Incremental growth on Compositional Pattern Producing Networks based optimization of biohybrid actuators

Michail-Antisthenis Tsompanas[1][0000−0002−6607−7831]

School of Computing & Creative Technologies, University of the West of England, Bristol, United Kingdom `Antisthenis.Tsompanas@uwe.ac.uk`

Abstract. One of the training methods of Artificial Neural Networks is Neuroevolution (NE) or the application of Evolutionary Optimization on the architecture and weights of networks to fit the target behaviour. In order to provide competitive results, three key concepts of the NE methods require more attention, i.e., the crossover operator, the niching capacity and the incremental growth of the solutions' complexity. Here we study an appropriate implementation of the incremental growth for an application of NE on Compositional Pattern Producing Networks (CPPNs) that encode the morphologies of biohybrid actuators. The target for these actuators is to enable the efficient angular movement of a drug-delivering catheter in order to reach difficult areas in the human body. As a result, the methods presented here can be a part of a modular software pipeline that will enable the automatic design of Biohybrid Machines (BHMs) for a variety of applications. The proposed initialization with minimal complexity of these networks resulted in faster computation for the predefined computational budget in terms of number of generations, notwithstanding that the emerged champions have achieved similar fitness values with the ones that emerged from the baseline method. Here, fitness was defined as the maximum deflection of the biohybrid actuator from its initial position after 10 seconds of simulated time on an open-source physics simulator. Since, the implementation of niching was already employed in the existing baseline version of the methodology, future work will focus on the application of crossover operators.

Keywords: Biohybrid machines · Compositional Pattern Producing Networks · optimization · evolutionary algorithms · machine learning

1 Introduction

Machine learning and, particularly, Artificial Neural Networks (ANNs) have become an increasingly prominent method for building accurate and efficient models with minimal required effort and background knowledge of the under study system. The widespread acceptance of ANNs is attributed to well established computational methods of training these networks, i.e., backpropagation, alongside other factors, such as big data availability. Another noteworthy and interesting training method of ANNs is Neuroevolution (NE), which proved to produce equally robust models [4].

NE is referring to the methodology of applying principles of Evolutionary Algorithms (EAs) to the process of training and optimizing ANNs [14]. The inspiration behind this technique was drawn from natural evolution, in order to realize a bio-inspired method of optimization. Populations of possible solutions or network instances (i.e. network architectures and connection weights) are produced as the result of simulated evolution and tested against a predefined fitness function. The fittest solutions are selected to mutate and reproduce, in order to provide more possible solutions that are then injected in the following generations, and so forth, until the computational budget is spent or a target efficiency is reached.

The implementation of NE can be employed in the evolution of different kinds of networks, like Compositional Pattern Producing Networks (CPPNs) [13]. Since CPPNs are formalized in a similar way as ANNs, there is no need for extensive changes in well-established methodologies applied on the latter, while favourable results are expected. The main difference between the two types of networks is the activation functions of their nodes; whereas, CPPNs are not restricted in any way around this area, ANNs are mainly employing monotonic functions. As a result, CPPNs are better suited for applications that require the production of complex patterns and structures [3].

Here, the use of CPPNs is studied as a tool for the primary discovery of morphologies of biohybrid machines (BHMs), in a similar way as previous works that delivered promising results [1, 2, 7]. By adopting the open source code developed previously [7], we aim to empower the integration of advanced algorithms during the initial stages of the BHM development, helping identify crude designs of efficient morphologies. The primary objective of our project is to establish a BHM design process and employ this framework to pioneer the development of a ground-breaking medical device, namely a biohybrid catheter capable of delivering pharmaceuticals to challenging to reach areas of the human physiology. In specific, the main goal of the developed software module is to produce BHM actuators that will facilitate robust angular movement of a catheter in the labyrinth-like environment of the circulatory system of humans.

Some NE algorithms, i.e., NeuroEvolution of Augmenting Topologies (NEAT) [15], have been proved to be more efficient than others, because of three critical factors described in the following. These methodologies (i) employ means that enable crossover during evolution without complicated topology analysis, (ii) include niching capacity that is able to protect innovative individuals from premature exclusion and (iii) encourage the incremental growth of complexity in solutions, on account of the initial populations being structures of minimal complexity. Reviewing the algorithmic approach implemented in [7], we could locate a variant method for niching, but there were no provisions for the other two characteristic factors. While the crossover factor was not included intentionally for simplicity reasons based on the authors' reasoning, we could not pinpoint the motivation behind initialization of population with networks that were not of minimal complexity.

