Concurrency and Computation: Practice and Experience

WILEY

RESEARCH ARTICLE OPEN ACCESS

Optimizing the Substrate for Hypercube-Based Neuroevolution of Augmented Topologies to Design Soft Actuators

Hugo Alcaraz-Herrera[1] | Michail-Antisthenis Tsompanas[1,2] | Igor Balaz[3] | Andrew Adamatzky[1]

[1]Unconventional Computing Laboratory, University of the West of England, Bristol, UK | [2]School of Computing & Creative Technologies, University of the West of England, Bristol, UK | [3]Laboratory for Meteorology, Physics and Biophysics, Faculty of Agriculture, University of Novi Sad, Novi Sad, Serbia

Correspondence: Michail-Antisthenis Tsompanas (antisthenis.tsompanas@uwe.ac.uk)

Received: 29 December 2024 | **Revised:** 20 June 2025 | **Accepted:** 29 June 2025

Funding: This work was supported by the European Union's Horizon Europe research and innovation programme (Grant No. 101070328) and by the UK Research and Innovation grant (Grant No. 10044516).

Keywords: client-server model | HyperNEAT | neuroevolution | optimization | soft robotics

ABSTRACT

The characteristics of soft robots make them better candidates for applications such as healthcare, due to their enhanced safety, adaptability, and more natural human-robot interaction compared to traditional counterparts. Different actuating systems have been proposed for soft robotics. On the other hand, since this technology is fairly young, the design process of soft actuators is not yet well formalized. In an attempt to enhance the applicability of this type of actuator, the utilization of a NeuroEvolution algorithm to automatically design them is proposed here. More specifically, Hypercube-based NeuroEvolution of Augmented Topologies (HyperNEAT) is investigated for different substrate architectures. These substrates are Artificial Neural Networks that encode the three-dimensional representation of the soft actuators. The produced three-dimensional sketches are tested within a simulated environment under two different targets (the maximum displacement and the combination of maximum displacement and minimum actuator volume) to identify the suitability of HyperNEAT as an efficient designing methodology. Since the evaluation of candidate solutions under a physics simulator is the most computationally demanding process, the proposed methodology was realized under a client-server setting, with the aim of accelerating the evolutionary optimization of actuator sketches. The evaluation part of the algorithm was outsourced to the server side, which can be a specialized and high-performing computational entity. The resulting soft actuators of this study proved to be of higher competence when compared with actuators derived under previously published evolutionary methodologies.

1 | Introduction

Robotics research has enabled the conceptualization and fabrication of machines supporting mankind in a plethora of fields, from agriculture [1] to surgery [2]. Nonetheless, a subcategory of robotics that employs flexible, deformable materials, under the term of soft robotics, has recently gained a lot of interest as they proved to perform better in specific areas [3–6]. This can be attributed to the fact that the bodies of such robots are compliant and, thus, they move in a way that mimics biological entities. Consequently, they can better imitate the behaviors of living creatures, which are the result of Nature's evolutionary optimization.

There are several noteworthy paradigms of these bio-inspired soft robotics [7]. Mechanical octopus-like tentacles have been derived from the further development of soft robotic suckers [8],

and showcase the use of flexible building materials combined with smart adhesion. Similarly, to highlight the application of advanced materials and methods of manufacturing, a 3D-printed soft robot driven by a combustion reaction has been studied [9]. Moreover, a soft robot was developed that mimics plant-like growth for environmental navigation [10]. Numerous concepts and designs have also drawn inspiration from plant root systems for soft robotics [11, 12].

Soft robotics is rapidly evolving, with biology serving as a key source of inspiration for designing robots capable of handling delicate environments and objects. The soft robotics discipline shows significant potential across various applications, with ongoing research expanding the possibilities of soft robotic capabilities as the associated technologies progress. Recently, this field has moved beyond mere biological inspiration to incorporating biological tissue directly into robotic systems [13, 14].

To address the challenges of using and controlling soft materials (and in some cases, living tissues), automated methods for designing soft actuators for robots have been explored here to reveal their full potential. The approach outlined in this work marks an early step toward creating a soft actuator, that is, suitable for biomedical applications.

Even conventional robot design is inherently complicated, often requiring numerous real-world testing iterations to mitigate the reality gap injected in the production methodology from the use of imperfect simulations. Consequently, earlier studies demanded significant time and material resources to identify optimal designs [15]. Now considering soft robots, these challenges are even greater [16], largely due to the use of pliable materials that are characterized by nonlinear mechanical properties.

An intriguing method called NeuroEvolution (NE) was utilized in this work to address the challenges of soft robot design. The proposed methodology uses genetic algorithms (GAs) to optimize both the topologies and connection weights of artificial neural networks (ANNs). Among NE techniques, the most notable one is NeuroEvolution of Augmenting Topologies (NEAT) [17], which has been effectively applied in various robotic domains, including robotic morphology design [18] and gait generation [19]. A similar methodology was developed to mimic natural patterns during the learning process. This method is an extension of NEAT and is known as HyperNEAT; it uses Compositional Pattern-Producing Networks (CPPNs) to take advantage of the symmetry or any patterns that may exist in the evolved topologies [20]. Based on previous work [21], the morphologies produced by the HyperNEAT methodology have been shown to be of equivalent fitness as the ones produced by CPPN-NEAT. Moreover, the inefficiencies of Genetic Algorithms with direct representations when compared to CPPN-NEAT (and, thus, HyperNEAT) have been investigated previously [22].

Thus, the main objective of our research is to assess the suitability of HyperNEAT as a design engine of soft actuator morphologies (SAMs). We evaluate the methodology's performance using various predefined substrates that HyperNEAT evolves, and they are decoded to generate three-dimensional sketches of SAMs. The comparison between substrates is based on two key metrics: The maximum displacement achieved within a given time frame and the trade-off between the total volume of the morphologies and the displacement attained. The rationale behind the trade-off fitness function (penalizing higher volumes of the actuator) is two-fold. Specifically, higher volumes of the actuator will require more material, thus having a negative impact on costs and sustainability. Moreover, higher volumes of actuators imply a higher amount of active material that will require increased energy for achieving the target displacement.

Moreover, to accelerate the computations, since a computationally demanding physics engine is used to provide the fitness value of each candidate, the implementation of the HyperNEAT method is designed with the provision of multiprocessing attributes. Specifically, the implemented technique employs a client-server architecture, wherein the computationally demanding components of the optimization process are offloaded to distributed high-performance computing platforms. This approach leverages parallelism and resource scalability to enhance computational efficiency and reduce processing time.

2 | Background

Evolutionary optimization has been widely used in the field of robotics during the past three decades. A highly influential work [23] that shaped the future of this field has studied the evolution of both the morphology and the controller of robots. The robotic sketches produced through a genetic algorithm were simulated and proved to have natural locomotion attributes for swimming, flying, and walking behaviors. Despite the fact that the first ever work on this field took into account the evolution of both morphology and controller, the majority of the early subsequent works would mainly investigate fixed morphologies with a focus on evolving the controllers of these machines [24]. On the other hand, there have recently been numerous works in this area that can be categorized into three groups: (i) the ones focused on evolved morphologies, (ii) the ones focused on evolved controllers, and (iii) the ones studying the hybrid evolution of both aspects of a robot.

Moreover, earlier works mostly dealt with rigid structures being used to evolve a morphology of a robot. Namely, for some of these works [25, 26], these rigid structures were crane arms and bridges, made out of unbending plastic bricks, and for some works [27] these structures would additionally include wheels to enable locomotion. However, recent advances in material science made soft robotics popular; thus, the use of evolutionary methods on this subcategory of soft robotics was highly motivating [22].

Nonetheless, the utilization of indirect representations for the evolutionary optimization and appropriate algorithmic methodologies offered better outputs. For instance, an implementation of CPPN-NEAT on the designing process of morphologies for rigid robots manages to replicate the interplay between physical structures and their controllers. This foundational work laid the groundwork for subsequent studies where both morphology and control systems were co-evolved [18]. As mentioned previously, the application of these methodologies in soft robots proved to be highly beneficial. This has been proved in an approach [28] of using CPPN-NEAT to evolve amorphous robots capable of

locomotion through the rhythmic expansion and contraction of their constituent materials.

Lately, HyperNEAT has been employed for the co-design of robotic controllers and morphologies. For example, a novel methodology was proposed to simultaneously optimize robot shapes and control strategies [29]. This approach was evaluated across four experimental scenarios, focusing on the adaptive capabilities demonstrated by the robots. The findings indicated that the method effectively generates viable morphologies and corresponding controllers capable of executing the specified tasks with proficiency.

HyperNEAT has also been used to design controllers for driving purposes. A study presents a simulation of autonomous robots whose controller was designed by HyperNEAT [30]. The robots employed a 180° wide sensor array, which simulated a camera as input for the controller that could be used in a real robot. The simulations of robots were performed on ViVAE (Visual Vector Agent Environment), and they were trained to drive, maximizing the average speed and trying to avoid collisions with obstacles and with other robots. Results indicated that HyperNEAT was able to design controllers capable of driving at an acceptable speed and avoiding collisions in a limited number of evolutionary steps.

