribution of HV over 10 runs when a GRN controls both node movement and edge/member thickness of structures. (a) WT, (b) CB, (c) MBB, (d) TL, (e) WT, (f) CB, (g) MBB, (h) TL.

Table 3. Comparing the best, median, and worst values of HV obtained from the last-generation Pareto front on four problems over 10 runs for the topology optimization problem.

	WT			CB			MBB			TL		
	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
Best	**0.54**	0.51	0.5	0.24	0.06	**0.35**	0.43	0.39	**0.46**	**0.82**	0.74	0.81
Median	**0.51**	0.49	0.435	0.175	0.02	**0.265**	0.38	0.3	**0.385**	**0.63**	0.52	0.59
Worst	0.47	0.39	0.36	0.05	0.0	0.19	0.19	0.26	0.26	0.39	0.2	0.43

Note. Boldface number indicates the best result obtained using different methods.

5.4 GRNs for Size, Shape, and Topology Optimization

In previous experiments, the initial structure/seedlings had more than one cell, as shown in Figure 6, and in the growth phase, only the member's thickness and node locations were changed. In this section, experiments are devised to closely mimic the evo-devo-based concept.

In the proposed evo-devo-based approach, the integration of size, shape, and topology allows for a unified problem setting, in which GRNs govern the CS area of members, the relocation of nodes, and even the addition of new nodes to modify the structure's geometry and topology. Among the various experimental setups, this particular configuration is the most complex due to the significantly larger search space compared to optimizing for size or topology alone. Here all three types of growth mechanisms ($\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$) can happen simultaneously at every growth step.

At the start of the simulation, initial seedlings are generated with minimal cells (minimum of one). In each developmental step, the GRN takes m_{cs}, m_{se}, and n_l as input and determines δ_{cs} and δ_n and whether to add a node by dividing a cell (δ_{cd}). If the cell division takes place, then new members are added to the structure, and this results in a volume increment. Owing to fitness-based selection pressure, evolution prefers solutions/GRNs that decide not to divide cells (to keep the volume lower), which, over the generations, decreases diversity in the population. Thus, to evolve a diverse set of structures, the constraint limit on this experiment is relaxed. Note that Table 1 gives the design space boundaries of three problems considered here. Throughout the simulation process, it is possible to adjust the initial design in a manner that leads to a deflection exceeding the prescribed space constraints. Such outcomes represent solutions that are deemed infeasible. When computing the HV of the rank zero Pareto front from each generation, obviously infeasible solutions were excluded, if there were any.

Experiments were conducted on two 2-D problems, CB and MBB, and on a 3-D problem, TL, where the initial seedling for CB and MBB was a single triangular cell structure and TL's initial seedling had more than one tetrahedral cell. Figure 10 shows the Pareto fronts obtained using the three algorithms over 10 runs on the 3-D TL problem. Figure 10 shows that RNEAT-evolved Pareto solutions are better than those of the other two methods. In this set of experiments, cell division adds new nodes and edges/members to the structure, resulting in a volume increment and where modifications to the CS area and node movement can lead to a reduction in deflection.

Comparative results are shown in Figure 11, where Figures 11(a)–11(c) compare the average HV per generation over 10 runs. Figures 11(d)–11(f) show the distribution of HVs from the last generation over 10 runs. Figure 11 shows that RNEAT is better than the other two on all three problems. Additionally, p values (P1 and P2) are less than 0.05, indicating that RNEAT performance is statistically significantly better than NEAT and CGP performance on CB, MBB, and TL.

Table 4 compares the best, worst, and median HV values from the last-generation Pareto front obtained using the three algorithms. For each problem and in each case, RNEAT's HV is higher than those of NEAT and CGP. These results (Figure 11 and Table 4) provide evidence that as the complexity of the problems starts increasing, RNEAT performs better than NEAT and CGP.