As a result, this work takes into consideration the initialization of the populations with minimal structure networks and compares the outputs with the method followed by the original work [7]. To test the appropriateness of starting at minimal dimensions and, as a result, allowing incremental growth of complexity, the open source code was altered towards including that characteristic. Moreover, a comparative analysis of the champions discovered was performed, in order to justify previous findings [15], i.e., that this characteristic enables higher effectiveness. The results show that starting at minimal dimension provides solution candidates with similar effectiveness in terms of fitness, however a significant acceleration of the computation for the same target of total generations is achieved. This can be attributed to the lower complexity of the networks being managed throughout the computational process.

The rest of the manuscript is organized as in the following. Section 2 provides some background on NE and the aspects that render it a suitable surrogate of other training methods, along with basic characteristics of CPPNs. Section 3 describes the methodology used in this study, i.e., details of the simulators, algorithms and the proposed initialization method. Then, Section 4 presents the results of the tests for both initialization methods and Section 5 concludes this study.

2 Background

Some typical paradigms of NE methods [5, 9] assumed a fixed structure for the networks that were studied and their dimensionality was manually set before evolution began. One hidden layer was included with neurons fully connected with the input and output neurons, while the evolution was assessing the weights of the connections. Because of the fixed topology of networks, crossover and mutation operators were trivially applied to the weights of the connections and optimization enabled the training of networks towards a desired behaviour.

Nevertheless, the weights of network connections are not a sole indicator of how neural networks function. The structure that defines the number of nodes and how they are connected, plays a significant role as well. Thus, enhanced NE methods were proposed under the term Topology and Weight Evolving Artificial Neural Networks (TWEANNs) employing evolution of both topologies and connection weights [8, 11]. These techniques take advantage of increasing structural complexity through mutations. Although the addition of randomly formed nodes may cause a decrease in fitness initially, the modulation of connection weights during subsequent evolution steps can result in an ultimately higher fitness.

An innovative NE method, named NEAT [15] has motivated a large range of variants relevant till this day [10] and managed to outperform previous methods, as it was more thoroughly designed, in order to exploit the fact that smaller dimensionality networks can be optimized faster. It proved to be a superior methodology, because, according to the authors [15], (i) it would include a crossover operator, while previous versions did not, (ii) it would safeguard the innovation in network architecture with initial low fitness against premature exclusion of these promising architectures and (iii) it would allow for incremental growth on the complexity of networks by initializing populations at minimal complexities. The authors tested what each of these three aspects contributed to the overall efficiency and concluded that all aspects and their combinations were significant for providing even better efficiency.

In a similar setting, the conclusions in [16] argue that the robustness in evolutionary methods is achieved by an initial population of minimal and non-complicated genomes. As generations lapse these genomes undergo the introduction of additional genes that serve as enablers to the expansion of the search space. Therefore, novel dimensions are introduced and evolutionary exploration is initially exploring a relatively small and manageable space, before moving to additional dimensions that are included only if necessary, i.e., after the search to the given search space dimensionality stagnates. This incremental process is called complexification and is a technique used to partially tackle the curse of dimensionality. Moreover, complexification is not limited to enhance the results of NE methods, but, also, the efficiency of more typical evolutionary algorithms. In our previous works on optimization of individuals with variable genome lengths [17, 18], the ability to optimize and complexify the genome were both included in the methodology to produce fitter solutions, while the initial populations were of minimal genome lengths.

CPPNs are similar to ANNs, with the main difference being the relaxation of rules on the activation functions of nodes of the former type of networks. Namely, CPPNs are better suited for generating complex patterns and structures [3], since the graph that represents them define associations between a variety of functions (or activation functions) that are depicted as nodes (as depicted in an example of CPPN in Fig. 1). Connections are characterised by weights that determine the impact of each node output to the input of the next layer node. In cases where multiple connections terminate to the same node, all weighted outputs of the previous nodes are aggregated and used as inputs to the current node. An additional difference, that is also essential for CPPNs' functionality is that the topology of the graph is not restricted in any way, thus, enabling higher levels of representation liberty that achieves more complex patterns.