Whereas HyperNEAT primarily focuses on evolving controllers that leverage the geometric properties of morphologies (as in the aforementioned studies), it can also be utilized for the evolution of morphologies based on voxel representations (arrays of three-dimensional coordinates). Typically, a bounded region consisting entirely of potential voxels is defined, and passing a voxel's coordinates to the substrate yields an output that determines the voxel's inclusion within the hypercube space and its characteristics. This was previously performed by CPPN-NEAT [22, 28] but not by HyperNEAT. The method we propose here and an investigation of the best substrate architecture for the evolutionary optimization of morphologies relies on HyperNEAT and is carried out for a specifically defined target problem of an evolved soft robot actuator.

3 | HyperNEAT

HyperNEAT is an extension of NEAT, an algorithm designed to evolve the topology of ANNs, typically generating arbitrary topologies [17]. HyperNEAT employs NEAT to evolve a specific type of neural network (CPPNs) [31]. HyperNEAT employs CPPNs due to their ability to compose patterns such as repetition and symmetry [20]. One of the primary differences between NEAT and HyperNEAT is the use of diverse activation functions. NEAT only generates ANNs that include hidden nodes with sigmoid functions. In contrast, HyperNEAT can utilize other activation functions (e.g., periodic and Gaussian functions) in each node. HyperNEAT was designed to evolve ANNs by exploring a wider range of network architectures.

HyperNEAT can embody the geometry of the domain problem due to the CPPNs' properties since they help to compute the topology of ANNs, considering their geometry. The geometric space or layout where HyperNEAT operates is called *substrate*, which can be configured in numerous ways. One of the most popular configurations is *grid*, composed of a set of nodes (i.e., neurons) allocated in a bi-dimensional plane. Another well-known configuration is called *three-dimensional grid* [20], where a set of neurons is allocated in a three-dimensional space. It is noteworthy that "*substrate*" and "ANN representation of the SAM" are equivalent terms and henceforth only the term "*substrate*" will be used.

Figure 1a exhibits an example of a two-dimensional grid, whereas an example of a three-dimensional grid is shown in Figure 1b. Thus, the objective of HyperNEAT consists of evolving the connectivity (i.e., topology) and weights of substrates by employing CPPNs.

CPPNs generate the topology of the substrate considering the position of neurons. Therefore, if a two-dimensional substrate is implemented, a four-dimensional function to compute the weight between the neuron a and the neuron b is defined as follows:

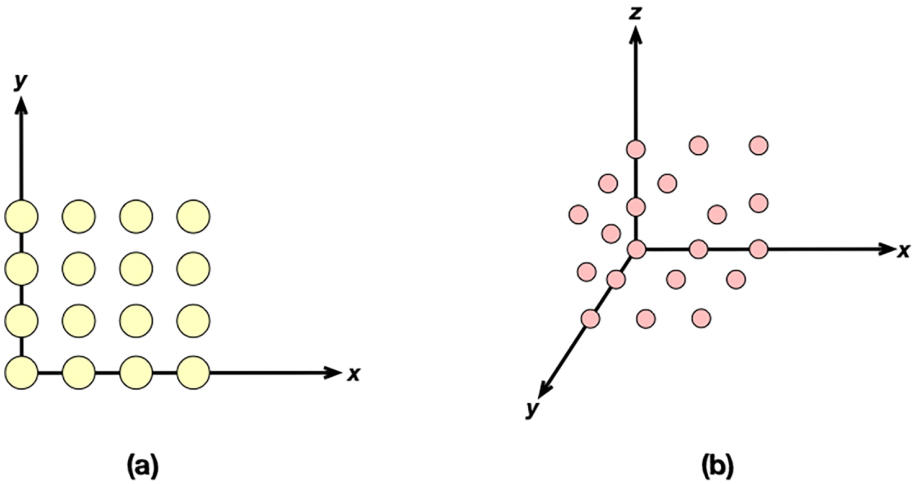

FIGURE 1 | Examples of substrates commonly used in HyperNEAT: (a) Two-dimensional, (b) three-dimensional.

ALGORITHM 1 | General HyperNEAT algorithm.

```
1: procedure EXECUTE HYPERNEAT
2:     define substrate
3:     generate CPPNs population
4:     while stop condition not met do
5:         for each CPPN do
6:             generate connections for substrate
7:             generate bias for substrate
8:             assign CPPN aptitude
9:         end for
10:        evolve CPPNs through NEAT
11:    end while
12: end procedure
```

$$CPPN(x_a, y_a, x_b, y_b) = weight_{ab} \tag{1}$$

Following the formal definition of HyperNEAT [32], the bias of neurons is calculated utilizing the coordinates of the neuron whose bias is required in the "origin" position. Furthermore, the coordinates of the "destination" position are set to zero. Hence, the bias of the neuron a is calculated as follows:

$$CPPN(x_a, y_a, 0, 0) = bias_a \tag{2}$$

Regarding a three-dimensional substrate, to obtain the weight between neuron a and neuron b, a six-dimensional function is used:

$$CPPN(x_a, y_a, z_a, x_b, y_b, z_b) = weight_{ab} \tag{3}$$

The bias of the neuron a is calculated as follows:

$$CPPN(x_a, y_a, z_a, 0, 0, 0) = bias_a \tag{4}$$

Algorithm 1 describes the general workflow of HyperNEAT. The generation of the topology and, consequently, the weights and bias of the substrate (lines 6 and 7) are obtained by querying each CPPN employing Equations (1) and (2) if a two-dimensional substrate is being utilized. Or Equations (3) and (4) if a three-dimensional substrate is being used.

It is important to highlight that the outputs of CPPNs are usually in the $[-1.0, 1.0]$ range. Therefore, it is necessary to normalize or process CPPN outputs as required.

4 | Methodology

The primary purpose of this research is to find a suitable substrate configuration that enables HyperNEAT to design morphologies of soft actuators capable of performing locomotion tasks such as reaching a significant displacement in a specific time frame. To determine the most suitable substrate, four aspects are considered: (i) simulation of SAMs, (ii) HyperNEAT configuration, (iii) software deployment, and (iv) experimental setup. The following sections describe these aspects in detail.

4.1 | Simulation of Soft Actuator Morphologies

Building physical soft actuators to test them is not feasible since it implies a significant effort in terms of resources and time. To overcome this issue, SAMs are simulated in a three-dimensional physics engine called *Voxelyze*, due to its capacity to simulate numerous mechanical dynamics such as friction and gravity [16]. A *voxel* is the primary building block in Voxelyze that can simulate different materials (e.g., passive and active).

The output of Voxelyze contains: (a) the absolute displacement of the soft actuator from its initial position during a specific time frame, and (b) the number of voxels composing the SAM. The output is used to evaluate the performance of soft actuators and, consequently, the performance of the substrate utilized during the design process. Details of the implementation of Voxelyze are provided in previous research [33, 34]. Voxelyze is deemed as an appropriate simulator to provide the fitness function of SAMs despite possible reality gap issues, since it has been previously validated on a range of prototype soft robots made from a range of material (i.e., only artificial material [35] up to biohybrid machines [33]).

4.2 | HyperNEAT Configuration

In the scope of this research, HyperNEAT evolves CPPNs that encode substrates [20]. Substrates decode into three-dimensional SAMs. Thus, based on the three-dimensional layout (x, y, z) required for Voxelyze, where SAMs are designed, the number of input neurons of substrates is three. Moreover, the number of output neurons is two since it is necessary to determine: (i) the presence of a voxel for each (x, y, z) position across the layout, and (ii) the type of material the voxel is made. Namely, active or passive.

Considering the aforementioned definition, a substrate is queried as follows:

$$SUBSTRATE(x_i, y_i, z_i) = PV_i, M_i \tag{5}$$

where $[x_i, y_i, z_i]$ represent the coordinates of the i^{th} point in the three-dimensional layout. PV_i represents the presence of a voxel in the i^{th} point of the three-dimensional layout, and M_i refers to the type of material the voxel is made of in the same point of the three-dimensional layout.

Equation (6) defines the procedure to map PV_i as presence or absence of a voxel:

$$Presence(x_i, y_i, z_i) \begin{cases} \text{yes}, & |PV_i| \geq 0.5 \\ \text{no}, & \text{otherwise} \end{cases} \tag{6}$$

Regarding M_i, it defines the type of material the voxel is made of. Following previous studies [33, 34], this research considers only two types of materials: (a) *active* (i.e., contractile), which is numerically encoded as 3, and (b) *passive*, encoded as 1. Thus, Voxelyze uses these codes to interpret how morphologies are built. In addition, Equation (7) describes the mechanism of how M_i is mapped as a material id.

$$Material\ code(x_i, y_i, z_i)\begin{cases}1, & |M_i| < 0.5\\ 3, & \text{otherwise}\end{cases} \quad (7)$$

Finally, following the layout configuration used in Reference [21], the x and y axis range is $[0, 8]$, while the z axis has a range of $[0, 7]$.

4.3 | Software Deployment

Evaluating individuals requires an intense interaction with Voxelyze, which implies a considerable computation time. To counteract this issue, the implementation of HyperNEAT utilized in this research was designed under the multiprocessing paradigm. The software has been developed using a *client-server* architecture to take advantage of the distributed computing features.

