On the Impact of Morphological Encoding for Evolutionary Robotics in ARIEL

Olivier Moulin[1,3], Davide Pasero[2,3], Lukas Bierling[2,3], and Kevin Godin-Dubois[1]

[1] Vrije Universiteit Asmterdam o.moulin@vu.nl
[2] Universiteit van Amsterdam davide.pasero@student.uva.nl
[3] indicates first authors, who have contributed equally to this work.

Abstract. In Evolutionary Computing, and more precisely in the Evolutionary Robotics field, the characteristics of each individual are stored in the genotype. The phenotype, the actual shape of the individual, is the result of the decoding of the genotype. In this paper, we are looking at different representations for the genotypes, and especially indirect encodings such as L-Systems or CPPNs, alongside the more traditional Tree representation. We also compare these with the NDE baseline, implemented in ARIEL. By experimenting with the different representations, we demonstrate how much of an impact it has on the type of individuals produced, on their novelty and on the performance on different evolutionary tasks. It makes the choice of a representation for the genotype a critical point when implementing Evolutionary Robotics algorithms. We highlight that overall the Tree and CPPN representations perform better on the selected tasks. However, each encoding demonstrated relative expertise in diverse areas, providing experimenters with an actionable bias depending on their requirements.

Keywords: Evolutionary Computing, Evolutionary Robotics, Morphogenetic Engineering, Indirect encoding

1 Introduction

Over its existence, the field Evolutionary Robotics has explored numerous ways of representing robots genes or genotype. In this paper, we are looking at assessing how each genotype representation (NDE, Tree, L-Systems, CPPN) has an impact on the morphological space of the robots created but also on their capacity to evolve. In order to fairly assess the direct impact of the choice of genotype representation, we have decided not to include any brain in the robots, so as to not balance the weaknesses/strengths of each genotype representation. The different genotype representations are evaluated over 4 different tasks which gives good insights on how each approach impacts different aspects of the evolution algorithms: ability to copy an existing robot morphology through evolution, ability to copy changing robot morphologies through evolution and maximum diversity of the population through evolution. We also looked at the level of diversity across the population while trying to replicate a robot morphology through

evolution. We also run another experiment where the fitness function is always set to 0, to see if each representation embed biases regarding the morphology of the individuals produced. This paper is organized as follows, section 2 focuses on explaining how our research work integrates with the other works conducted in the same area, as well as describing the works from which we have taken inspiration. In section 3 we describe the different theoretical elements behind our work, while section 3 details how our experiments were constructed and the assumptions and limits attached to them. Finally, section 4 presents the results gathered from our experiments and section 5 discusses the conclusions that can be drawn from these results and presents ideas for future research.

2 Related work

We want to highlight that another paper addressing a similar type of research has been published by Veenstra et al. [13] and covers experiments around robot locomotion, where both the brain and the body are evolved. In our paper we focus only on the impact of the genotype representation on the phenotype and on the capacities of a given genotype representation to be used to achieve a set of Evolutionary Robotics tasks. The choice to exclude the brain from our experiments was made to avoid having the brain compensate for the deficiencies of a given representation. To conduct our experiments, we have leveraged approaches presented by other papers. The morphological traits used in our paper to describe a given robot as well as to compare the embedded bias in each representation comes from the work from Miras et al. [9], which also demonstrated that generative encodings have a tendency to generate more diverse robots than a direct encoding approach like the tree representation. The Tree Edit Distance used in our paper has been described by Zhang and Shasha [14]. The L-systems concept, which is not specific to Evolutionary Robotics, has been first introduced by Winfried Kurth [7]. The L-Systems representation used in this paper is inspired from the work of Miras et al. [9] which describes how L-Systems can be used to create the genotype of a robot and then generate its corresponding phenotype. Our approach introduces a different crossover from the one described in this paper, allowing crossover to happen at the individual token level in the L-Systems rules instead of happening at the rules level, offering more granularity. The turtle approach used by Miras et al. [9] has also been used for our experiments for building the robots. L-systems have also been used in other Evolutionary Robotics papers by Luo et al. [8] for its tendency to produce a more diverse population and by Bie et al. [2] to highlight how this representation can be used to create robots which can auto-reconfigure to match a given model. The Tree approach is directly derived from the work of Koza and Poli [6], for the representation of the genome as well has the crossover and the mutation operators. The CPPN approach is inspired by the work of Auerbach et al. [1] and it has also been used to evolve robots as described in the work by Raghab Singh [10]. The NDE representation used in this paper is the standard implementation included in ARIEL system [4], used in research on fertility during learning [3].