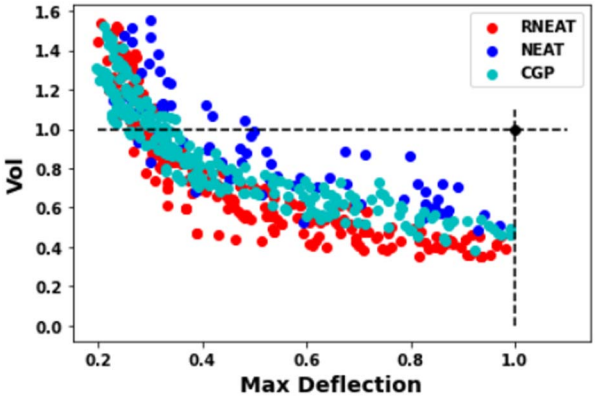

Figure 10. Last-generation Pareto fronts obtained using the three algorithms in 10 runs on the TL problem. The black circle at top right marks the fitness of the initial seedling/structure.

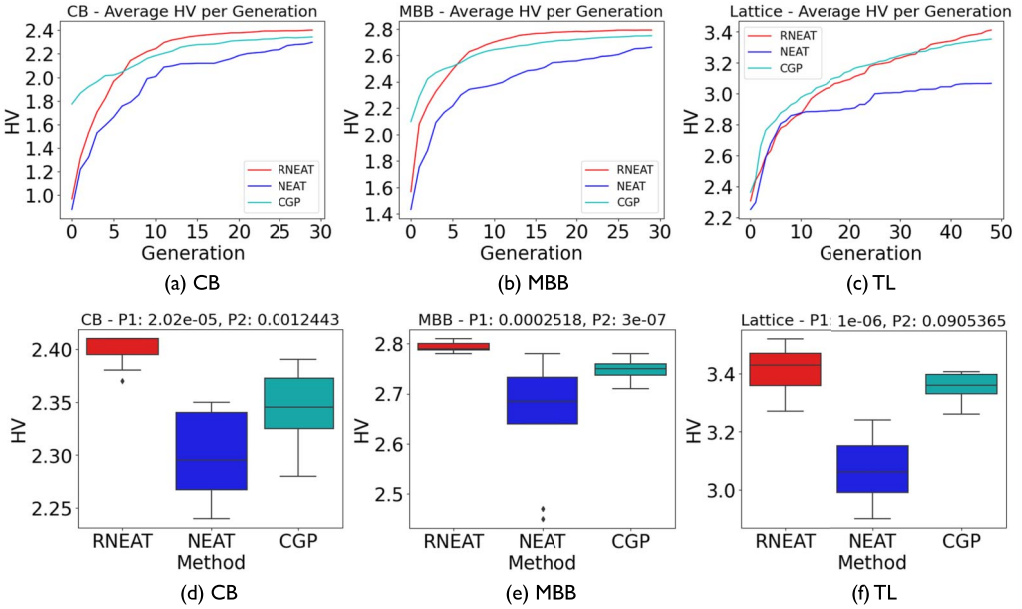

Figure 11. Results for size, shape, and topology, comparing average HV and distribution of HV over 10 runs when a GRN controls node movement, edge thickness, and cell division.

Table 4. Comparing the best, median, and worst values of HV obtained from the last-generation Pareto front on four problems over 10 runs for the combined optimization problem.

		CB			MBB			TL		
		RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
HV	Best	**2.41**	2.35	2.39	**2.81**	2.78	2.78	**3.52**	3.24	3.41
	Median	**2.41**	2.295	2.34	**2.79**	2.685	2.75	**3.43**	3.06	3.36
	Worst	2.37	2.24	2.28	2.78	2.45	2.71	3.27	2.9	3.26

Note. Boldface number indicates the best result obtained using different methods.

Table 5. Rankings of three algorithms based on the quality of evolved Pareto fronts, the distribution of HV values, and statistical analysis.