The general workflow of the HyperNEAT implementation used in this research is presented in Figure 2. The evaluation of individuals can occur concurrently, which provokes an asynchronous execution. Figure 3 shows in detail the evaluation stage. First, the client side sends the SAMs to the server, which simulates them using numerous Voxelyze instances and returns the displacement observed and the number of voxels of the SAMs. This data is employed to calculate the fitness of SAMs and, hence, of the substrate.

Due to its serialization features, JSON is the data format selected for transferring data between HyperNEAT (running on the client side) and the fitness function (running on the server side). Communication between both ends is established by HTTP, using the GET and POST methods. It is important to highlight that the server has implemented a reverse proxy acting as a load balancer.

One of the software's main features is its ease of implementation; it can be deployed across numerous infrastructures. For instance, the client is independent of the operating system and

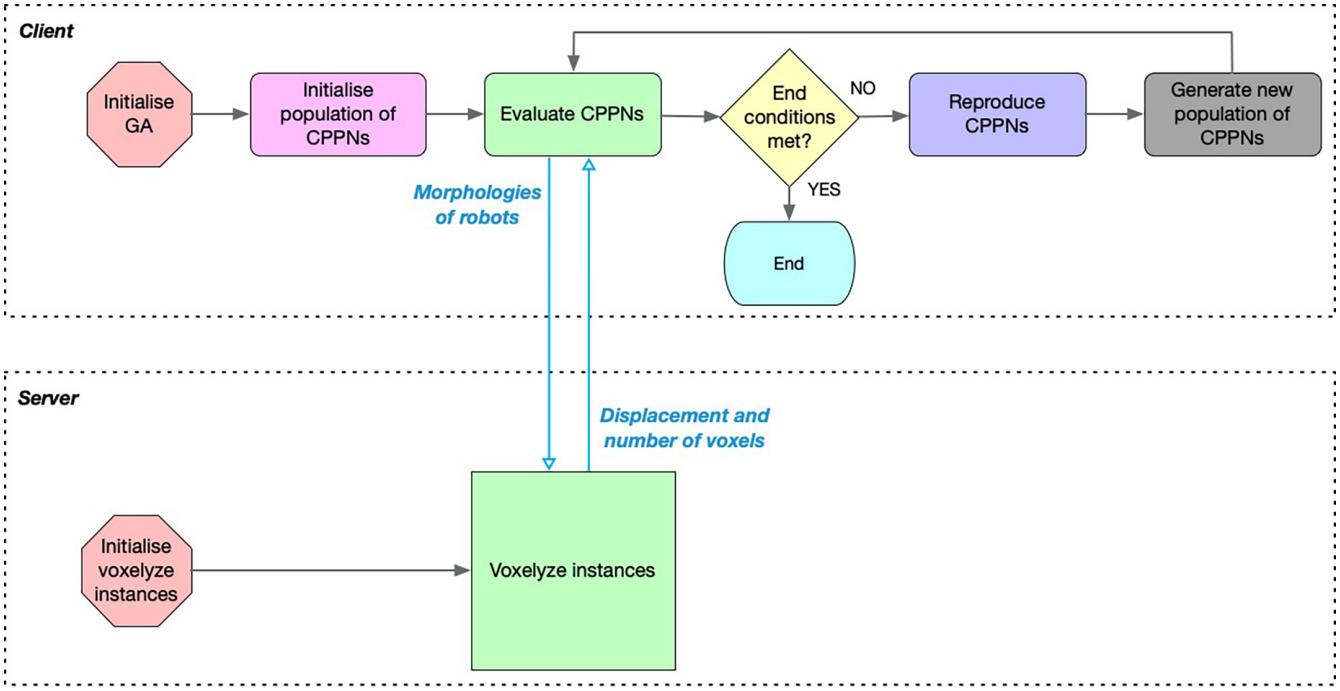

FIGURE 2 | Diagram of the implementation of HyperNEAT for producing SAMs. Figure adopted from Reference [21].

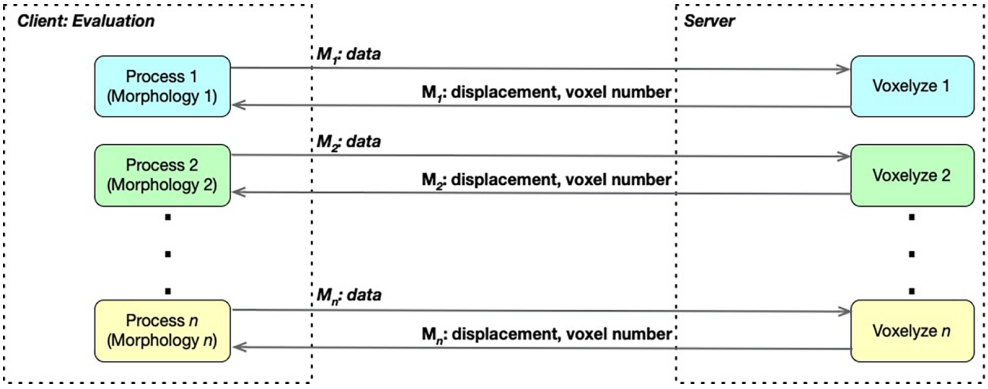

FIGURE 3 | Evaluation stage: Voxelyze instances concurrently simulate SAMs, providing as output the displacement observed and the number of voxels composing SAMs. These values are used to assign a fitness value through the procedures described in Sections 5.2 and 5.3. Figure adopted from Reference [21].

processor architecture. On the other hand, the server may be deployed in any cloud computing platform or a virtual machine. Furthermore, the software takes advantage of the computational power (i.e., the number of cores in the CPU) at both ends. For this research, a virtual machine was utilized during experimentation. Finally, the hardware configured for the virtual machines is described as follows:

- *Processor*: ARM, 9 cores (18 threads), 3.20GHz.
- *RAM memory*: 16GB, LPDDR5.

4.4 | Experimental Set-up

The population contains 50 encoded individuals (i.e., CPPNs producing substrates that will be decoded into SAMs). Each evolutionary run lasts 1000 generations. In addition, the activation function dictionary contains the following functions: *sine*, *negative sine*, *absolute*, *negative absolute*, *square*, *negative square*, *square root of absolute*, *negative square root of absolute*, and *sigmoid*. The population is initialized following the procedure described in Reference [33] to provide a meaningful comparison of the outputs. Table 1 shows the parameters used during the evolution of CPPNs.

To fine tune the substrate architecture, the allocation of the neurons in the instances used in this research is commanded by a combinatorial procedure with the following rules: (i) the number of hidden layers is in the $\{1, 2, 3\}$ set, and (ii) the number of neurons per layer is in the $\{1, 3, 5, 7\}$ set. Thus, each substrate is defined by one of all possible combinations of the previous rules. For instance, the minimal substrate has only one hidden layer containing one neuron, whereas the most complex substrate has three hidden layers containing seven neurons each. Thus, 84 different substrate architectures were generated.

Due to the significant number of substrates involved in this research, we clustered them by the number of neurons in the first hidden layer. Thus, there are four clusters: (a) First layer with one neuron (L-1); (b) First layer with three neurons (L-3); (c) First layer with five neurons (L-5); and (d) First layer with seven neurons (L-7). To refer to a specific substrate architecture the following notation is used: L-{number of neurons in the first layer}{number of neurons in the second layer}{number of neurons in the third layer}; thus, a substrate with one neuron in the first layer, three neurons in the second layer and five neurons in the third layer will be denoted as L-135. Moreover, the activation function implemented for substrates is *ReLU* due to its sparsity properties and the fact that it can induce a linear behavior [36].

5 | Results

Since the primary purpose of this research is to identify a suitable substrate for HyperNEAT, whereas these substrates can encode SAMs, it is feasible to study the performance of the produced SAMs. Thus, three metrics are utilized: (i) identifying the fittest substrates through the general performance of the fittest SAM during evolution, (ii) studying the displacement of SAMs during ten simulated seconds regardless of the external conditions, and (iii) analyzing the trade-off between displacement and the number of voxels composing SAMs.

5.1 | Evolutionary Optimization Based on the Maximum Displacement

In this metric, the primary target for SAMs is their potential ability to move the farthest. The first step involves comparing the general performance of the top substrates through the evolutionary behavior of the fittest individual. The fitness of each SAM candidate is obtained as the maximum displacement.

It is important to emphasize that: (i) the experiment does not consider the number of voxels of SAMs as a part of the performance analysis, and (ii) for each evolutionary trial, 50 controllers are randomly generated and used throughout the trial. For each

TABLE 1 | Parameters utilized to evolve CPPNs under HyperNEAT to generate SAMs.

Parameter	Value	Description
Compatibility threshold	3	If the *genomic distance* of individuals is less than this value, they are in the same species [17].
Compatibility disjoint coefficient	1.0	Contribution coefficient for the excess and disjoint gene counts when the genomic distance is calculated.
Compatibility weight coefficient	0.5	Contribution coefficient for each weight and bias when the genomic distance is calculated.
Maximum stagnation	15	If a species does not exhibit improvement in at least this number of generations, it is removed.
Survival threshold	0.3	The proportion of each species that is allowed to reproduce during evolution.
Activation function mutate rate	0.4	The probability to replace the activation function of neurons.
Adding/deleting connection rate	0.3/0.2	The probabilities to add/delete a connection between existing neurons.
Activating/deactivating connection rate	0.5	The probability to activate/deactivate an existing connection between neurons.
Adding/deleting node rate	0.3/0.2	The probabilities to add/delete a neuron.

generation, the aforementioned 50 controllers are shuffled and paired with SAMs during the simulation stage. Figure 4 presents the mean performance of the fittest individual (i.e., the fittest SAM) in terms of reaching the maximum displacement found by the top three substrates per cluster (see Section 4.4) across 15 evolutionary trials. Each curve (i.e., mean performance) exhibits 95% confidence intervals shown by the shaded regions.