3 Approach

In Evolutionary Computing, the genome of each individual can be coded in different ways, using either a static representation (direct encoding) or a generative representation (indirect encoding). In the direct encoding, genotypes are reversibly mappable to phenotypes whereas the generative approach involves more complicated computation oftentimes not reversible. The direct encoding approach is based on a series of genes, where each gene or group of genes can be converted to a given characteristic in the individual phenotype. The indirect encoding approach is based on a series of genes and associated rules / functions which transforms the initial genotype into an intermediary phenotype, by applying the rules and functions to the initial genotype. This intermediary phenotype can then be used to create the final phenotype of the individual.

Assessing genome representation impact. Our goal is to see if the choice of different genotype representations have an impact on the individuals created, mutated and reproduced by the Evolutionary Computing algorithm.

EC performance indicators. During the different experiments conducted, we have adjusted the fitness to match the desired behavior of the population, by using either the similarities of the individuals created, when the task assigned was to copy an existing phenotype, or, conversely, the novelty of the individuals created, when the task was focused on maximizing diversity in the population.

3.1 ARIEL system and Evolutionary Computing algorithm

In order to run our experiments, we use the ARIEL system [4], to simulate robots composed of basic components:

CORE central piece of the robot integrating the processor and the battery
HINGE serving as a low-level actuator
BLOCK passive structural component

ARIEL offers the functionalities to simulate and render robots based on a networkx [5] object representation of the phenotype (Figure 1). Each genotype representation assessed in this paper has been designed to produce such a networkx object. In our experiments, the direction is always from the genotype representation to the phenotype. The modifications to each individual are always performed at the genotype level and then the phenotype is created in ARIEL following the process described above. In the following paragraphs, we detail the four representations used in this paper according to their initialization, mutation and crossover operators.

3.2 ARIEL NDE representation

ARIEL implements by default the Neural Developmental Encoding (NDE) representation to generate individuals, as described in [3]. This representation will

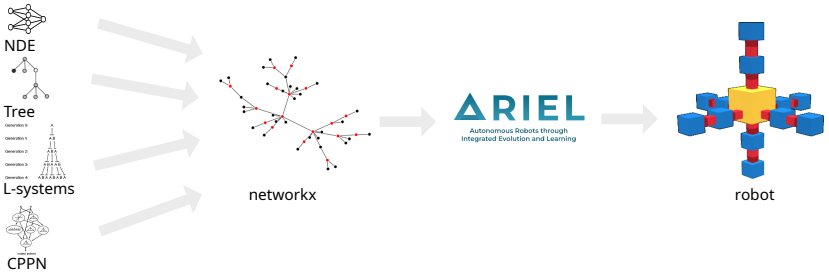


Fig. 1: From the genotype representation to the robot in ARIEL

be used as the baseline to which we will compare the other approaches to determine which one performs the best and in which conditions, we have thus not made any modification to this genotype representation and its associated EC operators and only used the existing ARIEL code. This approach is based on using a Neural Network randomly initialized and submitting a genome representation in a form of 3 arrays of floats to determine which element should be placed, how the elements should be connected between each other and what rotation should be applied to each element. The output of the neural network are arrays representing the probability of each option, which are then passed to a high probability decoding algorithm to produce the robot phenotype.

Initialization the robot genotype is composed of 3 arrays of floats. The number of modules to be created is also provided as a hyperparameter.

Mutation the mutation operator is accomplished by impacting randomly the floats in the genotype by a small variation.

Crossover the crossover implemented for the ARIEL NDE is based on a single-point crossover.

