Article:
Dubey, Rahul orcid.org/0000-0003-1524-7797, Hickinbotham, Simon John orcid.org/0000-0003-0880-4460, Colligan, Andrew et al. (4 more authors) (2024) Evolving Novel Gene Regulatory Networks for Structural Engineering Designs. Artificial Life. ISSN 1064-5462

https://doi.org/10.1162/artl_a_00448

Artificial Life Manuscript Submission

Evolving Novel Gene Regulatory Networks for Structural Engineering Designs

Rahul Dubey [1], Simon Hickinbotham [1], Andrew Colligan [2], Imelda Friel [2], Edgar Buchanan [1], Mark Price [2], Andy M Tyrrell [1]

Corresponding: Rahul Dubey (rahul.dubey@york.ac.uk)

1. School of Physics, Engineering and Technology, University of York, UK
2. School of Mechanical and Aerospace Engineering, Queen's University Belfast, NI

Abstract. Engineering design optimization poses a significant challenge, usually requiring human expertise to discover superior solutions. While 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 paper 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 multi-cellular organisms through the use of Gene Regulatory Networks (GRNs). GRNs are inherently complex and highly nonlinear, and this paper explores the use of neural networks and genetic programming as artificial representations of GRNs to emulate such behaviors. In order to evolve a wide range of Pareto fronts for artificial GRNs, this paper introduces a new technique, a real-value encoded neuro-evolutionary method termed "real-encoded NEAT" (RNEAT). The performance of RNEAT is compared with two well-known evolutionary search techniques across different 2D and 3D problems. The experimental results demonstrate two key findings: Firstly, the proposed framework effectively generates a population of GRNs that can produce diverse structures for both 2D and 3D problems. Secondly, the proposed RNEAT algorithm outperforms its competitors on more than 50% of the problems examined.

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 like 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 multi-organ 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 the most crucial component of the evolutionary developmental cycle.

This paper takes inspiration from the Evo-Devo concept to generate different engineering designs by evolving an artificial gene regulatory network. 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 gene regulatory networks 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 in order 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 non-linear and thus different non-linear models have been studied to mimic the behavior of GRNs.

In this paper, an artificial gene regulatory network is represented by two different models: neural networks (NN) and computer programs encoded in graphs where problems are formulated as multi-objective optimization problems. The nonlinearity of both NNs and graphs can be varied by changing their hyper-parameters. Since the problems are multi-objective in nature, two different evolutionary search methods have been employed: multi-objective cartesian genetic programming (CGP) (Miller & Harding, 2008), and multi-objective neuro-evolution of augmenting topologies (NEAT) (Stanley & Miikkulainen, 2002) are used to evolve complex computer programs and neural networks respectively. However, extending NEAT to evolve for multi-objective problems while maintaining speciation poses a significant challenge (van Willigen et al., 2013). To evolve multi-objective neural network topologies, this paper 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 2D and 3D 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, node locations, as well as information obtained through finite element analysis (FEA), such as member force or strain energy (SE). 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 its competitors on more than 50% of the problems, indicating its superiority in terms of effectiveness and performance.

The two major contributions of this paper are as follows: 1) the paper introduces a generalizable and scalable approach based on the Evo-Devo concept to evolve diverse Pareto front engineering designs, and 2) the paper presents RNEAT by taking inspiration from NEAT and CGP to evolve multi-objective NNs. The rest of this paper 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 are characterized by a nonlinear relationship between parameters that generate phenotypes (Zou et al., 2022). This paper 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 in 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, Vuk (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 (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. Jon (Mccormack & Gambardella, 2022) presented “growing and evolving 3D printable designs” where a covariance matrix adaptation evolutionary strategies algorithm (CMA-ES) tunes five genetic parameters for optimizing 3D printable structures. Bidlo (Bidlo & 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 onto comparator structures which 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 paper 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 investigated significantly to evolve designs.

The GRN plays a crucial role in the development of organisms 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; Karlebach & Shamir, 2008; Schlitt & Brazma, 2007). These techniques range from simple logical functions such as Boolean functions to complex non-linear functions represented by neural networks. In the literature, various evolutionary and swarm algorithms have been used to evolve GRNs. In 2005 Swain (Swain et al., 2005) used evolutionary algorithms to generate computational models of GRNs using data obtained from observations. Xu (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 neural networks to model the behavior of a GRN on time series data-set, and Sylvain (Cussat-Blanc et al., 2015) used NEAT to evolve neural network topologies. In a manner similar to neural networks, computer graphs (evolving using genetic programming) have also been studied as a representation for GRN e.g. (Streichert et al., 2004). In this paper, neural networks and genetic programming are considered proxies for artificial gene regulatory networks 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 GPs suffer from bloating whereas CGP does not (Turner & Miller, 2014).