Regarding the performance of the substrates of L-1 (Figure 4a), although their evolutionary pace seems similar, some statistically significant differences exist. First, all the data collected were tested and found not to be normally distributed (Shapiro-Wilk test; $p < 0.05$). Then, using Dunn's test, it is feasible to confirm that no significant differences exist between the performances of L-1 and L-113 ($p > 0.05$). In contrast, there are significant differences between the performance of L-177 and the rest ($p < 0.05$). Therefore, a rank performance can be conducted: L-1 and L-113 > L-177.

Furthermore, when the performance of the substrates L-3 is studied (Figure 4b), some statistically significant differences can be visually appreciated in the evolutionary pace of the substrates. Moreover, all the data gathered are not normally distributed (Shapiro-Wilk test; $p < 0.05$), and through Dunn's test, it is possible to confirm that significant differences exist between the performance of L35 and the performance of the other substrates ($p < 0.05$). On the other hand, there are no significant differences between the performances of L-333 and L-355 ($p > 0.05$). Based on the previous result, the performance of the substrates can be ranked as follows: L-35 > L-333 and L-355.

When the performances of the substrates of L-5 are analyzed (Figure 4c), they present some statistically significant differences in their evolutionary pace. All the data collected were tested, and their distribution is not normal (Shapiro-Wilk test; $p < 0.05$). By utilizing Dunn's test, it can be confirmed that no significant differences exist between the performances of L-555 and L-511

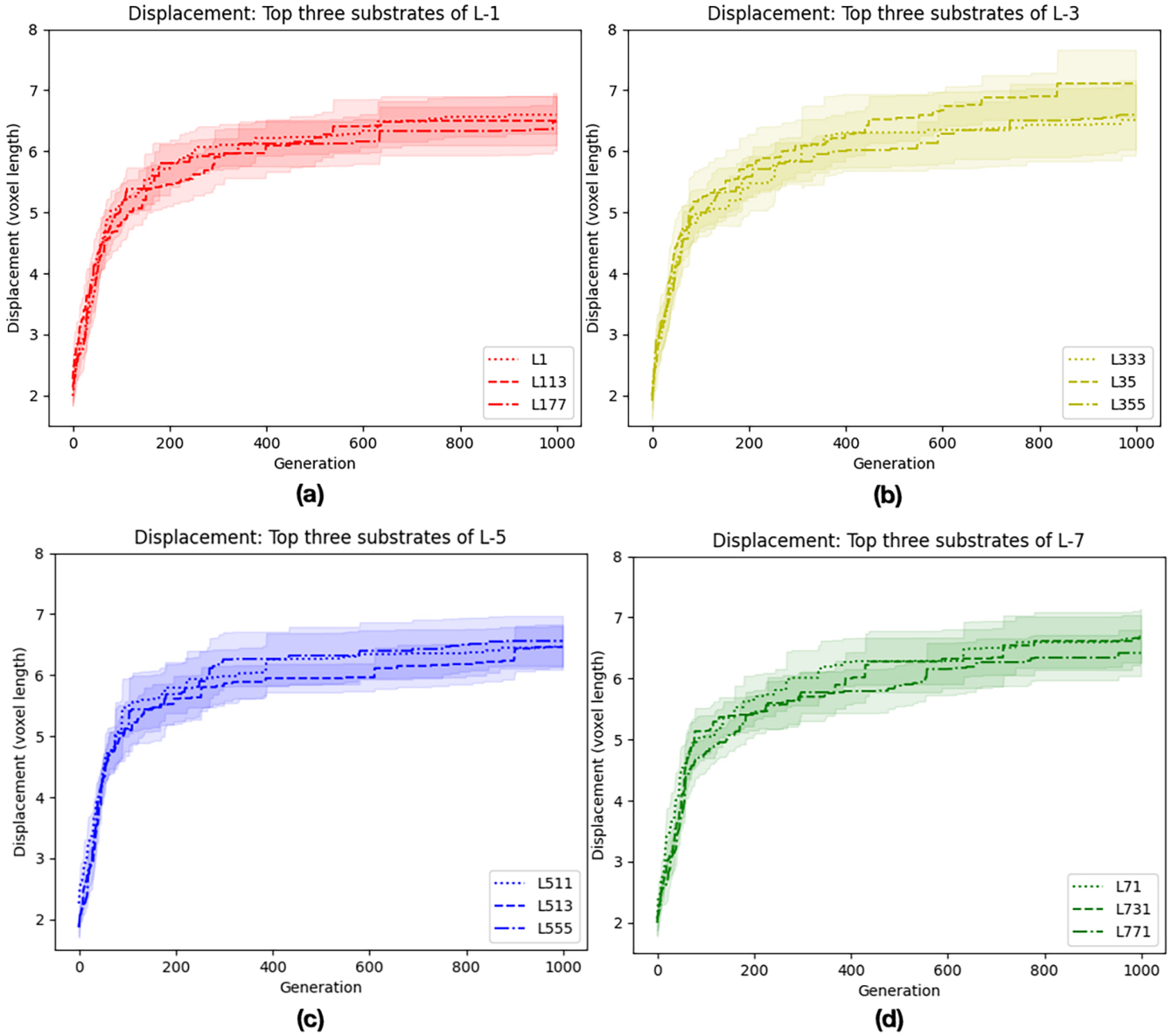

FIGURE 4 | Mean performance in terms of maximizing displacement with 95% of confidence intervals depicted by the shaded regions of the fittest SAM found by the top three substrates associated to: (a) Cluster L-1, (b) cluster L-3, (c) cluster L-5, and (d) cluster L-7.

($p > 0.05$). However, significant differences exist between the performance of L-513 and the rest of the substrates ($p < 0.05$). Thus, ranking the performance is suitable: L-555 and L-511 > L-513.

In addition, examining the performances of the substrates of L-7 (see Figure 4d), some statistically significant differences can be observed. All the data gathered exhibit a non-normal distribution (Shapiro-Wilk test; $p < 0.05$). Employing Dunn's test makes it possible to confirm that no significant differences exist between the performances of L-731 and L-71 ($p > 0.05$). In contrast, there are significant differences between the performance of L-771 and the performance of the rest of the substrates ($p < 0.05$). Considering the previous outcome, it is suitable to rank the performance of the substrates: L-731 and L-71 > L-771.

Finally, for research completeness, Figure 5 shows the mean performance of the fittest SAM in terms of reaching the maximum displacement found by the top substrate per cluster. The shaded regions surrounding the curves represent the 95% confidence intervals. The performances present some statistically significant differences. Through Dunn's test, it is possible to confirm that there are no significant differences among the performances of L-1, L-555, and L-731 ($p > 0.05$). Significant differences exist, nevertheless, between the performance of L-35 and the performance of the rest of the substrates ($p < 0.05$). Therefore, the performance of the substrates can be ranked: L-35 > L-731, L-1, and L-555. The previous result can be confirmed visually: L-1, L-555, and L-731 exhibit a steady evolutionary behavior around generation 780. However, the evolutionary process of these three substrates stagnates, showing either minimal or no improvement. On the other hand, L-35 evolves at the same evolutionary pace as the other substrates until generation 410. From this point, it presents a faster evolutionary process, which stagnates around generation 830.

5.2 | Evaluation of Best SAMs Against Random Controllers

The results obtained from previous experiments (see Section 5.1) demonstrated that the L-35 substrate is the most suitable for finding SAMs capable of reaching the maximum displacement. However, the exhibited performance was achieved by utilizing a certain number of phase offset scenarios during evolution (i.e., 50 randomly selected scenarios per run). Thus, a new metric is investigated here that involves determining which substrate yields the best performance in terms of producing SAMs capable of achieving the maximum displacement possible, despite the external conditions, that is, with a vague control strategy. Figure 6 presents violin plots comparing the fittest SAM (i.e., the fittest of 15 evolutionary runs as mentioned in the previous section) found by each substrate per cluster in terms of reaching the maximum displacement. Each violin plot displays the minimum, maximum, median, and kernel density estimation of the frequency distribution of the displacement values observed across 500 randomly generated phase offset scenarios.