3.3 Tree representation

The tree genotype is different from other genotypes described in this paper as it directly encodes the morphology of the robot, where the other approaches are using an indirect encoding. A tree is made of nodes, each one of them being a module (either CORE, BLOCK or HINGE). In order to build a robot, the tree is parsed using the BFS (Breadth-First Search) or DFS (Depth-First Search) approaches, set by default to the BFS algorithm.

Initialization the robot is composed by default of a CORE module and a given number of nodes with a predefined depth, set by default to 2.

Mutation the mutation algorithm for the tree representation is based on the approach described by Koza and Poli [6], starting with an uniform selection of a random sub-tree, which is then replaced by a newly randomly created sub-tree with a given maximum depth and branching probability.

Crossover we implemented a variant of the Koza standard tree crossover algorithm (Koza, Genetic Programming, 1992). This version differs in the way we randomly choose sub-trees as follows:

- Select a sub-tree of parent 1 using a high probability for a non-leaf node and a low probability for a uniform selection of any node.
- Select randomly a sub-tree in parent 2
- Use the sub-tree of parent 1 with offspring 2 and the sub-tree of parent 2 with offspring 1
- Keep the other elements of parent 1 for offspring 1 and the other elements of parent 2 for offspring 2

3.4 L-System representation

The L-system implementation follows the exact setup defined in [9], using the turtle approach to build the phenotype of the robot based on the axiom and rules defined. If an incorrect instruction is given (which can happen after mutation and crossover), it is simply ignored by the builder, like moving to an element which does not exist, or adding a new element on a position where there is already an element. The elements are always connected from their back face.

Initialization the random initialization of L-System based individuals is composed of the following elements : one axiom string "C" and for each of the potential elements (C, H, B and N) a rule composed of the element itself and n randomly selected tokens, where n is a random number between 2 and an hyperparameter. Possible elements are *C*ore, *H*inge, *Brick* and *N*one.

Mutation the mutation of the L-System is made by adding or removing a token in the string part of the rule. When removing an element, the rules cannot contain less than the identical rule. For example the rule for H is at minimum H = H. The behavior of the mutation can be tweaked by a hyper-parameter indicating if the mutation should focus more on adding or removing elements.

Crossover for combining L-Systems and creating offspring, we have chosen an approach slightly different from the one implemented by Miras et al. [9] as the cross over is conducted at the token level and not the rule level. We have created a uniform crossover over the tokens in each rule, by randomly assigning in a given rule each token of each parent to one or the other offspring.

3.5 CPPN representation

The CPPN representation is based on a network of functions. To move from the genotype (network of functions) to the phenotype, a decoder function is in charge of iteratively building the phenotype and works as described in the work from Auerbach et al. [1].

Initialization For each individual, the corresponding genotype of CPPN function is created as follow:

- the input layer of the network is created with the nodes needed to represent the parent and child coordinates
- the output layer is created with two nodes representing the type of module and the associated rotation
- the input nodes and the output nodes are fully connected
- each connection is assigned a random weight
- a random activation function is assigned to each output node

Mutation For the CPPN approach, the mutation is created by using a complexifying function which can randomly apply one of the following functions: add a node, add a connection, mutate a weight, remove a connection or remove a node. The impact of the functions is defined as presented by Stanley and Miikkulainen [11].

Crossover The crossover function for the CPPN representation is accomplished by matching up genomes for different network topologies using innovation numbers [11].

3.6 Descriptors

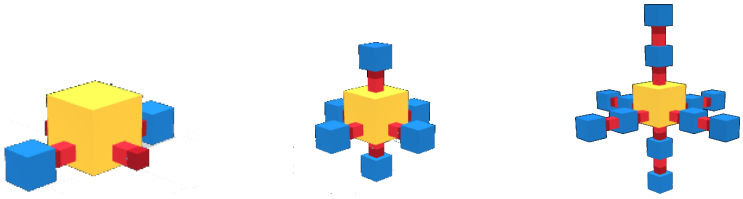
In order to assess the characteristics of robots we have implemented the morphological descriptors from [9] to account for the 3D nature of ARIEL robots. These make it possible to represent any robot in $[0, 1]^6$.