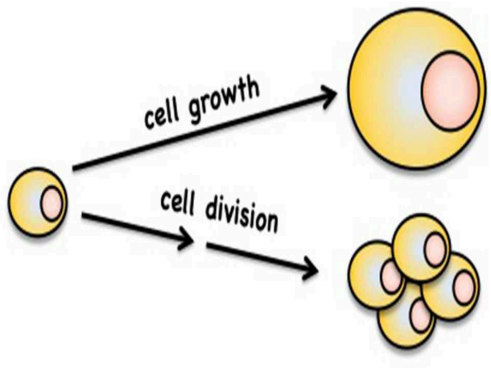
Evolving multi-objective neural architectures has been recognized as a challenging problem. Willigen (van Willigen et al., 2013) modified standard NEAT using strength Pareto evolutionary algorithms (SPEA2) to evolve multi-objective network topologies. However, in order to preserve speciation in the population, fitness-based domination is used to convert multi-objective fitness into single-objective fitness and thus is not a true multi-objective NEAT. Schrum (Schrum & Miikkulainen, 2008) used NSGA2 (Deb et al., 2002) to modify NEAT and evolved different topologies. The concept of speciation is not used here because fitness sharing in multi-objectives is difficult, as there is more than one objective value. This paper 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, the better chance of searching for optimal solutions. As the grid resolution increases, the search space increases, and thus these approaches are not scalable as well as generalizable. The next section describes in detail the new algorithm and framework that form the foundation work of this paper.

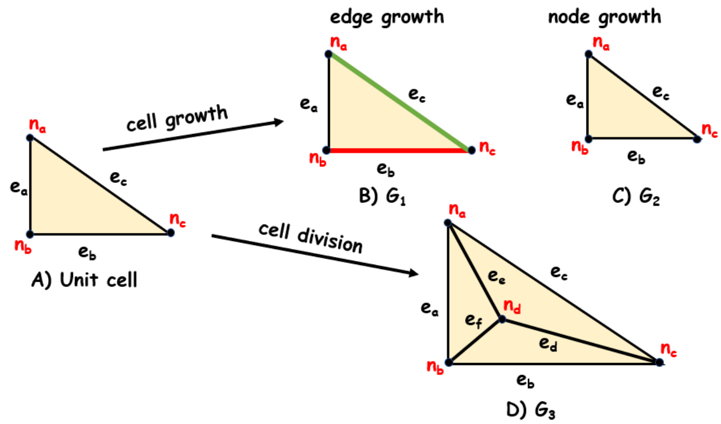
3 Methodology

3.1 Biological to Structural Cell Analogy

As biological cells are the basic building block of any organ (Kaldis, 2016), this paper considers a triangular shape cell for 2D problems and a tetrahedral shape cell for 3D 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) biological cell growth and division (Kaldis, 2016)



(b) 2D triangular cell, its growth, and division

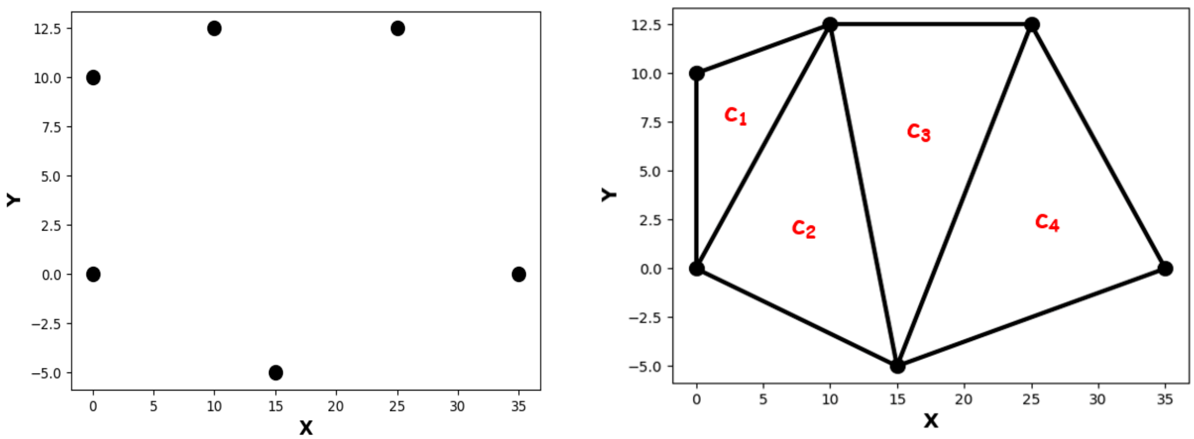
Figure 1: Biological and structural cell analogy. Figure show (a) an example biological unit cell, its growth, and division, (b) show examples of cell growth in three different types of growth mechanisms: $\mathcal{G}_1$ edge growth, $\mathcal{G}_2$ node growth, and $\mathcal{G}_3$ cell division.

a set of GRNs based on local environmental factors.

Figure 1(b) shows an artificial cell, its growth, and multiplication mechanisms. An artificial 2D triangular shape 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, cell division is also referred to as a type of growth mechanism here. A cell grows in one of the following ways: edge growth ($\mathcal{G}_1$), node growth ($\mathcal{G}_2$), and cell division ($\mathcal{G}_3$) as shown in Figure 1(b). A GRN takes the state information of a cell (such as the cross-section area of edges, locations of nodes, and other properties computed using finite element analysis 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)(B) where the red line shows an increment in CS area, 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

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



(a) Nodes

(b) Structure with four cells

Figure 2: Illustrates an example of the initial seedling/structure generated using a set of points, (a) shows a set of points which are locations of supports and loads, and (b) shows the resulting geometry with four cells (c_1, c_2, c_3, c_4) using Delaunay triangulation.