When the displacements of the SAMs of cluster L-1 (see Figure 6a) are analyzed, they present some statistically significant differences. The data collected are not normally distributed (Shapiro-Wilk test; $p < 0.05$). Then, using Dunn's test, it is feasible to confirm that no significant differences exist between L1 and L177 ($p > 0.05$). However, these substrates significantly outperform L-113 ($p < 0.05$). Consequently, it is possible to rank the substrates as follows: L1 and L177 > L113. It is essential to note that the criterion for selecting the fittest SAM, and consequently, the fittest substrate within the cluster, is based on the complexity of the topology, where substrates with fewer neurons are considered more suitable. Thus, L1 has been chosen as the fittest substrate of the cluster since it has one hidden neuron. Moreover, studying the displacements produced by the SAMs of

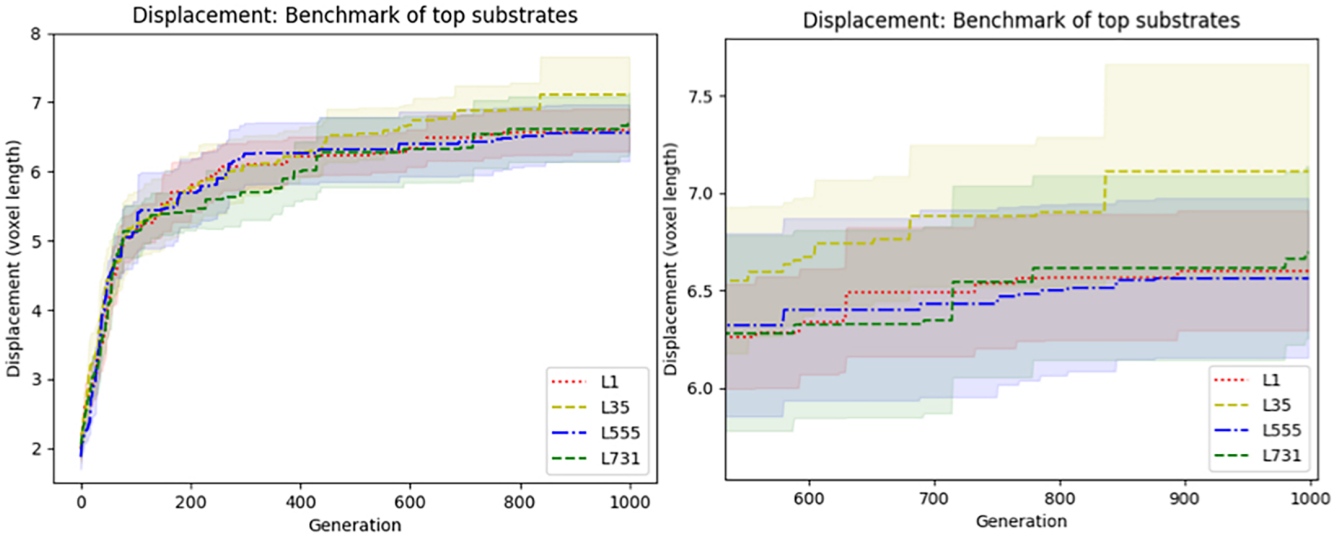

FIGURE 5 | Left: Mean performance in terms of maximizing displacement with 95% of confidence intervals shown by the shaded regions of the fittest SAM found by the top substrate per cluster; Right: Close up of the last evolutionary stage of the fittest SAM found by the top substrate per cluster.

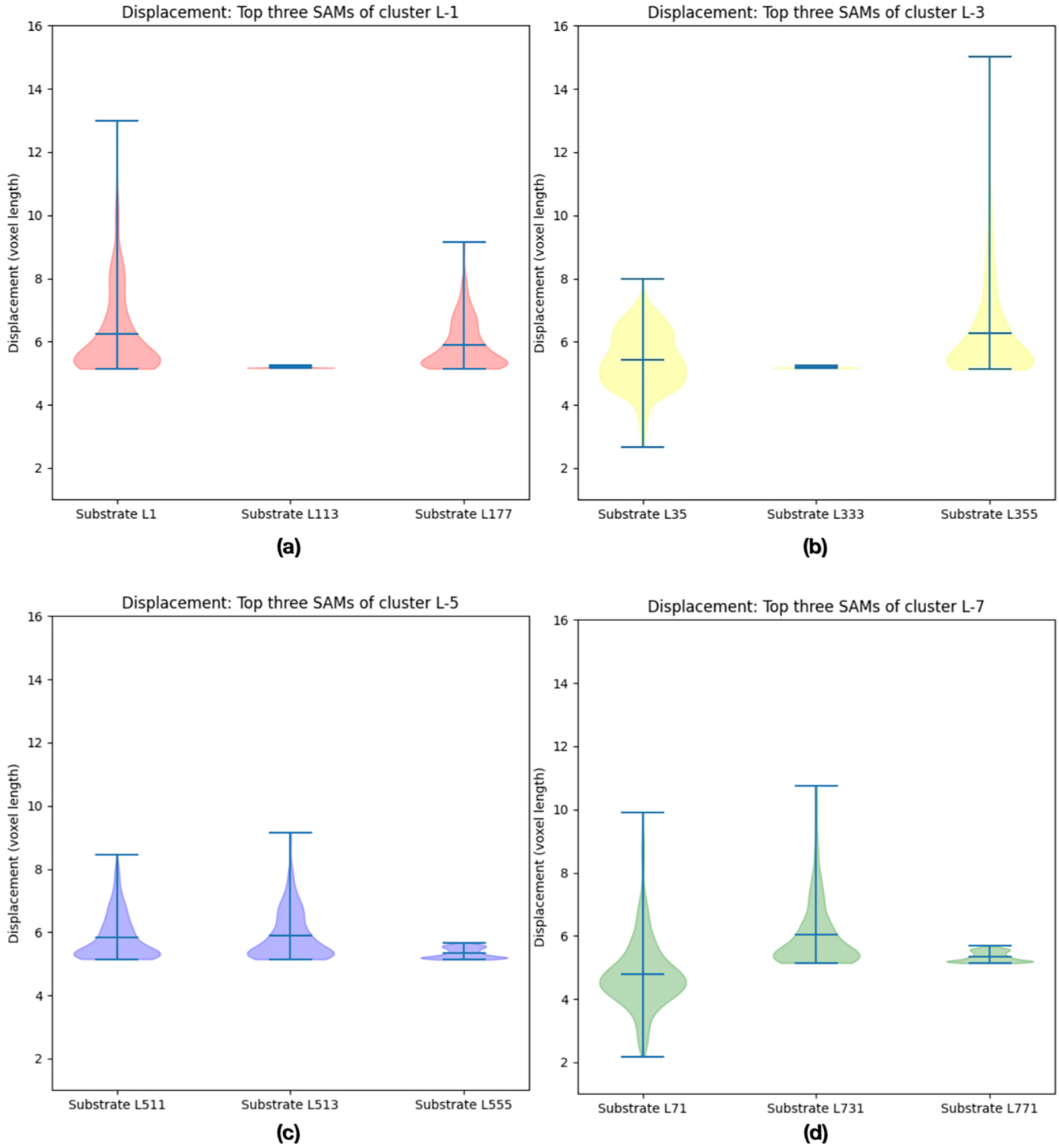

FIGURE 6 | Top three SAMs in terms of reaching the maximum displacement of: (a) Cluster L-1, (b) cluster L-3, (c) cluster L-5, and (d) cluster L-7.

cluster L-3 (see Figure 6b), statistically significant differences are observed. All the data gathered show a non-normal distribution (Shapiro-Wilk test; $p < 0.05$). Through Dunn's test, it is possible to confirm significant differences among the substrates, and hence, they can be ranked: L355 > L35 > L333 ($p < 0.05$).

Regarding the performance of the SAMs in cluster L-5 (see Figure 6c), some statistically significant differences were observed. The Shapiro-Wilk test confirmed that the data are not normally distributed ($p < 0.05$). Subsequently, using Dunn's test, it is feasible to confirm that there are no significant differences between the displacements of L511 and L513 ($p > 0.05$). On the other hand, significant differences exist between these substrates and L555 ($p < 0.05$). Considering this result, a ranking can be performed: L511 and L513 > L555. Following the topology-complexity criterion, the L511 substrate is selected as the most suitable substrate for the cluster. Similarly, for the SAMs in cluster L-7 (see Figure 6d), the data were found to be

non-normally distributed (Shapiro-Wilk test; $p < 0.05$). Then, Dunn's test further revealed the performance ranking for this cluster as: L-731 > L-771 > L-71 ($p < 0.05$).

Once the fittest SAM, and hence, the fittest substrate per cluster, has been identified, it is suitable to perform a broader comparison. Thus, Figure 7 exhibits violin plots comparing the fittest substrate per cluster in terms of reaching the maximum displacement with the wider range of control scenarios. Each violin plot displays the maximum, minimum, median, and kernel density estimation of the frequency distribution of the displacement values observed across 500 randomly generated phase offset scenarios. In general, no significant differences exist among the substrates. This is corroborated by employing Dunn's test ($p > 0.05$). This result can be interpreted as all substrates generate SAMs capable of maximizing the displacement regardless of the external conditions or with a wide range of control strategies. However, if the topology-complexity criterion is considered for the morphology designing tool, a rank can be conducted: L1 > L355 > L511 > L731.