	$\mathcal{G}_1$				$\mathcal{G}_1, \mathcal{G}_2$				$\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3$		
Method	WT	CB	MBB	TL	WT	CB	MBB	TL	CB	MBB	TL
RNEAT	I	II	I	II	I	II	II	I	I	I	I
NEAT	II	I	II	I	II	III	III	II	III	III	II
CGP	III	III	III	III	III	I	I	III	II	II	III

To summarize these findings, Table 5 provides rankings for the different algorithms on all problems. These rankings are derived from results such as average HV per generation, the distribution of last-generation HV values, and statistical analysis under different experimental setups. Out of the total of 11 experiments, RNEAT outperformed the other algorithms seven times, while NEAT achieved the best performance two times and CGP emerged as the top performer twice. These promising results serve as compelling evidence that, first, evo-devo-based approaches can effectively generate Pareto fronts of GRNs that generate designs and, second, RNEAT (a hybrid of NEAT and CGP) performs well against NEAT and CGP, particularly for more complex design problems. The experimental outcomes across these four problems demonstrate the generalizability and scalability of the proposed method. Nonetheless, because RNEAT evolves nonlinear NNs as black-box models, delving into the behaviors of the evolved network will necessitate additional experimental analysis.

6 Conclusions and Future Work

This article introduces a framework based on evolutionary developmental biology (evo-devo) in which the growth of structures is controlled by GRNs represented by NNs and genetic programming. To evolve high-quality Pareto solutions (GRNs), the article presents a multiobjective neuroevolutionary algorithm termed RNEAT, which draws inspiration from NEAT and utilizes a CGP-style encoding of genotypes. The performance of RNEAT is compared to that of NEAT and CGP on various 2-D and 3-D problems, considering different initialization and types of growth. A total of 11 different experiments were conducted to assess the viability of the proposed evo-devo approach for evolving structural designs and to compare the effectiveness of RNEAT with that of the other two algorithms.

The results indicate that by considering diverse growth mechanisms and structural initialization, evolved GRNs were able to enhance performance in terms of objective functions. This provides compelling evidence that the proposed evo-devo approach can effectively facilitate the growth of designs under different environmental conditions. Evaluating the performance using the HV as a performance indicator, RNEAT outperformed the other algorithms on seven problems, whereas NEAT and CGP performed better on only two problems each. These results on the different problems under different types of growth rules show that the proposed approach is generalizable and scalable.

The experimental outcomes demonstrate the efficacy of evo-devo when local information is incorporated into a GRN. Further work will involve extending the proposed approach to evolve even more complex engineering designs, incorporating global structural information to determine local growth. Additionally, the proposed approach will be applied to problems characterized by nonlinear loading conditions, where the search space is highly constrained.

Funding Information

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) Programme under grant EP/V007335/1.

References

Albarracín-Molina, D. D., Moya, J. C., & Vico, F. J. (2016). An evo-devo system for algorithmic composition that actually works. In *Proceedings of the 2016 Genetic and evolutionary computation conference companion* (pp. 37–38). ACM. https://doi.org/10.1145/2908961.2909023

Balamurugan, R., Ramakrishnan, C. V., & Singh, N. (2008). Performance evaluation of a two stage adaptive genetic algorithm (TSAGA) in structural topology optimization. *Applied Soft Computing*, *8*(4), 1607–1624. https://doi.org/10.1016/j.asoc.2007.10.022

Bidlo, M., & Dobeš, M. (2020). Evolutionary development of growing generic sorting networks by means of rewriting systems. *IEEE Transactions on Evolutionary Computation*, *24*(2), 232–244. https://doi.org/10.1109/TEVC.2019.2918212

Chi, H., Zhang, Y., Tang, T. L. E., Mirabella, L., Dalloro, L., Song, L., & Paulino, G. H. (2021). Universal machine learning for topology optimization. *Computer Methods in Applied Mechanics and Engineering*, *375*, Article 112739. https://doi.org/10.1016/j.cma.2019.112739

Christensen, P. W., & Klarbring, A. (2008). *An introduction to structural optimization* (Vol. 153). Springer Science & Business Media.