mechanism ($\mathcal{G}_2$) updates the location of the nodes as shown in Figure 1(b)(C). 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 by placing a node (n_d) at the centroid of the parent cell Figure 1(b)(D). 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 3D 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 2D initial structure is shown in Figure 2 where (a) shows the location of points, and (b) shows the resulting geometry. The initial

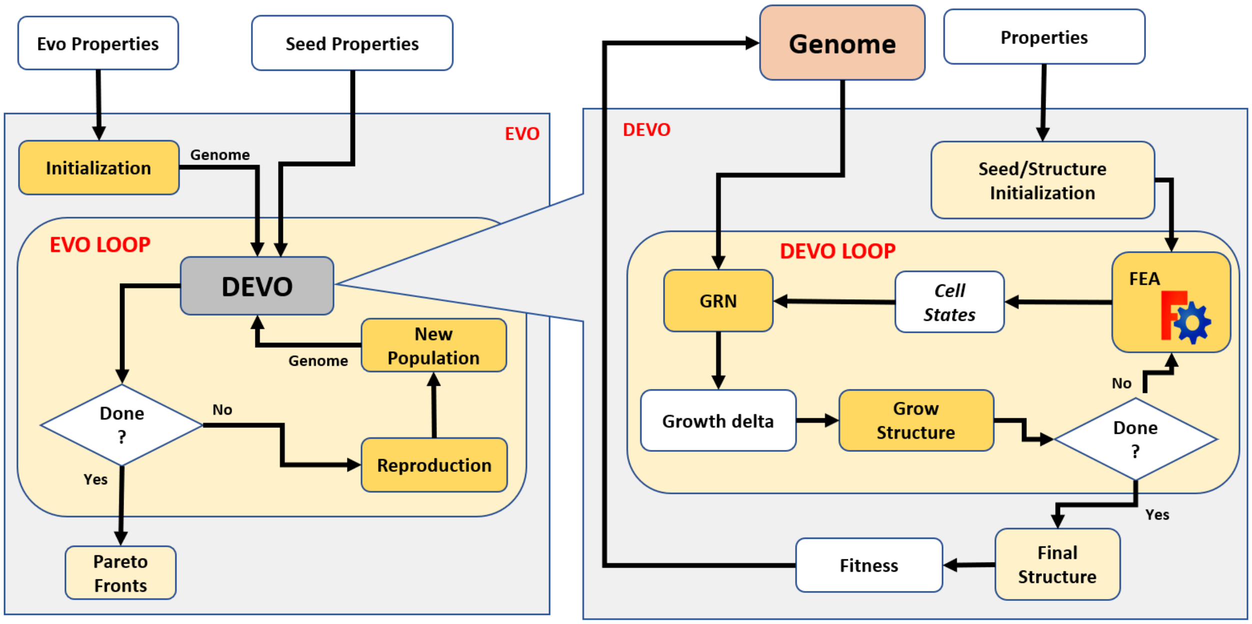


Figure 3: Block diagram representation of the Evo-Devo framework for growing structural design where Evo evolves solutions/GRNs and Devo grows the structure for a predefined number of developmental steps for each GRN. A GRN controls the growth and at the end, the physical properties of the structure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

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. 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. Based on 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 multi-objective evolutionary search techniques evolve a Pareto front of GRNs that generates diverse designs. Section 4 discusses in detail GRNs representation and the proposed hybrid evolutionary search algorithm.

3.4 Devo: Growing Structure

In the Devo component, a structure undergoes in a growth phase through various mechanisms such as altering cross-sectional areas, relocating nodes, adding new nodes, or combinations thereof. Within the evo-devo framework, Evo provides an indirect encoding of growth rules, while Devo carries out the actual growth process based on local environmental factors. Figure 3, right, shows the block diagram representation of the Devo component. Here, first, a structure is generated using pre-defined 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 on simple engineering design problems such as brackets (Price et al., 2022) and trusses (Hickinbotham et al., 2022).

Algorithm 1: Developmental Phase

Input : $GRN, d, \mathcal{S}$

for *i in d* **do**
  $cells_p$ = FEA($\mathcal{S}$)
  $cells_g$ = getGeometry($\mathcal{S}$)
  $cells_{norm}$ = normState($cells_p$, $cells_g$)
  δ = []
  for $cell$ *in* $cells_{norm}$ **do**
    $\delta_{cell} = GRN(cell)$
    $\delta.append(\delta_{cell})$
  end
  $\mathcal{S}$ = GrowStructure($\mathcal{S}$, 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 finite element analysis (FEA) tool, CalculiX (Dhondt, 2017) and node locations from structure geometry ($cells_g$). As discussed earlier, artificial cells are the basic building block 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 that generates delta (δ). Depending on the δ generated by GRNs, the structure grows by changing the cross-section 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 Constraints 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 paper aim to minimize the total volume and minimize the maximum deflection recorded at nodes. Volume is selected alongside deflection as this represents a stiffness to weight optimization problem which is commonplace in engineering design especially in structural design. When evolving designs, several constraints must be taken into consideration. These constraints are volume and max deflection to restrict the evolved structure from being too thin to manufacture or too heavy.

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

Equation 1 shows the two objectives (f_1, f_2) subjected to two inequality constraints ($C1$, $C2$). Assuming the structure has m members where A_i and L_i are the cross-sectional area and length of the i^{th} member respectively. Assuming there are n nodes in the structure, each node's deflection (nd) is computed using finite element analysis, 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 max deflection of the initial structure/seedling respectively.