Finally, to benchmark the fittest SAM found in this experiment, a comparison with the fittest SAM found in a previously published experimental methodology [21] with NEAT methodology is considered. Furthermore, it is noteworthy that the controller scenarios, the parameters used to evolve CPPNs, and the activation function dictionary from previous research are employed in this experiment to ensure a fair comparison of the substrates producing morphologies. Figure 8 depicts violin plots comparing the performance of the fittest SAM found in previous research and the fittest SAM found in this research across 500 phase offset scenarios generated a priori at random. It is essential to note that, to avoid biased experimentation, this set of phase offset scenarios is entirely different from the one used in previous experiments (see Figure 6). Thus, using the Wilcoxon test, it is feasible to confirm statistically significant differences between the two approaches ($p < 0.05$). To assess the robustness of the results with respect to variations in the randomized controller scenarios, four different additional sets of 500 controller scenarios each and with perturbations of −20%, −10%, +10%, and +20% of the baseline phase offsets were tested. No statistically significant difference was observed in any of these tests (data not shown here).

Results suggest that the fittest SAM found in this research can outperform the SAM discovered in Reference [21] to reach the maximum displacement possible, regardless of the controller scenario. Furthermore, the significantly high density of data observed at the bottom of the violin plot suggests that the performance of the SAM found in this research is more consistent despite variations in the controller scenario. Thus, HyperNEAT with a substrate whose configuration is layer one with one neuron outperforms NEAT in designing SAMs capable of maximizing the displacement. It is noteworthy here that since HyperNEAT uses an additional representation in encoding the 3D space (i.e., the substrate) compared to NEAT, there is some computational overhead associated with its computations. However, the main computational demand is dominated by the physics simulator used for the fitness calculation; thus, the aforementioned overheads are considered minuscule.

5.3 | Finding the Balance Between Volume and Displacement

The ability of SAMs to move is essential. However, other aspects must be considered since they represent devices planned to be manufactured on a significantly small scale. Therefore, a SAM with a smaller volume (i.e., fewer active voxels) is considered more suitable because it consumes less energy, and less material is necessary to build it. Thus, in this experiment, the evaluation considers two aspects: (a) the ability to maximize the displacement, and (b) minimizing the number of voxels composing

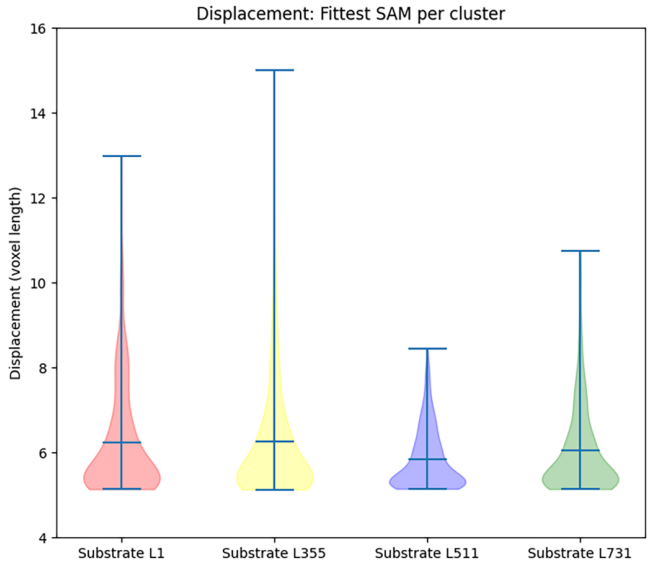

FIGURE 7 | Fittest SAMs in terms of reaching the maximum displacement per cluster.

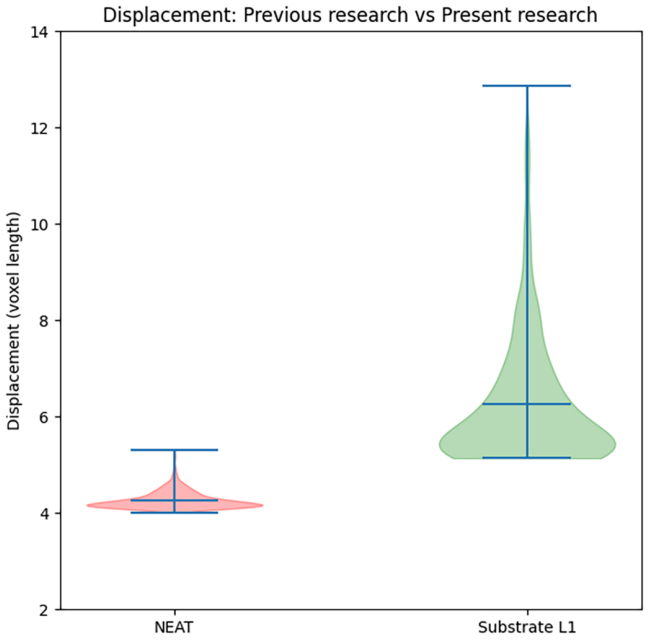

FIGURE 8 | Fittest SAM in terms of reaching maximum displacement found in: Previous research (left), and present research (right).

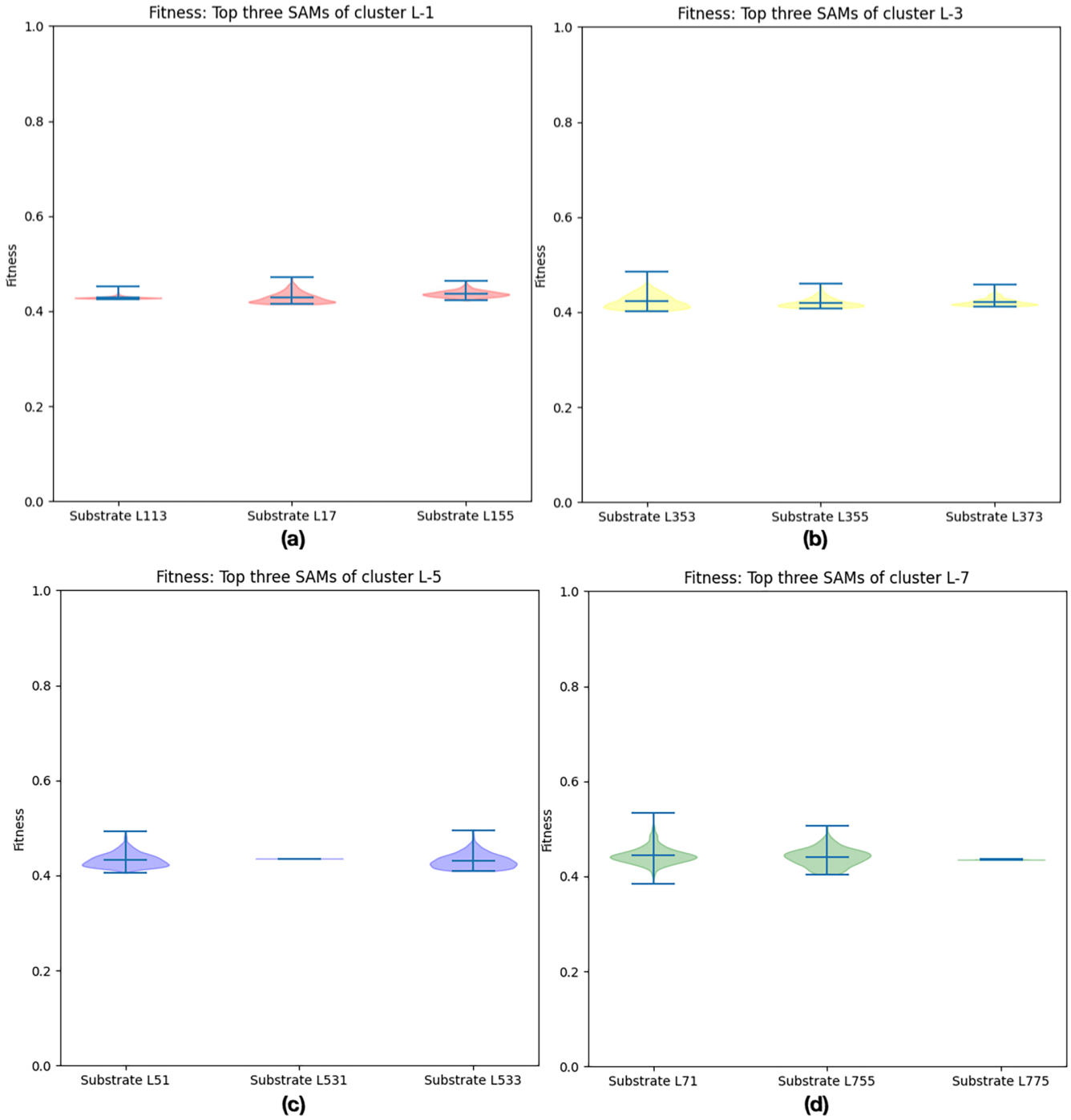

FIGURE 9 | Fitness observed under 500 different controller scenarios of the top three SAMs associated to: (a) Cluster L-1, (b) cluster L-3, (c) cluster L-5, and (d) cluster L-7.

SAMs. Thus, the displacement (δ) of SAMs is evaluated through the following equation:

$$\delta_m = \frac{\Delta_m}{\Delta_{\max}} \tag{8}$$

where Δ_m is the displacement of the SAM m, which is part of the output of Voxelyze, and $\Delta_{\max}$ is the maximum displacement configured for this experiment, which is set to 20 voxel lengths. Thus, δ is normalized in the $[0.0, 1.0]$ range. On the other hand, the evaluation of the number of voxels (ν) of a SAM is obtained by applying the following equation:

$$\nu_m = 1 - \frac{\Upsilon_m}{\Upsilon_{\max}} \tag{9}$$

where Υ_m represents the number of voxels contained in the SAM m and $\Upsilon_{\max}$ is the maximum number of voxels possible, which is calculated by the dimension of each axis of the layout where SAMs are designed: $8 \times 8 \times 7 = 448$. Hence, $\Upsilon_{\max} = 448$.