Another, more semantic difference between these networks is that while ANNs emulate the functionality of human brain in learning, CPPNs simulate a completely different biological process, namely, the developmental process [13]. Consequently, another attractive feature of CPPNs in applications of producing patterns is that when they are queried on an absolute coordinate frame (i.e. x, y in a two-dimensional space), there is no need for the specific definition of local interactions within the representation. When using each specific point in a Cartesian coordinate system as an input of CPPN, the outputs will formulate a pattern without the phenotype (i.e. the CPPN) requiring local interactions or temporal sequencing. The network will use the coordinates of all points in a

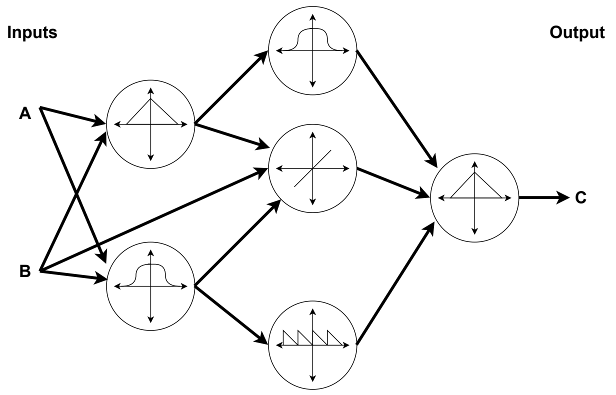


Fig. 1: An example of a CPPN with two inputs and one output. Each node represents a specific function, while connections are weighted and represent the sum of weighted intakes of each function.

space as inputs and the output provided will precisely specify the entities and characteristics of the specific location in space that was used as input every time.

3 Methods

To evaluate some primary morphologies of BHM catheter actuators with no detailed investigation of all the possible components of the underlying mechanisms and no biotechnology laboratory overheads, the *in silico* investigation is preferred. Thus, a simulator that would be capable of mimicking behaviours of truly heterogeneous materials is required. Thus, Voxelyze [6] was employed as the test-bed of morphologies composed of different materials with diverse physical properties, such as Poisson's ratio, stiffness, density and friction coefficients. Moreover, Voxelyze has the capacity to simulate external forces along with volumetric actuation of entities; a characteristic that enables the representation of novel architectures like the ones found in BHMs, namely accommodating contracting muscle cells. In Voxelyze each elementary volume, designated as a voxel, can encode a different material and the distance between neighboring voxels is modeled as Euler-Bernoulli beams. Moreover, additional environmental settings can be defined to illustrate specific scenarios, such as gravitational acceleration, collision rules and friction between the range of different voxels and a static floor. Here, to follow the scenarios investigated in previous studies [7], two types of voxels were outlined with the parameters depicted in Table 1. Specifically, one type is an active voxel that can contract and provide the energy required for movement; whereas, the other type is a passive voxel with similar physical properties, but, no motion capacity included.

Voxelyze acts as a test-bed for the fitness function, namely, morphologies of $8 \times 7 \times 7$ voxels in a Cartesian grid are evaluated based on their simulated

Table 1: Parameters of active and passive voxel.

Parameters	**Active voxel**	**Passive voxel**
Elastic modulus (MPa)	5	5
Density (kg/m^3)	1,000,000	1,000,000
Poisson's ratio	0.35	0.35
Coefficient of Thermal Expansion ($1/°C$)	0.01	0
Coefficient of static friction	1	1
Coefficient of dynamic friction	0.5	0.5

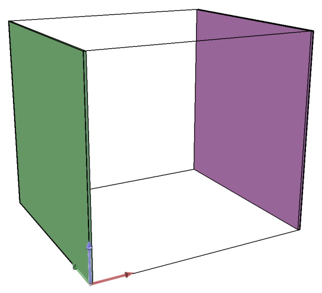


Fig. 2: Boundaries in Voxelyze representing a fixed end and a free end of a catheter actuator.

behaviours. The morphologies of a maximum of 392 voxels are constructed in the virtual environment of Voxelyze, the simulation starts and after 1 second of initial simulation time, the morphology will be settled from possible gravitational motion into the starting point for the evaluation. Following this, a further 10 seconds are simulated, in order to record the final displacement of the whole morphology and calculate the deflection achieved by the simulated actuator. Note here, that in order to better represent the scenario of a catheter actuator, one end of the morphology is fixed (the YZ plane for $x = 0$, depicted as the green plane in Fig. 2), whereas the other end is free to perform translational and rotational motion based on the global behavior derived from all the individual active voxels' activity (the YZ plane for $x = 8$, depicted as the purple plane in Fig. 2). The fitness for each candidate morphology is provided by the total deflection at $t = 10s$ of simulated time, i.e., the distance of the projection on any YZ plane of one of the top and outer voxels that are adjacent to the free end of the morphology.