4 GRN Representations and Evolution

Gene regulatory networks are complex and highly non-linear functions, thus GRNs are represented by non-linear models. In this work, neural network and symbolic programs are used as proxies for GRNs where muti-objective 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 non-linear systems. NEAT evolves neural network architecture where during the evolutionary phase, it randomly adds or removes nodes to the networks, and due to this, the non-linear 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 neural network 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 purely based 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 non-linear problems. This is one of the drawbacks of the NEAT algorithm in a multi-objective setting.

Similar to NEAT, genetic programming has also been employed for searching non-linear functions. However, when evolving functions or programs for intricate problems using standard GP, 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,

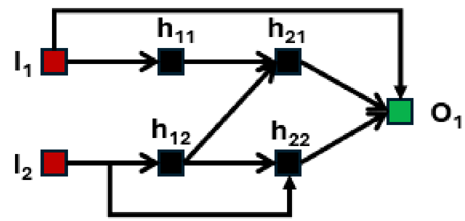


Figure 4: An example of a two-input and one-output neural network encoding in CGP style genotype representation where a grid 2×2 is defined as hidden layers.

users are required to define the number of rows and columns based on the number of inputs, 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 multi objective problems, neural networks can scale well as problem complexity increases, but NEAT algorithm is unable to evolve complex networks, whereas CGP provides flexible tree structure representation but does not scale with problem complexity. This paper combines the useful features from NEAT and CGP, and presents a new algorithm as described in the next subsection. Figure 4 shows a neural network 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 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 Real-value encoded NEAT (RNEAT)

To evolve multi-objective neural network topologies in a controlled manner, a hybrid representation based on NEAT and CGP is presented where connection, weights, and biases can be tuned using existing evolutionary search algorithms. The primary motivation of RNEAT is to evolve neural networks of different architectures for multi-objective problems.

Algorithm 2: RNEAT Algorithm

Input : $popSize, maxItr, numH, numN, \mathcal{S}$

$numInput,\ numOutput = Size(\mathcal{S})$
$C_l = Chromosome\ Size(numInput, numOutput, numH, numN)$
$P_0 = Initialize\ Population(popSize, C_l)$
Evaluation($P_0, \mathcal{S}, numInput, numOutput, numH, numN$)
for *t in maxItr* **do**
    $P_c = Reproduction(P_t)$
    $Evaluation(P_c, \mathcal{S}, numInput, numOutput, numH, numN)$
    $Fronts = Ranking(P_t, P_c)$
    $cwd = Crowding\ Distance(Fronts)$
    $P_{t+1} \leftarrow NextGenIndividuals(Fronts, cwd)$
end
return $Best\ Front$

Algorithm 3: Evaluation

Input : $genomes, \mathcal{S}, numInput, numOutput, numH, numN$

$GRN = Network\ Architecture(numInput, numOutput, numH, numN)$
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 ***/
    $\mathcal{S} = Developmental\ Phase(GRN, d, \mathcal{S})$
    $g_f, g_c = computePerformance(\mathcal{S})$ (using eq.1)
end

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 (S) as inputs, and returns the best evolved Pareto solutions. To generate the initial population, first, the number of inputs and outputs are computed using the initial seedling/structure, and then the length of the chromosome is calculated. The length depends on the type of experiment being performed. Table 1 shows the number of inputs and outputs for different experimental setup. Figure 5 shows the genome encoding of the network shown

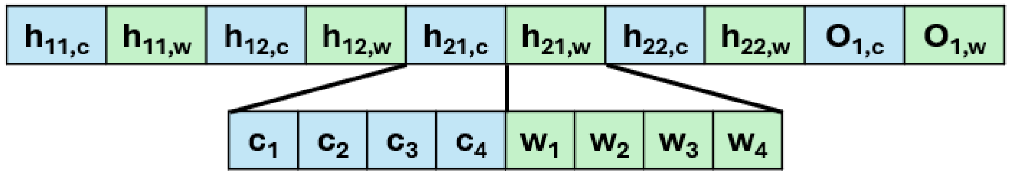
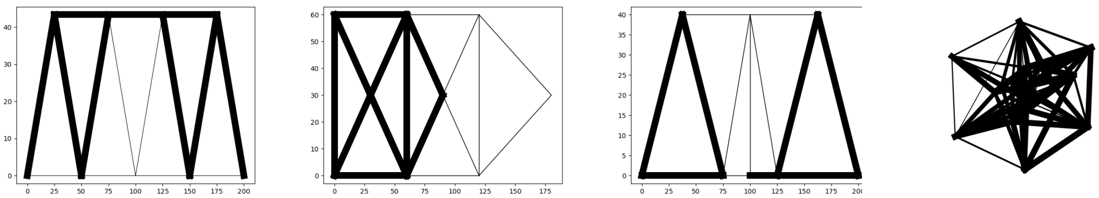


Figure 5: Genome encoding of the example network as shown in Figure 4 where each hidden and output node has two types of genes: connection genes and weight genes. The connection genes (c) are represented by blue color and weight genes (w) by green color. Hidden and output nodes can take inputs from all 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.

in Figure 4. Note that here each node (hidden and output) has two different types of genes: connections 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 multi objective 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 3takes genomes, initial structure, number of inputs, outputs, hidden layers, 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 pre-defined whereas a real-value encoded chromosome defines the connectivity and associated weights. Line 1 of the Algorithm 3 creates a fixed size network (which is termed as GRN), and 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 neural network is the gene regulatory network that controls the growth of the initial structure in d developmental steps as presented in Algorithm 1. The performance of the full grown structure in computed using eq. 1.