Cussat-Blanc, S., Harrington, K., & Banzhaf, W. (2019). Artificial gene regulatory networks—a review. *Artificial Life*, *24*(4), 296–328. https://doi.org/10.1162/artl_a_00267, PubMed: 30681915

Cussat-Blanc, S., Harrington, K., & Pollack, J. (2015). Gene regulatory network evolution through augmenting topologies. *IEEE Transactions on Evolutionary Computation*, *19*(6), 823–837. https://doi.org/10.1109/TEVC.2015.2396199

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Delgado, F. M., & Gómez-Vela, F. (2019). Computational methods for gene regulatory networks reconstruction and analysis: A review. *Artificial Intelligence in Medicine*, *95*, 133–145. https://doi.org/10.1016/j.artmed.2018.10.006, PubMed: 30420244

Dhondt. (2017). *CalculiX CrunchiX user's manual version 2.12*. http://www.dhondt.de/ccx_2.12.pdf

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms* (Vol. 1). Springer. https://doi.org/10.1007/978-1-4757-3643-4_12

Guerreiro, A. P., Fonseca, C. M., & Paquete, L. (2021). The hypervolume indicator: Computational problems and algorithms. *ACM Computing Surveys*, *54*(6), 1–42. https://doi.org/10.1145/3453474

Hall, B. K. (2012). Evolutionary developmental biology (evo-devo): Past, present, and future. *Evolution: Education and Outreach*, *5*(2), 184–193. https://doi.org/10.1007/s12052-012-0418-x

Hickinbotham, S., Dubey, R., Friel, I., Colligan, A., Price, M., & Tyrrell, A. (2022). Evolving design modifiers. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1052–1058). IEEE. https://doi.org/10.1109/SSCI51031.2022.10022087

Kaldis, P. (2016). Quo vadis cell growth and division? *Frontiers in Cell and Developmental Biology*, *4*, Article 95. https://doi.org/10.3389/fcell.2016.00095, PubMed: 27626033

Karlebach, G., & Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, *9*(10), 770–780. https://doi.org/10.1038/nrm2503, PubMed: 18797474

Mccormack, J., & Gambardella, C. C. (2022). Growing and evolving 3-D prints. *IEEE Transactions on Evolutionary Computation*, *26*(1), 88–99. https://doi.org/10.1109/TEVC.2021.3095156

Miller, J. F., & Harding, S. L. (2008). Cartesian genetic programming. In M. Keijzer (Ed.), *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation* (pp. 2701–2726). ACM. https://doi.org/10.1145/1388969.1389075

Navarro-Mateu, D., & Cocho-Bermejo, A. (2019). Evo-devo algorithms: Gene-regulation for digital architecture. *Biomimetics*, *4*(3), Article 58. https://doi.org/10.3390/biomimetics4030058, PubMed: 31412569

Navarro-Mateu, D., & Cocho-Bermejo, A. (2020). Evo-devo strategies for generative architecture: Colour-based patterns in polygon meshes. *Biomimetics*, *5*(2), Article 23. https://doi.org/10.3390/biomimetics5020023, PubMed: 32456101

Olson, E. N. (2006). Gene regulatory networks in the evolution and development of the heart. *Science*, *313*(5795), 1922–1927. https://doi.org/10.1126/science.1132292, PubMed: 17008524

O'Neill, M., Vanneschi, L., Gustafson, S., & Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, *11*, 339–363. https://doi.org/10.1007/s10710-010-9113-2

Osaba, E., Villar-Rodriguez, E., Del Ser, J., Nebro, A. J., Molina, D., LaTorre, A., Suganthan, P. N., Coello, C. A. C., & Herrera, F. (2021). A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. *Swarm and Evolutionary Computation*, *64*, Article 100888. https://doi.org/10.1016/j.swevo.2021.100888

Perez, R. E., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers and Structures*, *85*(19–20), 1579–1588. https://doi.org/10.1016/j.compstruc.2006.10.013

Price, M., Zhang, W., Friel, I., Robinson, T., McConnell, R., Nolan, D., Kilpatrick, P., Barbhuiya, S., & Kyle, S. (2022). Generative design for additive manufacturing using a biological development analogy. *Journal of Computational Design and Engineering*, *9*(2), 463–479. https://doi.org/10.1093/jcde/qwac016