The indirect encoding concept that exploits CPPNs is utilized to symbolize candidate morphologies in the evolutionary optimization process. To decode the individuals, the Cartesian coordinates (x, y and z) are used as inputs for the network (in addition with the distance from the center of the available space d and a bias b) and the output represents the voxel type for the respective combination of coordinates (as illustrated in Fig. 3). After querying the network

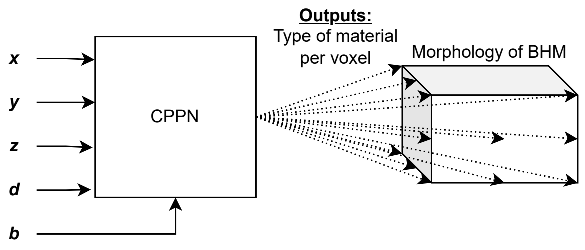


Fig. 3: Decoding of the CPPN into a morphology by using coordinates in a three-dimensional space as inputs and material types as outputs. The functionality of the CPPN is illustrated in Fig. 1

with all the possible combinations on the aforementioned $8 \times 7 \times 7$ grid, the types of all the 392 voxels are provided, thus, the morphology is decoded into the 3D space and can be inserted to Voxelyze to calculate its fitness. Note here, that the CPPN output can denote active or passive voxels, but, also empty space in order to permit more elaborate morphologies.

The nodes in the hidden layers of the CPPNs can represent any of the predefined mathematical functions (i.e., sine, absolute value, negative absolute value, square, negative square, square root and negative square root). The weight of the connection between two nodes represent the multiplication factor of the outgoing result. When several connections terminate to the same node, then, the addition of the weighted in-going results is used as input to the node's activation function.

For the evolutionary algorithm, the genotype of the individuals is in a form of a CPPN, whereas the phenotype is in a form of a 3D morphology of the BHM actuator, which is derived by querying the CPPN genotype. Following the concept of the open source code [7] and for a clearer implementation, no crossover operator was implemented, however, the Age-Fitness Pareto Optimization (AFPO) algorithm [12] was utilized. Particularly, a population of 50 randomly generated CPPNs was produced through an intricate initialization process. Afterwards, these 50 individuals were decoded into BHM morphologies and evaluated through Voxelyze. Then, 50 additional individuals were created through mutation operations over the initial population. These 50 additional individuals with the inclusion of one more randomly generated individual were evaluated and from the total of 101 available individuals the 50 fittest were selected to comprise the next generation. This would complete one evolution cycle and the new generation would go again through the mutation operator and so forth, until 2000 generations were evaluated.

The mutation process for each individual involves the application of one type out of six possible alternations in the CPPN genotype with a 0.167 probability. The possible alternations are the addition of a node or connection, the modification of a node or connection and the removal of a node or connection. As a result, both the weights of the CPPN and its architecture can be modified to permit training and complexification (or even simplification) of the network. It is noteworthy, that if the decoding of the CPPN genotype produces a morphology phenotype that is already evaluated, the mutation is considered neutral and the process is repeated until a non-neutral mutation is found, for a maximum of 1500 attempts.

The utilization of AFPO [12] provides a basic niching capacity. On the condition that this multi-objective optimization is pursuing the dominance over fitness and age for individuals to survive for consequent generations, premature convergence is avoided. In specific, the individuals are selected based on their higher fitness value and lower genotypic age, in a multi-objective Pareto front optimization. As a result, individuals that have emerged latter in evolution can coexist in the same population with older and fitter individuals, because they are not dominated on the age dimension of the Pareto front.

The final aspect that needs to be clarified is the initialization process, as it was determined one of the three key factors for NE effectiveness. The original initialization process, used in [7] and denoted here as the baseline, begins with building the minimal possible network, namely connecting input and output nodes with edges of weight zero. Then, the mutation operator is executed multiple times, i.e., 10 times for random node addition, 10 times for random connection addition, 5 times for random connection removal, 100 times for random node modification and 100 times for random connection weight modification. After that, the network is pruned to remove any erroneous nodes and connections. The notable operations here are the 10 node additions, the 10 connection additions and the pruning of the network, which can result to a network with a maximum of 10 hidden layer nodes. However, there is no guarantee that this is the minimal or, even, close to the minimal possible structure.