Since RNEAT draws inspiration from NEAT and CGP, it shares some common features, but



(a) WT (b) CB (c) MBB (d) TL

Figure 6: Initial seedling of (1) Warren truss with seven cells, (b) cantilever with nine cells, (c) MBB with six cells, and (d) 3D 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.

also has some differences. For example, NEAT relies on complexification to add hidden layer nodes and uses speciation to preserve the diversity in the population, whereas RNEAT defines a fixed 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 cross-sectional 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 gene regulatory network 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, and employed three different algorithms[1] RNEAT, NEAT, and CGP. The results obtained from these experiments are examined and discussed.

[1]We also compared CPPN but results of NEAT and CPPN were almost the same.

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

	Problems			
	WT	CB	MBB	3L
Support Points	(0, 0, 0) (200, 0, 0)	(0, 0, 0) (0, 60, 0)	(0, 0, 0) (200, 0, 0)	(0, 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 $\mathcal{G}_1$: Out	3 m_{se}, 3 m_{cs} 3 δ_{cs}			6 m_{se}, 6 m_{cs} 6 δ_{cs}
$\mathcal{G}_1$, $\mathcal{G}_2$: In $\mathcal{G}_1$, $\mathcal{G}_2$: Out	3 m_{se}, 3 m_{cs}, 9 n_l 3 δ_{cs}, 9 δ_n			6 m_{se}, 6 m_{cs}, 12 n_l 6 δ_{cs}, 12 δ_n
$\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$: In $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$: Out	3 m_{se}, 3 m_{cs}, 9 n_l 3 δ_{cs}, 9 δ_n, 1 δ_{cd}			6 m_{se}, 6 m_{cs}, 12 n_l 6 δ_{cs}, 12 δ_n, 1 δ_{cd}

This paper encompasses a total of 11 diverse experiments from three 2D problems and one 3D problem, wherein 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 on these diverse problems show the generalizability and the scalability of the proposed approach and the effectiveness of the proposed RNEAT in evolving Pareto optimal/near-optimal solutions/GRNs.

5.1 Experimental Set-up

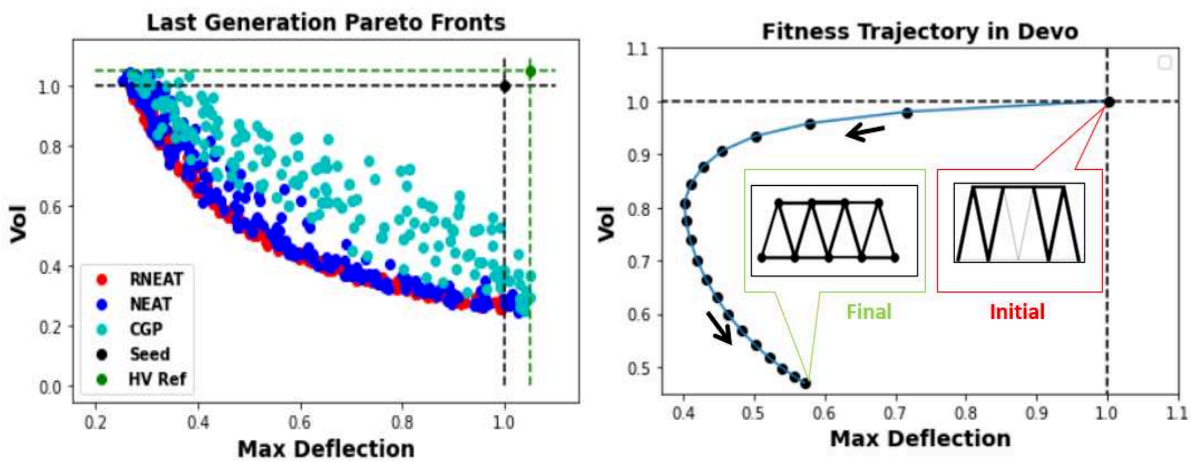
The experiments encompassed the evaluation of GRNs on a Warren Truss (WT), a Cantilever Beam (CB), Messerschmitt-Bolkow-Blohm (MBB) beam structures in a 2D space, as well as a Trabecular Lattice (TL) in a 3D space as shown in Figure 6. In literature, these sample structures are commonly used to test the effectiveness of new algorithms and approaches. Note that structures are made of cells, where the WT has seven cells, CB nine cells, and MBB has 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 shows support and load locations for these problems, and local structure state information as input to GRN, in addition to output deltas suggested by the GRNs. In each experiment, the loads act in the direction of the negative y-axis.