Furthermore, we also used a complementary way to define the proximity of two robot phenotypes by calculating the Tree Edit Distance between the trees. The TED calculation counts the number of actions needed to move from a tree A to a tree B, which gives a strong indicator on how similar two trees are. The lower the number of actions, the more similar the trees are. For this paper we used the TED as described by Zhang and Shasha [14]. In our specific work, the trees represent the phenotype of the robots, which are obtained by directly converting the networkx graph to a tree representation. The TED calculation is made on the networkx object using the zss library [12].

3.7 Experiments to compare the genotype representations

The following experiments have been designed to highlight the impact of the genotype representations on different tasks. In all cases, population size is 100 and evolutions were left to run for 200 generations.

Copying a robot phenotype through evolution The goal of this experiment is to determine if there is a variation in the performance of the same EC algorithm on a simple task to copy a target phenotype according to the genotype representation used. The fitness function is defined as the similarity of the morphological



(a) Simple (b) Medium (c) Complex

Fig. 2: Target morphologies the evolution is asked to copy, from the simplest one to the most complex one

descriptors between the model to copy and the individual produced. The same experiment is also run using the TED calculation as fitness function. The goal of the evolution is to maximize the likeliness between the morphological descriptor / TED of the individuals produced and the morphological descriptor / TED of the model. The morphology to copy is the one labeled as Complex in Figure 2c.

Copying a changing robot phenotype through evolution In this experiment, the goal is similar to the previous experiments, but in addition we periodically change the model to be copied in order to highlight the adaptability of each genotype representation. The fitness function is similar to the one used in the previous experiment, except that the model used to compare the morphological likeliness is modified each time the model changes. The morphologies to copy are the ones presented in Figure 2.

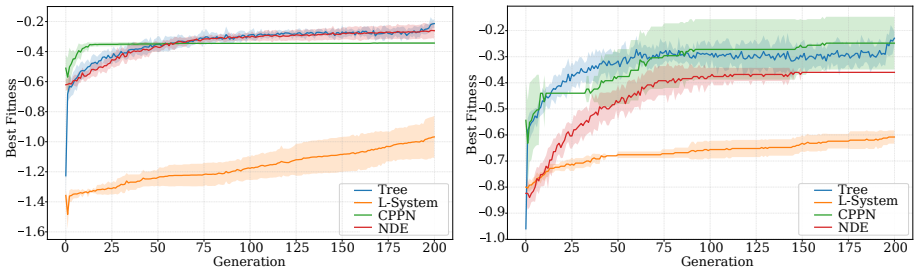
Keeping diversity of population through evolution In this experiment, conducted at the same time than the one copying a robot phenotype through evolution, we compute the average TED of each individual in the population to the 10 other population members with which it is the closest. The goal is to see how long each genotype representation keeps a diversity in the population before converging to more uniformity when reaching the goal.

Generate the most diverse population through evolution In this experiment, we want evolution to produce the most diverse individuals possible, in order to assess the verbosity of each genotype descriptor. The fitness function is defined as the average distance between each individual and the 10 other individuals in the population to which it is the closest. The goal is to maximize this distance.

Determine if the representations embed biases In this experiment, we run an evolution algorithm where the fitness of each individual is always 0. This means that the mutation and the crossover are performed fully randomly and no specific behavior is rewarded for the individuals. Looking at the average of the 6D descriptors of the individuals created by the evolution will highlight for each representation if some patterns of bias emerge.

4 Results

4.1 Copying a robot phenotype through evolution



(a) Morphological descriptors (b) Tree Edit Distance

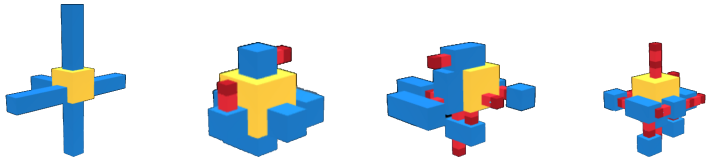
Fig. 3: Performance over 10 experiments for the "Copy" task with 4 genotype representations (Tree, L-System, CPPN & NDE) and two distance metrics (6D euclidian distance and Tree Edit Distance).