To study the potential of incremental growth in the NE of CPPNs, we altered the aforementioned initialization process to build a population with lower amount of hidden layer nodes. Specifically, we kept the same methodology as described previously, however, we altered the amount of random node additions during initialization from 10 to 2. This small number was enough to allow the production of an initial population with acceptable diversity and a range of hidden layer nodes from zero to 2. This is obviously an initial population of simpler network structures that will more probably permit complexification alongside optimization.

4 Results

In the following, the comparison of the outputs for 10 runs with the same random generator seeds are presented for both initialization processes. Execution times of the two variants are presented in Fig. 4. It is apparent that the original initialization stems a significantly slower evaluation of the 2000 generations, when compared with the minimal structure initialization proposed here (Wilcoxon rank-sum test, $p < 0.001$). In specific, execution times of the original initialization have a mean of 19.71 hours (samples= 10, minmax= (18.69, 20.88),

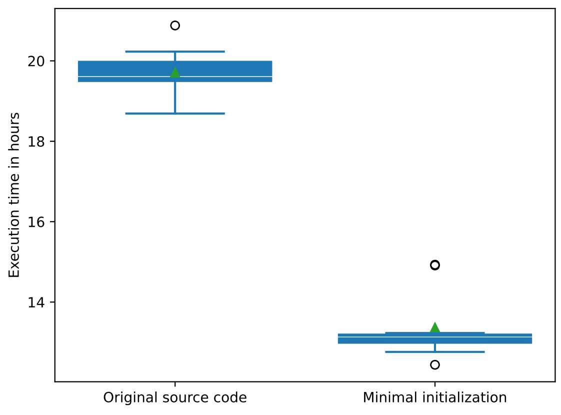


Fig. 4: Execution times for the two variants of the population initialization.

variance= 0.35, skewness= 0.35, kurtosis= 0.0998), while for the proposed minimal initialization a mean of 13.37 hours (samples= 10, minmax= (12.44, 14.93), variance= 0.72, skewness= 1.19, kurtosis= -0.0620) was observed. So, the minimal initialization economizes 6 hours of computational time per run of 2000 generations.

This acceleration can be attributed to the bottleneck in the whole process of finding non-neutral mutated networks. As mentioned previously, after each mutation operation the phenotype (3D morphology) produced from the genotype (CPPN) was compared with the phenotype of the pre-mutated genotype. If the new CPPN would produce the same morphology, a new mutation would be attempted, unless 1500 unsuccessful attempts are executed. We realised that this technique introduces significant overheads to the whole evolutionary computation process. Moreover, it is evident that this procedure requires more computational resources when assessing networks with higher numbers of nodes in the hidden layer. On the contrary, when assessing less complex networks the discovery of non-neutral mutations would happen faster, as realised from the execution times in Fig. 4.

In order to compare the complexity of the networks within the initial population for both variants of the initialization process, Fig. 5 illustrates the number of nodes in the hidden layer. These data are collected for all 50 initial individuals for each of the 10 runs of both variants (i.e. 500 individuals). As expected the original source code initialization produces more complex networks, with a mean of 7.734 nodes (variance= 2.436), whereas, the proposed minimal initialization produces simplified networks with a mean of 1.9 nodes (variance= 0.110) in the hidden layer.

Moreover, Fig. 6 depicts the distribution of the nodes in the networks within the population after 2000 generations of evolutionary optimization. Note that

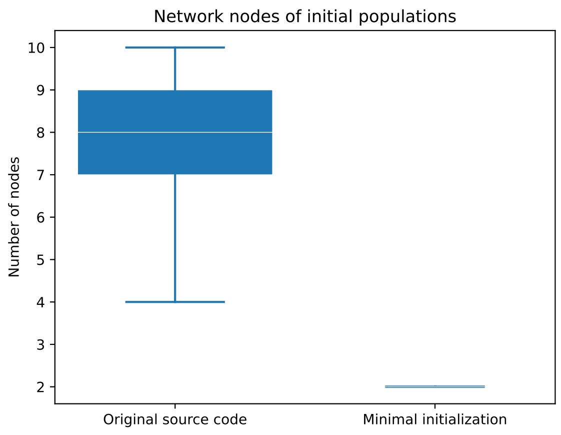


Fig. 5: Distributions of nodes of networks in the initial populations.

the data collected include all 50 final individuals for each of the 10 runs of both variants (i.e. 500 individuals). While the final populations of the baseline method have higher mean (=8.49) and variance (=10.35) in terms of network nodes, the proposed methodology illustrates sufficient diversity in the final populations as well (mean= 3.45, variance= 6.80). Nonetheless, by comparing the means of the initial and final populations, it can be seen that the complexification of the networks is more prominent in the minimal initialization process. Fig. 7 outlines the distribution of the complexity (or number of nodes) of the fittest individuals at the end of the 2000 generations. It can be concluded here that champions are found in the higher network complexity available in the populations for both variations. Moreover, the minimal initialization methodology manages to discover champions encoded by networks with up to c. 10 hidden layer nodes, despite the starting point of only 2 nodes.