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

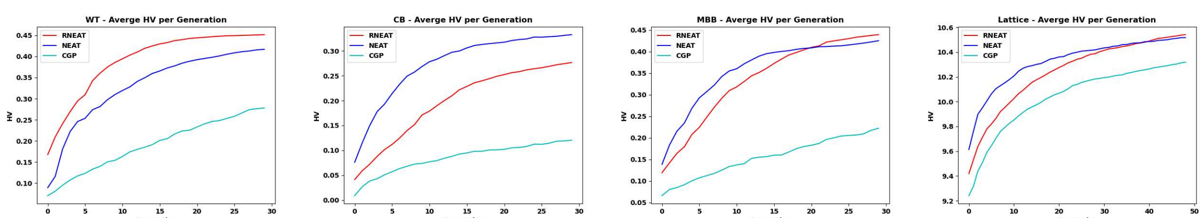
In size optimization, the CS area of members is modified to minimize material usage. Note that, a structure is made up of cells, and a GRN takes input from each cell and generates deltas to update the state of each cell iteratively. In this case, the gene regulatory network takes into account the strain 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 2D size optimization problems, a GRN takes six inputs and provides three outputs as shown in Table 1 while $\mathcal{G}_1$ type of growth mechanism happens whereas, for the 3D size optimization problem, a GRN has 12 inputs and six 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, blue and aqua represents NEAT and CGP solutions respectively. Here a solution represents a GRN (ei-

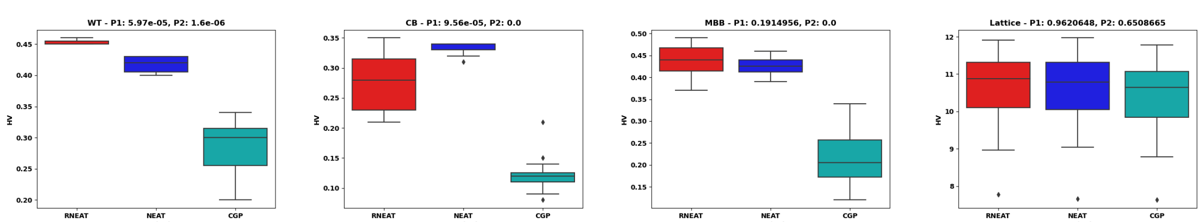


(a) Last generation Pareto front of 10 runs (b) Fitness trajectory in devo steps

Figure 7: (a) the Pareto fronts obtained using the three algorithms in 10 runs on the WT problem, and (b) an example of fitness movement in different developmental steps where the last developmental step fitness is recorded as the fitness of the genome.



(a) WT (b) CB (c) MBB (d) TL



(e) WT (f) CB (g) MBB (h) TL

Figure 8: Size optimization: Figures (a), (b), (c), and (d) compare the average value HV per generation obtained using different algorithms over 10 runs, and (e), (f), (g), and (h) compare the distribution of HV from the last generation over 10 runs.

ther 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 hyper-volume 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 max deflection reduces. Note that a GRN grows the structure for d developmental steps and the 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 GRN, chosen from a Pareto front, learned to minimize both objectives in the initial few steps and then a compromise has been made 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 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 subfigures (a), (b), (c) and (d) compare the average hypervolume obtained in each generation over 10 runs using the three algorithms. Subfigures (e), (f), (g), and (h) 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 paper. Figures 8(a) and (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 CB problem, RNEAT shows inferior results compared to NEAT, but better than CGP. The distribution figures, Figure 8 (e, f, g, and 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.

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 size optimization problem

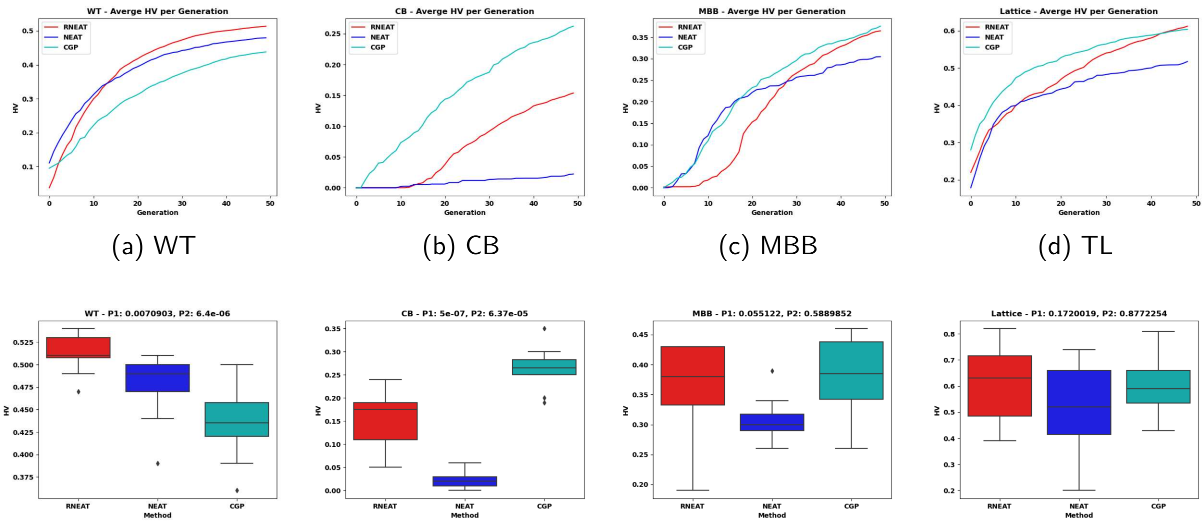
	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
	WT			CB			MBB			TL		
Best	**0.46**	0.43	0.34	**0.35**	0.34	0.21	**0.49**	0.46	0.34	11.91	**11.97**	11.78
Median	**0.45**	0.42	0.3	0.28	**0.33**	0.12	**0.44**	0.425	0.20	**10.87**	10.795	10.64
Worst	0.45	0.4	0.2	0.21	0.31	0.08	0.37	0.39	0.12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