Table 1: Fitness for each genotype representation (6D descriptors and TED)

Genotype	Runs	6D Distance	Tree-edit Distance
		Mean ± Std	Mean ± Std
Tree	10	-0.191 ± 0.029	-0.160 ± 0.042
L-System	10	-0.965 ± 0.132	-0.608 ± 0.024
CPPN	10	-0.343 ± 0.004	-0.248 ± 0.096
NDE	10	-0.258 ± 0.049	-0.360 ± 0.000

When using 6D morphological descriptors (Figure 3a and Table 1) to provide an euclidian distance metric in the "Evolve to Copy" task, Tree and NDE representations achieve the best final fitness. The CPPN representation is converging the fastest, but also plateauing earlier. L-System performs drastically worse, but keeps on improving, suggesting its generative grammar structure is slower to adapt to the assigned copy task. In the same setup, but using TED as fitness function (Figure 3b and Table 1), the CPPN and Tree representations achieve the best final fitness, with CPPN converging fastest in first 25 generations. NDE plateaus mid-range. L-System consistently under-performs the other representations, though it still shows slow improvement at generation 200, meaning that it can be only slower to converge.

Overall, the results indicate that genotype representation choice has a strong impact on the ability of the evolution algorithm to achieve this specific task. The



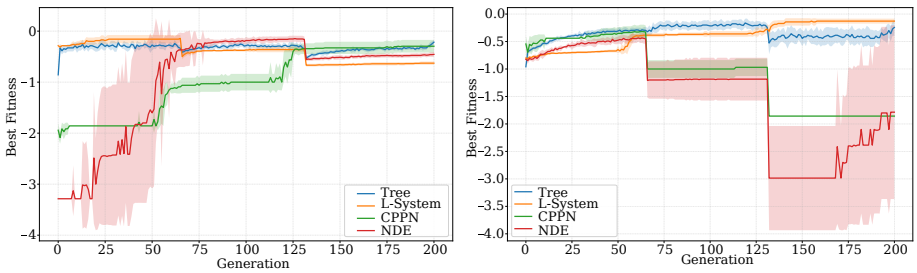
(a) CPPN (b) L-systems (c) NDE (d) Tree

Fig. 4: Example of robots generated by each genotype representation in the evolve-to-copy experiment with the TED as fitness function.

Tree approach shows the best performance by leading both approaches, followed by CPPN and NDE. Examples of the robots generated by each representation using the TED fitness are presented in figure 4.

As demonstrated in this task, the 6D descriptors perform, overall, worse than the Tree Edit Distance. For this reason, the next experiments have been conducted using the TED approach only.

4.2 Sequential copying



(a) Copy sequence (b) Copy sequence reversed

Fig. 5: Fitness comparison over 10 experiments with 100 individuals population trying for copy a different robot phenotypes through evolution with the 3 different genotype representations. Going from simple to complex morphologies on the left and in reverse order on the right.

In this experiment, evolution is asked to copy sequentially 3 different morphologies going from the simplest one to the most complex one. The morphologies are shown in Figure 2.

Based on the results shown in Figure 5a, we can see that the performance differs between the different representations. The L-System works very well on the simple morphology compared to the other representations, but then looses its performance when the representations become more complex. The CPPN and

the Tree representations are showing the best adaptability to changing conditions, even if CPPN is showing a consistent fitness increase when Tree sees a drop in performance during each target change. The NDE is showing a drop of performance adapting to the more complex change, even if the shape of the slope is showing that it is slowly catching up its previous fitness score.

Table 2: Best fitness achieved for each genotype when copying multiple target robots in increasing and decreasing complexity

Genotype	Runs	6D Distance Mean ± Std	Tree-edit Distance Mean ± Std
Tree	10	-0.127 ± 0.043	-0.067 ± 0.051
L-System	10	-0.143 ± 0.064	-0.129 ± 0.043
CPPN	10	-0.296 ± 0.118	-0.320 ± 0.120
NDE	10	-0.139 ± 0.046	-0.334 ± 0.116

Overall, the results indicate that genotype representation choice has, once more, a strong impact on the ability of the evolution algorithm to achieve this specific task and how quick the algorithms can adapt to changing goals. The Tree and the CPPN approaches show the best performance by leading this experiment results, followed by the NDE representation.