To investigate the impact of incremental growth that clearly manifested in the aforementioned, Table 2 describes the fitness of the champions (i.e., fittest individuals) after 2000 generations for both variants. Each line illustrates the fitness for both runs with the same random generator seed. The original initialization method is producing fitter champions in half of the runs (samples= 10, minmax= (0.253, 0.433), mean= 0.346, variance= 0.00345, skewness= -0.115, kurtosis= -0.8651). On the other hand, the minimum complexity initialization is not falling far behind and, despite the minimal networks present in the first populations, it manages to produce comparable champions (samples= 10, minmax= (0.176, 0.469), mean= 0.333, variance= 0.00679, skewness= -0.275, kurtosis= -0.2162). Moreover, comparing the distributions of the fitness of both sets of champions, we can not reject the null hypothesis that the two sets of fitnesses are drawn from the same distribution (Wilcoxon rank-sum test, $p = 0.879$). Thus, no advantage

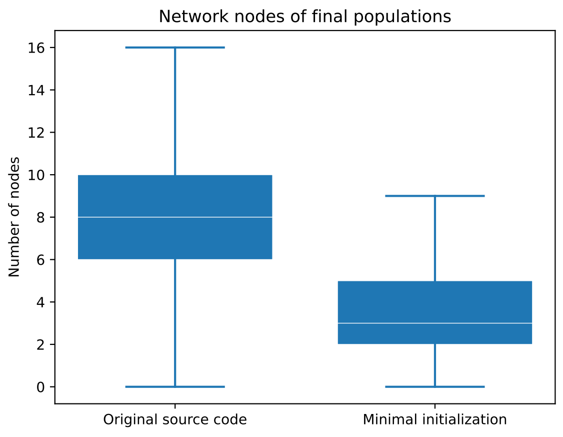


Fig. 6: Distributions of nodes of networks in the final populations.

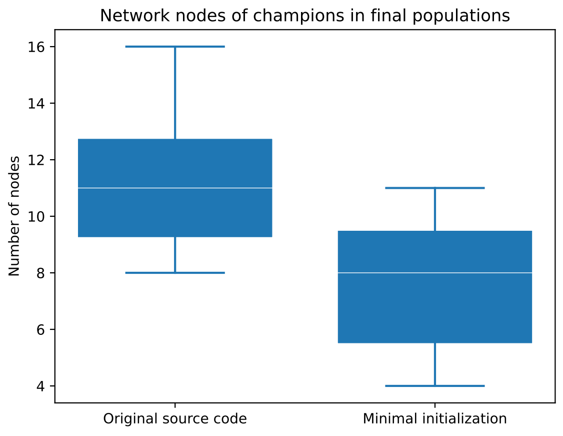


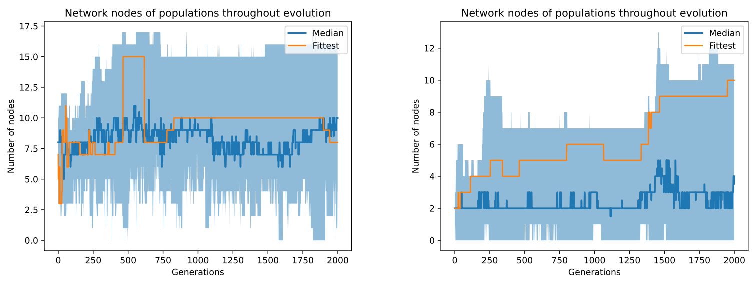
Fig. 7: Distributions of nodes of networks of the champions in the final populations.

is apparent for starting from a minimal or complex population, other than the acceleration in computations.

To demonstrate the incremental growth of network complexities throughout evolution, the range of the amount of nodes for whole populations, the median and the champions are provided for both variations in runs with seeds 52 and 58 (the fittest champions of both variations) in Figs. 8 and 9 respectively. For illustration reasons, the morphology of the highest performing champion of all

Table 2: Fitness of the champions after 2000 generations for both initialization procedures (higher is better and indicated by bold fonts).