Statistical analysis has been conducted from the data obtained using three methods and is indicated in Figures 8(e, f, g, and h) where P1 refers to the p-value calculated using the distribution of HV obtained using RNEAT and NEAT, and P2 when the p-value is computed using the HV distribution obtained using RNEAT and CGP. Experimental results show that RNEAT is statistically significantly better than NEAT on 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 shows 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 2D problems have different search space sizes where WT, CB, and MBB have 15, 17, and 13 members respectively. As the number of members in the structure increases, the complexity also increases, and it would be difficult for standard GAs to manage the problem's 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 GRN's inputs and outputs. In the case of topology optimization, the GRN



(a) WT (b) CB (c) MBB (d) TL

(e) WT (f) CB (g) MBB (h) TL

Figure 9: Results for topology optimization. The figure compares the average HV and distribution of HV over 10 runs when a GRN control both node movement and edge/member thickness of structures.

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 mechanism happen together, $\mathcal{G}_1$, and $\mathcal{G}_2$. The number of inputs and outputs for topology optimization is 15 and 12 respectively for 2D problems, and 24 and 18 for 3D problems as shown 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

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 topology optimization problem

	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
	WT			CB			MBB			TL		
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

Figure 9 (e) shows the distribution of last generation HV values over 10 runs. The figure 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's 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 CGP and better than NEAT. However, Table 3 shows that the best and the median HV values obtained using RNEAT are higher than both NEAT and CGP. Statistical analysis shows that RNEAT is better than NEAT on WT, CB, and MBB problems.

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 subsection, 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, wherein GRNs govern the cross-sectional 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 con-

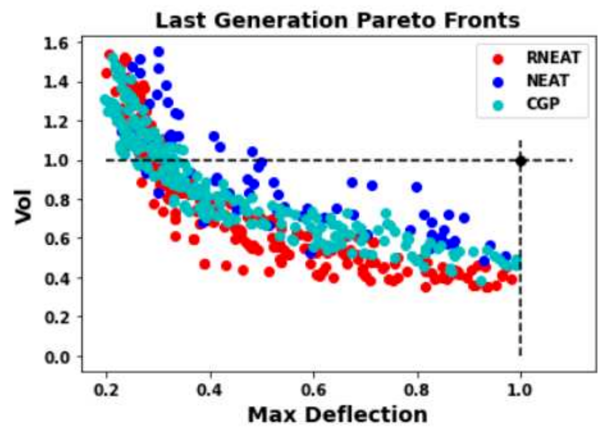
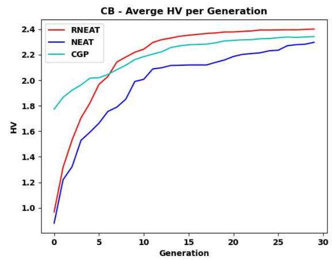


Figure 10: Figure shows the Pareto fronts obtained using the three algorithms in 10 runs on the TL problem. The black dot on the top right show the fitness of the initial seedling/structure.

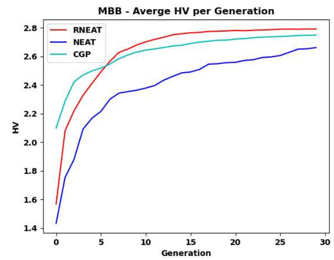
figuration is the most complex due to the significantly larger search space compared to optimizing for size or topology alone. Here all three types of growth mechanisms ($\mathcal{G}_1$, $\mathcal{G}_2$, and $\mathcal{G}_3$) can happen simultaneously at every growth step.

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 volume increment. Due 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 the diversity in the population. Thus to evolve a diverse set of structures, the constraint limit on this experiment is relaxed. Note that Table 1shows the design space boundary 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.

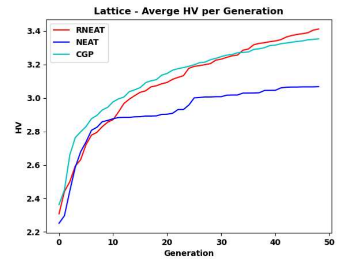
Experiments were conducted on two 2D problems: CB, MBB, and a 3D problem, TL, where



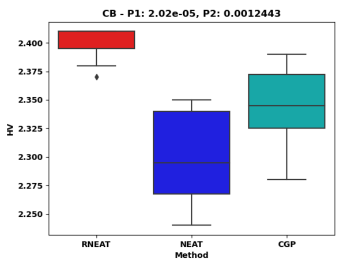
(a) CB



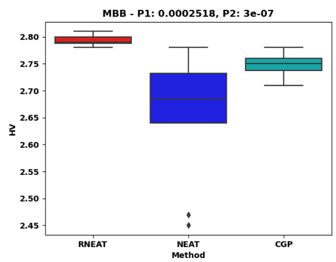
(b) MBB



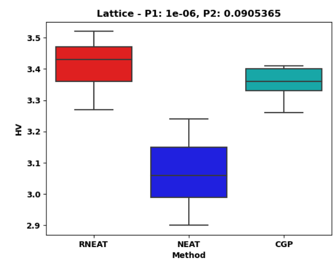
(c) TL



(d) CB



(e) MBB



(f) TL

Figure 11: Results for Size, shape, and topology. Figure compares average HV and distribution of HV over 10 runs when a GRN control both node movement, edge thickness, and cell division.