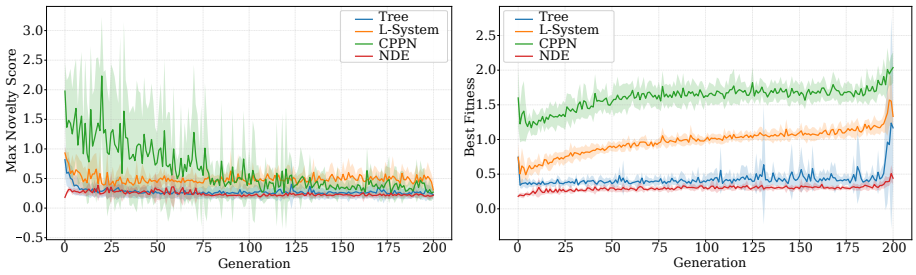
4.3 Reverse sequential copying

In this experiment, evolution is asked to copy sequentially 3 different morphologies, identical from the one described in the previous experiment (Figure 2), but this time going from the most complex one to the simplest one. The L-system performs well (Figure 5b and Table 2), but mostly because of the context of the experiment, its very slow adaptation and the decreasing complexity of the target gives it an advantage over the other representations. It was not able to reach the same level of performance while copying the initial more complex target, so when the target simplifies, its fitness improves without showing a lot of real adaptation, which is highlighted by the improvement being centered around the steps on which the target is changed and staying quite flat on all other generations. The CPPN and NDE representations seem to only lose performance on each step. For CPPN it seems the representation is not able to scale down to simpler morphologies and stops learning after the more complex (and first) target. The NDE representation, even if it is impacted by a drop of performance at each step is able to improve its performance at the end of the last target. The Tree representation is the real winner though, by showing a constant progression after a normal drop at each goal change. Overall, the Tree approach shows the best performance in this setting, followed by the NDE representation, albeit with a significant drop of performance. L-Systems even if showing good results

on the chart cannot be considered as the winner as it benefits from a side effect and overall is not showing great adaptability.

4.4 Spontaneous diversity

As described in Figure 6a, the capacity of each representation to keep a diverse population through evolution to reach a goal greatly varies. CPPNs generate the highest population diversity in the early generations. This diversity then decays rapidly and converges toward the level of other representations around generation 100. L-Systems maintain the most consistent mid-level diversity throughout the whole evolution. The high novelty of population for CPPN is also linked to the fast fitness gain before plateauing observed in the previous experiments, which relates to the relationship existing between diversity and fitness stability, or the tradeoff between exploration and exploitation. Overall the CPPN is the best performing representation for keeping an higher diversity through the evolution process, and L-system is showing the most consistent diversity score during the whole evolution. The results indicate that genotype representation choice has also a strong impact on the observable diversity of the population created during the evolution process.

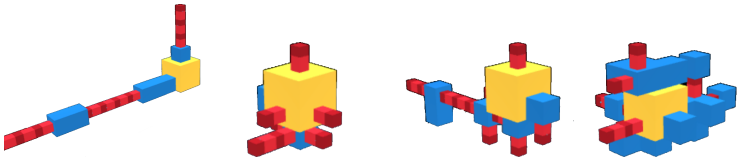


(a) Spontaneous diversity (b) Novelty-driven

Fig. 6: Representation of the diversity of the population at each generation over 10 experiments with 100 individuals. The left is showing how much diversity is kept in the populations when running the evolve to copy task and the right is showing the maximum novelty achieved for new individuals when the fitness function is set to maximize novelty

4.5 Novelty-driven evolution

In this experiment, the fitness to be maximized is computed by looking at how different each individual is from its 10 most similar individuals in the population. On Figure 6b and Table 3, we can see that the results vary significantly between the different genotype representations. The Tree and the NDE representations exhibit overall low performance while L-Systems and CPPNs obtain



(a) CPPN (b) L-systems (c) NDE (d) Tree

Fig. 7: Example of diverse morphologies created by the different representations when driving evolution to maximize novelty.

higher averages, throughout the whole evolutionary process, as well as a mostly positive derivative. Examples of the phenotypes created by this evolution can be seen in Figure 7. CPPN out-performs the other representations in this experiment by consistently showing the highest novelty scores, confirming its capacity to generate very diverse morphologies. Tree and NDE produce the worse novelty, plateauing early, which is an indication that their representation naturally converge to similar morphologies. In general, we can say that the neural encoding of CPPN seems best suited for open-ended morphological exploration. Tree and NDE low novelty scores explain their faster fitness convergence in the other tasks, by exploiting more than exploring the search space.