Random seed	Original init.	Minimum init.
50	**0.382**	0.352
51	0.253	**0.255**
52	0.328	**0.469**
53	0.266	**0.417**
54	**0.325**	0.176
55	**0.419**	0.293
56	0.341	**0.375**
57	**0.375**	0.344
58	**0.433**	0.303
59	0.337	**0.349**



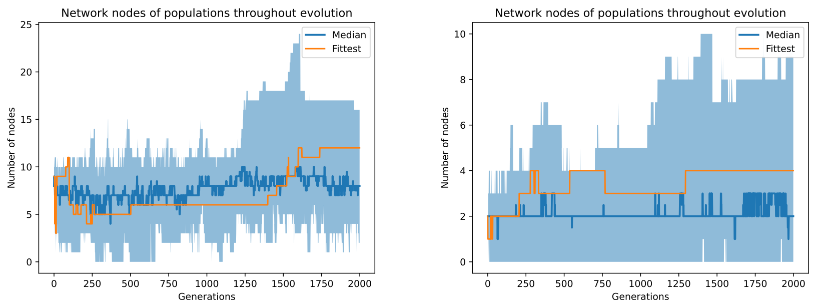
(a) (b)

Fig. 8: Evolution of CPPN node numbers for runs with seed 52 for (a) the original initialization process and (b) the proposed minimal initialization.

the runs (found in the minimal initialization variant with seed 52) is illustrated in Fig. 10 (a) at its initial position and in Fig. 10 (b) at its final position (after 10s of simulation time). Moreover, the CPPN that was evolved from the algorithm to encode this morphology is depicted in Fig. 11, where the input and output nodes, activation functions and connection weights are defined.

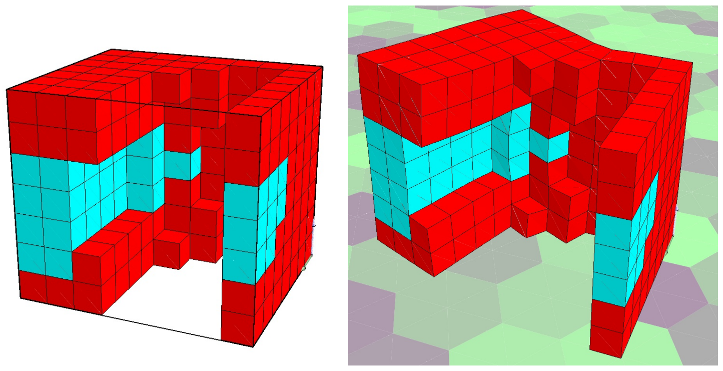
5 Discussion

The implementation of CPPNs, as an indirect representation of individuals, in evolutionary optimization has proved to be quite efficient. However, the initialization of a previously published work did not follow the incremental growth concept that is of paramount importance to the efficiency of NE methods. To prove the effects of starting at minimal complexity and, thus, allowing incremental growth through evolution (or complexification) along optimization, we altered the initialization methodology to apply this concept. The software framework



(a) (b)

Fig. 9: Evolution of CPPN node numbers for runs with seed 58 for (a) the original initialization process and (b) the proposed minimal initialization.



(a) (b)

Fig. 10: The highest performing champion of this study simulated in the Voxelyze environment at its (a) initial and (b) final position.

was an application of NE optimization on the first module of a design pipeline for a BHM catheter actuator.

The results show that there is no significant advantage in the fitness of the emerging champions for the populations that started from higher complexity for the baseline implementation. On the contrary, the methodology that employed minimal initialization performed at the same degree of efficiency, maintained high diversity in the final populations and required less computational resources to reach the same degree of efficiency. As a result, aspects of future work will be the comparison of the two methods with the same computational budget, but in terms of wall-time and not amount of generations. Moreover, the effect of the