Richards, D., Dunn, N., & Amos, M. (2012). An evo-devo approach to architectural design. In T. Soule (Ed.), *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (pp. 569–576). ACM. https://doi.org/10.1145/2330163.2330244

Schlitt, T., & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, *8*(Suppl. 6), 1–22. https://doi.org/10.1186/1471-2105-8-S6-S9, PubMed: 17903290

Schrum, J., & Miikkulainen, R. (2008). Constructing complex NPC behavior via multi-objective neuroevolution. In *Proceedings of the AAAI conference on Artificial intelligence and interactive digital entertainment* (Vol. 4, pp. 108–113). AAAI. https://doi.org/10.1609/aiide.v4i1.18681

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811, PubMed: 12180173

Streichert, F., Planatscher, H., Spieth, C., Ulmer, H., & Zell, A. (2004). Comparing genetic programming and evolution strategies on inferring gene regulatory networks. In *Genetic and evolutionary computation—GECCO 2004: Genetic and evolutionary computation conference, Seattle, WA, USA, June 26–30, 2004, Proceedings, Part I* (pp. 471–480). Springer. https://doi.org/10.1007/978-3-540-24854-5_47

Swain, M., Hunniford, T., Mandel, J., Palfreyman, N., & Dubitzky, W. (2005). Modeling gene-regulatory networks using evolutionary algorithms and distributed computing. In *CCGrid 2005: IEEE international symposium on Cluster computing and the grid, 2005* (Vol. 1, pp. 512–519). IEEE. https://doi.org/10.1109/CCGRID.2005.1558596

Turner, A. J., & Miller, J. F. (2014). Cartesian genetic programming: Why no bloat? In *Genetic programming: 17th European conference, EuroGP 2014, Granada, Spain, April 23–25, 2014, Revised Selected Papers* (pp. 222–233). Springer. https://doi.org/10.1007/978-3-662-44303-3_19

van Willigen, W. H., Haasdijk, E., & Kester, L. J. (2013). Evolving intelligent vehicle control using multi-objective NEAT. In *2013 IEEE symposium on Computational intelligence in vehicles and transportation systems (CIVTS)* (pp. 9–15). IEEE. https://doi.org/10.1109/CIVTS.2013.6612283

Vujovic, V., Rosendo, A., Brodbeck, L., & Iida, F. (2017). Evolutionary developmental robotics: Improving morphology and control of physical robots. *Artificial Life*, *23*(2), 169–185. https://doi.org/10.1162/ARTL_a_00228, PubMed: 28513207

Wang, S., Tai, K., & Wang, M. (2006). An enhanced genetic algorithm for structural topology optimization. *International Journal for Numerical Methods in Engineering*, *65*(1), 18–44. https://doi.org/10.1002/nme.1435

Wu, R., Chao, F., Zhou, C., Huang, Y., Yang, L., Lin, C.-M., Chang, X., Shen, Q., & Shang, C. (2022). A developmental evolutionary learning framework for robotic Chinese stroke writing. *IEEE Transactions on Cognitive and Developmental Systems*, *14*(3), 1155–1169. https://doi.org/10.1109/TCDS.2021.3098229

Xu, R., Venayagamoorthy, G. K., & Wunsch, D. C., II. (2007). Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Networks*, *20*(8), 917–927. https://doi.org/10.1016/j.neunet.2007.07.002, PubMed: 17714912

Yildiz, A. R. (2013). Comparison of evolutionary-based optimization algorithms for structural design optimization. *Engineering Applications of Artificial Intelligence*, *26*(1), 327–333. https://doi.org/10.1016/j.engappai.2012.05.014

Zou, F., Chen, D., Liu, H., Cao, S., Ji, X., & Zhang, Y. (2022). A survey of fitness landscape analysis for optimization. *Neurocomputing*, *503*, 129–139. https://doi.org/10.1016/j.neucom.2022.06.084