In addition, ν is normalized in the $[0.0, 1.0]$ range. Once δ and ν have been computed, the fitness of the SAM m is obtained as follows:

$$fitness_m = \frac{1}{2}\delta_m + \frac{1}{2}\nu_m \tag{10}$$

Figure 9 shows violin plots comparing the fitness observed of the top three SAMs per cluster. Each violin plot exhibits minimum, maximum, median, and kernel estimation of the frequency distribution of the fitness values across 500 controller scenarios.

When the fitness values of the SAMs of cluster L-1 (see Figure 9a) are studied, they exhibit significant differences. To confirm these differences, first, the data gathered are tested and are not normally distributed (Shapiro-Wilk test; $p < 0.05$). Then, the Kruskal-Wallis test is applied, confirming significant differences ($p < 0.05$). Based on the previous results, the performance of the actuator morphologies can be ranked as follows: L155 > L113 > L17 (Dunn's test; $p < 0.05$). Concerning the fitness values of the SAMs of cluster L-3 (see Figure 9b), there are some significant differences. The data collected are not normally distributed (Shapiro-Wilk test; $p < 0.05$), and Dunn's test confirmed that there are significant differences between the L355 soft actuator and the rest of the actuators ($p < 0.05$). However, no significant differences exist between L353 and L373 actuators ($p > 0.05$).

For the fitness values of the SAMs in cluster L-5 (see Figure 9c), significant differences are observed (Shapiro-Wilk test; Kruskal-Wallis test: $p < 0.05$). Accordingly, the SAMs can be ranked as follows: L531 > L51 > L533 (Dunn's test; $p < 0.05$). Similarly, for the fitness values of cluster L-7 (see Figure 9d), significant differences are also evident (Shapiro-Wilk test; Kruskal-Wallis test: $p < 0.05$). The ranking of the actuator morphologies is as follows: L71 > L755 > L775 (Dunn's test; $p < 0.05$).

Once the fittest SAM for each cluster, and thus the fittest substrate, has been identified, the next step is to determine the overall fittest morphology across clusters. Figure 10 presents violin plots comparing the fitness values of the fittest SAMs from each cluster. These plots display the maximum, minimum, and median values, as well as the kernel density estimation of the displacement value distributions across 500 controller scenarios. The Kruskal-Wallis test indicates significant differences among the fitness values ($p < 0.05$). Based on this result, the fitness ranking of the fittest actuator morphologies is as follows: L71 > L155 > L531 > L353 (Dunn's test; $p < 0.05$).

To benchmark the fittest SAM identified in this experiment, a comparison with the fittest SAM found in Reference [21] is presented under the same optimization fitness function, considering the amount of material used. The parameters to evolve CPPNs, the activation function dictionary, and the controller scenarios used in the previous research are employed in this experiment to make a fair comparison. Figure 11 presents violin plots comparing the fitness values of the fittest SAM found in the previous work and the fittest SAM found in this experiment. By employing the Wilcoxon test, it is possible to confirm the existence of significant differences between the two approaches ($p < 0.05$).

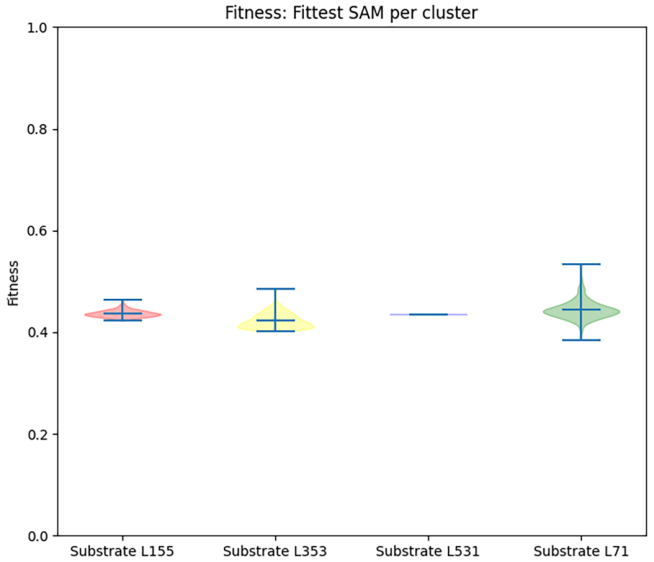

FIGURE 10 | Fittest SAMs in terms of maximizing the displacement and minimizing the number of voxels per cluster.

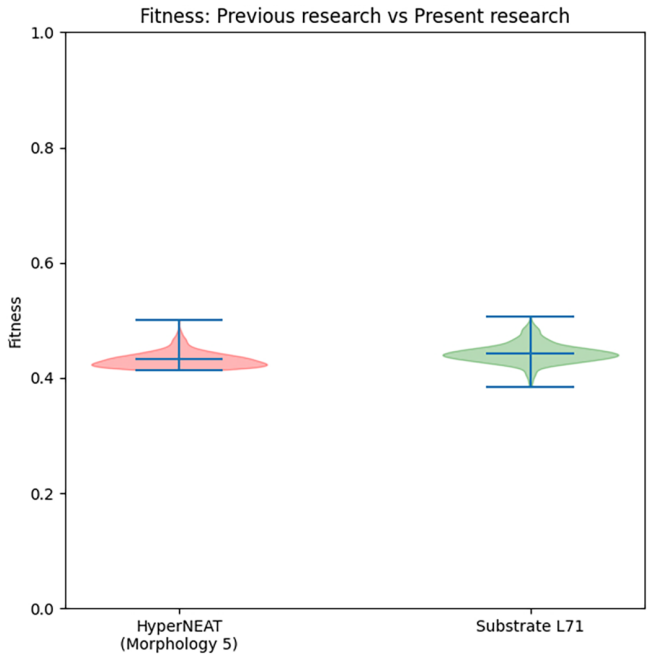

FIGURE 11 | Fittest SAM in terms of maximizing the displacement and minimizing the number of voxels found in: Previous research (left), and present research (right).

The results show that the substrate identified in this research outperformed the one used in Reference [21]. Specifically, the SAM discovered in this experiment demonstrates a superior trade-off between maximizing displacement and minimizing voxel count, regardless of the controller scenario. This improvement is evident when compared to the fittest SAM reported in previous research (see Table 2). Therefore, HyperNEAT with a substrate whose configuration is layer one with seven neurons, layer two with one neuron, outperforms the implementation of HyperNEAT utilized in Reference [21] in terms of generating

TABLE 2 | Number of voxels, mean displacement, and fitness value of the fittest morphologies found in previous research and this research.

Approach	Number of voxels	Mean displacement	Fitness value
Previous research	116	3.6879	0.4320
This research	113	4.1660	0.4433

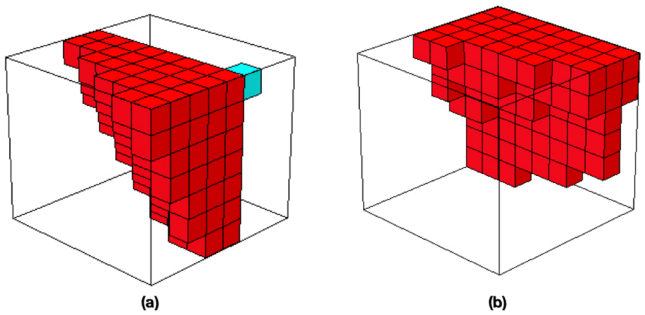

FIGURE 12 | Fittest morphology found in: (a) Previous research, (b) present research.

SAMs capable of maximizing the displacement, preserving the number of voxels at a minimum.

Finally, Figure 12 shows the fittest SAMs found in Reference [21] and in this research. The red cubes represent active (or contractile) voxels, whereas the blue cube represents a passive voxel. Regarding the fittest SAM found in previous research (see Figure 12a), its shape is a triangular-like prism with one passive voxel at the top. Arguably, the voxel asymmetry towards the top vertex observed in the SAM generates a stable displacement despite the variations in the controller scenario. On the other hand, the fittest SAM found in this research (see Figure 12b) exhibits an irregular shape with a descendant voxel gradient towards the bottom of the SAM. Similarly to the SAM found in previous research, the observed voxel asymmetry possibly produces a stable structure during displacement regardless of the controller scenario.

6 | Conclusions

This research focused on finding a substrate configuration for HyperNEAT that allows it to produce higher performing SAMs, simultaneously maximizing displacement and minimizing the number of voxels used. The SAMs produced by each substrate were simulated in a physics engine called Voxelyze. They were evaluated considering two metrics: (i) the maximum displacement observed, and (ii) the trade-off between the displacement and the number of voxels composing the morphologies. During experimentation, 84 different substrates were evolved for 1000 generations, and the champions that emerged were then tested utilizing 500 diverse controller scenarios generated at random. Under each metric, the performance of the fittest SAM found, and consequently the fittest substrate, was compared against the performance of the fittest SAM found under different research methodology published previously [21]. The results indicate that, under both metrics, the substrates found in this investigation outperformed those obtained from previous research. Therefore, the substrate's design (i.e., neuron allocation, number of layers, and number of neurons per layer) plays a crucial role in the algorithm's performance. When the morphologies are compared, although the number of voxels is similar, their shape significantly differs, which suggests that slight differences in substrates induce an utterly different design outcome.