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

Comparative results are shown in Figure 11 where subfigures (a), (b), and (c) compares the average HV per generation over 10 runs. Figure 11 (d), (e), and (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 on CB, MBB, and TL.

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 combined optimization problem

		RNEAT	NEAT	CGP	RNEAT	NEAT	CGP	RNEAT	NEAT	CGP
			CB			MBB			TL	
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

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

	Problems and Types of Growth										
Methods	$\mathcal{G}_1$				$\mathcal{G}_1$, $\mathcal{G}_2$				$\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$		
	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

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 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.

To summarize these findings, Table 5 provides rankings for the different algorithms on all problems. This ranking is 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 demonstrated the generalizability and scalability of the proposed method. Nonetheless, because RNEAT evolves non-linear neural networks as black box models, delving into the behaviors of the evolved network will necessitate additional experimental analysis.

6 Conclusions and Future Work

This paper introduces a framework based on evolutionary developmental biology (Evo-Devo), in which the growth of structures is controlled by gene regulatory networks represented by neural networks and genetic programming. To evolve high-quality Pareto solutions (GRNs), the paper presents a multi-objective neuro-evolutionary algorithm termed RNEAT, which draws inspiration from NEAT and utilizes a CGP-style encoding of genotypes. The performance of RNEAT is compared to NEAT, and CGP on various 2D and 3D 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 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 hypervolume as a performance indicator, RNEAT outperformed the other algorithms on seven problems, while NEAT and CGP only performed better on 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 gene regulatory network. 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 non-linear loading conditions, where the search space is highly constrained.

Acknowledgement

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. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 37–38.

Balamurugan, R., Ramakrishnan, C., & Singh, N. (2008). Performance evaluation of a two stage adaptive genetic algorithm (tsaga) in structural topology optimization. *Applied Soft Computing*, *8*(4), 1607–1624.

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*, 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

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.

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.

Dhondt. (2017). Calculix crunchix user's manual version 2.12. *Munich, Germany, accessed Sept*. http://www.dhondt.de/ccx%5C_2.19.pdf

Goldberg, D. E. (2002). *The design of innovation: Lessons from and for competent genetic algorithms* (Vol. 1). Springer.

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

Hall, B. K. (2012). Evolutionary developmental biology (evo-devo): Past, present, and future. *Evolution: Education and outreach*, *5*(2), 184–193.

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

Kaldis, P. (2016). Quo vadis cell growth and division?

Karlebach, G., & Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature reviews Molecular cell biology*, *9*(10), 770–780.

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. *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation*, 2701–2726.

Navarro-Mateu, D., & Cocho-Bermejo, A. (2019). Evo-devo algorithms: Gene-regulation for digital architecture. *Biomimetics*, *4*(3), 58.

Navarro-Mateu, D., & Cocho-Bermejo, A. (2020). Evo-devo strategies for generative architecture: Colour-based patterns in polygon meshes. *Biomimetics*, *5*(2), 23.

Olson, E. N. (2006). Gene regulatory networks in the evolution and development of the heart. *Science*, *313*(5795), 1922–1927.

O'Neill, M., Vanneschi, L., Gustafson, S., & Banzhaf, W. (2010). Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, *11*, 339–363.

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*, 100888.

Perez, R. E., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers & Structures*, *85*(19-20), 1579–1588.

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.

Richards, D., Dunn, N., & Amos, M. (2012). An evo-devo approach to architectural design. *Proceedings of the 14th annual conference on genetic and evolutionary computation*, 569–576.

Schlitt, T., & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC bioinformatics*, *8*(6), 1–22.

Schrum, J., & Miikkulainen, R. (2008). Constructing complex npc behavior via multi-objective neuroevolution. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *4*(1), 108–113.

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, *10*(2), 99–127.

Streichert, F., Planatscher, H., Spieth, C., Ulmer, H., & Zell, A. (2004). Comparing genetic programming and evolution strategies on inferring gene regulatory networks. *Genetic and Evolutionary Computation–GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*, 471–480.

Swain, M., Hunniford, T., Mandel, J., Palfreyman, N., & Dubitzky, W. (2005). Modeling gene-regulatory networks using evolutionary algorithms and distributed computing. *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, *1*, 512–519.

Turner, A. J., & Miller, J. F. (2014). Cartesian genetic programming: Why no bloat? *Genetic Programming: 17th European Conference, EuroGP 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers 17*, 222–233.

van Willigen, W. H., Haasdijk, E., & Kester, L. J. (2013). Evolving intelligent vehicle control using multi-objective neat. *2013 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, 9–15.

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

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.

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 II, D. C. (2007). Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. *Neural Networks*, *20*(8), 917–927.

Yildiz, A. R. (2013). Comparison of evolutionary-based optimization algorithms for structural design optimization. *Engineering applications of artificial intelligence*, *26*(1), 327–333.

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.