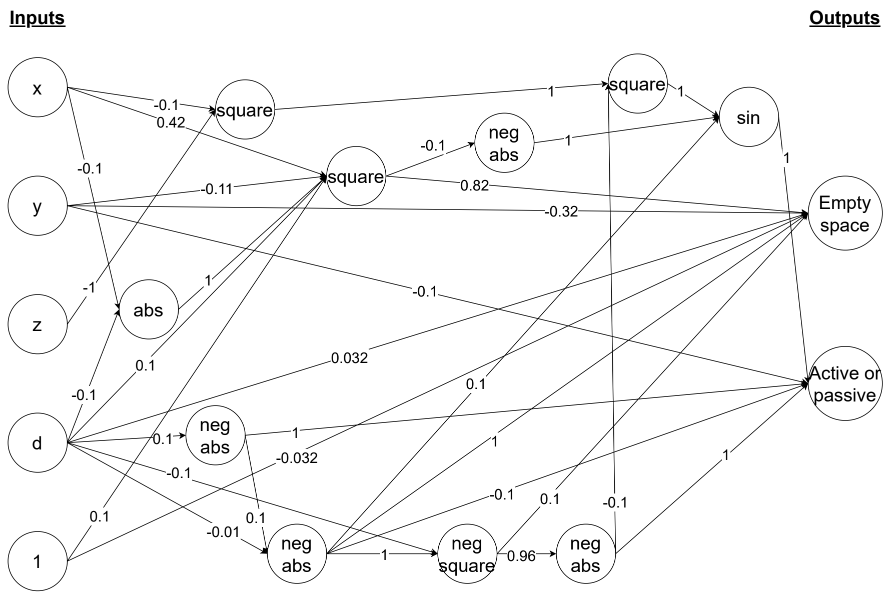


Fig. 11: The CPPN structure that decodes into the morphology of the highest performing champion of this study illustrated in Fig. 10

crossover will be studied as it has been proved [15] that enhances the capabilities of NE in well-established benchmarks.

Acknowledgement

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.

Code availability

The code to reproduce the results can be found here: `https://github.com/Antisthenis/reconfigurable_organisms/tree/biomeld_dev2`

References

1. Cheney, N., Bongard, J., Lipson, H.: Evolving soft robots in tight spaces. In: Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation. pp. 935–942 (2015)
2. Cheney, N., MacCurdy, R., Clune, J., Lipson, H.: Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. ACM SIGEVOlution **7**(1), 11–23 (2014)

3. Clune, J., Lipson, H.: Evolving 3d objects with a generative encoding inspired by developmental biology. ACM SIGEVOlution **5**(4), 2–12 (2011)
4. Galván, E., Mooney, P.: Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Transactions on Artificial Intelligence **2**(6), 476–493 (2021)
5. Gomez, F.J., Miikkulainen, R., et al.: Solving non-markovian control tasks with neuroevolution. In: IJCAI. vol. 99, pp. 1356–1361. Citeseer (1999)
6. Hiller, J., Lipson, H.: Dynamic simulation of soft multimaterial 3d-printed objects. Soft robotics **1**(1), 88–101 (2014)
7. Kriegman, S., Blackiston, D., Levin, M., Bongard, J.: A scalable pipeline for designing reconfigurable organisms. Proceedings of the National Academy of Sciences **117**(4), 1853–1859 (2020)
8. Lee, C.H., Kim, J.H.: Evolutionary ordered neural network with a linked-list encoding scheme. In: Proceedings of IEEE International Conference on Evolutionary Computation. pp. 665–669. IEEE (1996)
9. Moriarty, D.E., Mikkulainen, R.: Efficient reinforcement learning through symbiotic evolution. Machine learning **22**, 11–32 (1996)
10. Papavasileiou, E., Cornelis, J., Jansen, B.: A systematic literature review of the successors of "neuroevolution of augmenting topologies". Evolutionary Computation **29**(1), 1–73 (2021)
11. Pujol, J.C.F., Poli, R.: Evolving the topology and the weights of neural networks using a dual representation. Applied Intelligence **8**, 73–84 (1998)
12. Schmidt, M.D., Lipson, H.: Age-fitness pareto optimization. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. pp. 543–544 (2010)
13. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic programming and evolvable machines **8**, 131–162 (2007)
14. Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R.: Designing neural networks through neuroevolution. Nature Machine Intelligence **1**(1), 24–35 (2019)
15. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
16. Stanley, K.O., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. Journal of artificial intelligence research **21**, 63–100 (2004)
17. Tsompanas, M.A., Bull, L., Adamatzky, A., Balaz, I.: Evolutionary algorithms designing nanoparticle cancer treatments with multiple particle types [application notes]. IEEE Computational Intelligence Magazine **16**(4), 85–99 (2021)
18. Tsompanas, M.A., Bull, L., Adamatzky, A., Balaz, I.: Metameric representations on optimization of nano particle cancer treatment. biocybernetics and biomedical engineering **41**(2), 352–361 (2021)