It is noteworthy here that this investigation focused only on the optimization of morphologies. The controllers used throughout the experiments were randomly selected to replicate the inherent uncertainties of real-world applications, especially in the field of biology (similar to the process followed by [33]). Nonetheless, the approach of utilizing a "dictionary" of randomly produced control strategies has its merits, as it imposes more evolutionary pressure on the candidate morphologies to be robust despite the control strategy applied.

Future work can build on the results and insights gained from this research. For example, additional periodic functions, such as tangent or cosine, could be incorporated into the activation function dictionary to explore a broader range of morphological design patterns. Moreover, given the importance of substrate design, future studies could compare the substrate identified in this work with an approach that evolves substrate topologies, such as the *ES-HyperNEAT* method [37]. Finally, the methodology presented in this study can be extended to more complex domains within soft actuator design, including applications in medical devices. Future work involving the translation of simulated designs into real-world implementations could determine whether the statistically significant differences observed here also correspond to meaningful practical outcomes.

Acknowledgments

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No. 101070328. UWE researchers were funded by the UK Research and Innovation grant No. 10044516.

Disclosure

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. D. C. Rose, J. Lyon, A. de Boon, M. Hanheide, and S. Pearson, "Responsible Development of Autonomous Robotics in Agriculture," *Nature Food* 2, no. 5 (2021): 306–309, https://doi.org/10.1038/s43016-021-00287-9.

2. S. A. H. Perez, K. Harada, and M. Mitsuishi, "Haptic Virtual Fixtures to Assist Endonasal micro Robotic Surgery Through Virtual Reality Simulation," in *Proceedings of the 2018 International Symposium on Micro-NanoMechatronics and Human Science (MHS)* (IEEE, 2018), 1–3.

3. C. Lee, M. Kim, Y. J. Kim, et al., "Soft Robot Review," *International Journal of Control, Automation and Systems* 15 (2017): 3–15.

4. H. Lipson, "Challenges and Opportunities for Design, Simulation, and Fabrication of Soft Robots," *Soft Robotics* 1, no. 1 (2014): 21–27.

5. D. Rus and M. T. Tolley, "Design, Fabrication and Control of Soft Robots," *Nature* 521, no. 7553 (2015): 467–475.

6. H. Wang, M. Totaro, and L. Beccai, "Toward Perceptive Soft Robots: Progress and Challenges," *Advanced Science* 5, no. 9 (2018): 1800541.

7. D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft Robotics: Biological Inspiration, State of the Art, and Future Research," *Applied Bionics and Biomechanics* 5, no. 3 (2008): 99–117.

8. C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft Robot Arm Inspired by the octopus," *Advanced Robotics* 26, no. 7 (2012): 709–727.

9. N. W. Bartlett, M. T. Tolley, J. T. Overvelde, et al., "A 3d-Printed, Functionally Graded Soft Robot Powered by Combustion," *Science* 349, no. 6244 (2015): 161–165.

10. E. W. Hawkes, L. H. Blumenschein, J. D. Greer, and A. M. Okamura, "A Soft Robot That Navigates Its Environment Through Growth," *Science Robotics* 2, no. 8 (2017): eaan3028.

11. B. Mazzolai, G. Meloni, and A. Degl'Innocenti, "Can a Robot Grow? Plants Give Us the Answer," in *Bioinspiration, Biomimetics, and Bioreplication 2017*, vol. 10162 (SPIE, 2017), 24–33.

12. A. Sadeghi, A. Mondini, E. Del Dottore, et al., "A Plant-Inspired Robot With Soft Differential Bending Capabilities," *Bioinspiration & Biomimetics* 12, no. 1 (2016): 015001.

13. R. Mestre, T. Patiño, and S. Sánchez, "Biohybrid Robotics: From the Nanoscale to the Macroscale," *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology* 13, no. 5 (2021): e1703.

14. Y. Morimoto, H. Onoe, and S. Takeuchi, "Biohybrid Robot Powered by an Antagonistic Pair of Skeletal Muscle Tissues," *Science Robotics* 3, no. 18 (2018): eaat4440.

15. A. Schulz, C. Sung, A. Spielberg, et al., "Interactive Robogami: An End-To-End System for Design of Robots With Ground Locomotion," *International Journal of Robotics Research* 36, no. 10 (2016): 1131–1147, https://doi.org/10.1177/0278364917723465.

16. J. Hiller and H. Lipson, "Dynamic Simulation of Soft Multimaterial 3d-Printed Objects," *Soft Robotics* 1, no. 1 (2014): 88–101.

17. K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks Through Augmenting Topologies," *Evolutionary Computation* 10, no. 2 (2002): 99–127, https://doi.org/10.1162/106365602320169811.

18. J. E. Auerbach and J. C. Bongard, "Evolving Complete Robots With Cppn-Neat: The Utility of Recurrent Connections," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11* (Association for Computing Machinery, 2011).

19. P. Reyes and M. J. Escobar, "Neuroevolutive Algorithms for Learning Gaits in Legged Robots," *IEEE Access* 7 (2019): 142406–142420, https://doi.org/10.1109/ACCESS.2019.2944545.

20. K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks," *Artificial Life* 15, no. 2 (2009): 185–212, https://doi.org/10.1162/artl.2009.15.2.15202.

21. H. Alcaraz-Herrera, M. A. Tsompanas, I. Balaz, and A. Adamatzky, "Neuroevolution Algorithms Applied in the Designing Process of Biohybrid Actuators," *Proceedings of the 17th International Symposium of Intellignet Distributed Computing 2024 (IDC'24)* (2024), https://doi.org/10.48550/arXiv.2408.07671.

22. N. Cheney, R. MacCurdy, J. Clune, and H. Lipson, "Unshackling Evolution: Evolving Soft Robots With Multiple Materials and a Powerful Generative Encoding," *SIGEVOlution* 7, no. 1 (2014): 11–23, https://doi.org/10.1145/2661735.2661737.

23. K. Sims, "Evolving Virtual Creatures," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (Association for Computing Machinery, 1994), 15–22.

24. S. Gupta and E. Singla, "Evolutionary Robotics in Two Decades: A Review," *Sadhana* 40 (2015): 1169–1184.

25. P. Funes and J. B. Pollack, "Computer Evolution of Buildable Objects for Evolutionary Design by Computers," *Evolutionary Design by Computers* 1999 (1999): 387–403.

26. G. B. Parker, A. S. Anev, and D. Duzevik, "Evolving Towers in a 3-Dimensional Simulated Environment," in *Proceedings of the 2003 Congress on Evolutionary Computation CEC'03*, vol. 2 (IEEE, 2003), 1137–1144.

27. G. B. Parker, D. Duzevik, A. S. Anev, and R. Georgescu, "Morphological Evolution of Dynamic Structures in a 3-Dimensional Simulated Environment," in *Proceedings of the 2007 International Symposium on Computational Intelligence in Robotics and Automation* (IEEE, 2007), 534–540.

28. J. D. Hiller and H. Lipson, "Evolving Amorphous Robots," in *Alife* (Citeseer, 2010), 717–724.

29. F. Tanaka and C. Aranha, "Co-Evolving Morphology and Control of Soft Robots Using a Single Genome," in *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2022), 1235–1242.

30. J. Drchal, J. Koutnik, and M. Snorek, "Hyperneat Controlled Robots Learn How to Drive on Roads in Simulated Environment," in *Proceedings of the 2009 IEEE Congress on Evolutionary Computation* (IEEE, 2009), 1087–1092.

31. K. O. Stanley, "Compositional Pattern Producing Networks: A Novel Abstraction of Development," *Genetic Programming and Evolvable Machines* 8, no. 2 (2007): 131–162, https://doi.org/10.1007/s10710-007-9028-8.

32. UCF, E.C.R.G, "The Hybercube-Based Neuroevolution of Augmenting Topologies (Hyperneat) Users Page," 2016, http://eplex.cs.ucf.edu/hyperNEATpage/.

33. S. Kriegman, D. Blackiston, M. Levin, and J. Bongard, "A Scalable Pipeline for Designing Reconfigurable Organisms," *Proceedings of the National Academy of Sciences* 117, no. 4 (2020): 1853–1859, https://doi.org/10.1073/pnas.1910837117.

34. M. A. Tsompanas and I. Balaz, "Outline of an Evolutionary Morphology Generator Towards the Modular Design of a Biohybrid Catheter," *Frontiers in Robotics and AI* 11 (2024): 1337722.

35. J. Hiller and H. Lipson, "Automatic Design and Manufacture of Soft Robots," *IEEE Transactions on Robotics* 28, no. 2 (2011): 457–466.

36. X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (Proceedings of Machine Learning Research, 2011), 315–323, https://proceedings.mlr.press/v15/glorot11a.html.

37. S. Risi and K. O. Stanley, "An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density, and Connectivity of Neurons," *Artificial Life* 18, no. 4 (2012): 331–363, https://doi.org/10.1162/ARTL_a_00071.