4.6 Identifying biases

Based on the figure Figure 8, we can clearly see that each representation has some internal biases. The CPPN representation creates individuals which tend to be symmetrical (similar construction on different sides of the core object). It is the representation with the highest bias and is very unbalanced regarding the 6D descriptors. The L-systems representation produces individuals balanced in term of numbers of joints and limbs, with also a tendency to be proportionate. It is the second representation showing the most biases regarding the 6D descriptors. The Tree representation creates individuals which are highly proportionate and have a tendency to display some symmetries with a balanced used of limbs and joints. The NDE representation is the representation showing the less biases compared to the other representations by creating individuals which are quite balanced across all the 6D descriptors. These results show that the choice of a

Table 3: Maximum average novelty achieved for each genotype at the last generation

Genotype	Runs	Mean ± Std
Tree	10	1.992 ± 1.365
L-System	10	1.712 ± 0.263
CPPN	10	2.269 ± 0.073
NDE	10	0.596 ± 0.091

(a) CPPN (b) L-systems

(c) NDE (d) Tree

Fig. 8: Radar view of the average 6D descriptors of individuals produced when running evolution with fitness set to 0 for each representation

given representations for the genotype of the individuals have a big impact on tasks the evolution computing algorithm will have to accomplish with them.

4.7 Tasks performance comparison

The Table 4 is showing the ranking of each genotype representation accross the different experiments conducted.

5 Discussion / conclusion

In this paper, we have looked at the impact of four genotype representations (Tree, L-System, CPPN, and NDE) on the performance of Evolutionary Robotics algorithms across multiple tasks. The results clearly show that the genotype representation has a direct impact on the ability of evolutionary algorithm to achieve a given task and to generate or keep a diverse population. Each representation is also embedding some biases which has a direct impact on the individuals created using it and the NDE is the most balanced representation. The Tree representation is the most robust representation, achieving strong results on all copy tasks, including when the morphology is changing. The CPPN representation is

Table 4: Performance ranking by task

Genotype	Tree	L-systems	CPPN	NDE
Copy through evolution	**1**	3	**1**	2
Sequential copying	**1**	3	**1**	2
Reverse sequential copying	**1**	3*	3	2
Spontaneous diversity	3	2	**1**	3
Novelty driven evolution	3	2	**1**	4
Identifying biases	2	3	4	**1**

** L-systems benefits of a side-effect, but does not perform well*

also performing quite well on most of the copy tasks and out-performs the other representations on generating diverse populations. Its neural encoding promotes exploration which explains why it converge faster than the other representations, but at the same time this limits its long-term exploitation capacity, potentially hindering its long term performance. The L-system is also a good choice for diversity of population, but suffers from a very slow convergence compared to the other representations. The NDE representation is showing an acceptable performance in simple copy tasks, but suffer from instability when the target is changing. It is also not performing well in term of diversity of the population. We can clearly say that there is not one best genotype representation for all Evolutionary Computing tasks, even if Tree and CPPN have shown an overall better performance than others in our experimental setup. This highlights the fact that it is really critical to make a wise choice of genotype representation when designing an Evolutionary Computing system. Tree is recommended when the most important aspects are stability and adaptability, CPPN when diversity and exploration are needed, L-system when a consistent diversity is needed through evolution and NDE when representation bias is a problem. Future research should be focused on trying to find a way to combine the strength of the Tree and the CPPN representations, as well as investigating what impact the addition of locomotive tasks has on the results.

References

1. Auerbach, J.E., Bongard, J.C.: Evolving complete robots with cppn-neat: the utility of recurrent connections. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. p. 1475–1482. GECCO '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2001576.2001775
2. Bie, D., Zhu, Y., Wang, X., Zhang, Y., Zhao, J.: L-systems driven self-reconfiguration of modular robots. International Journal of Advanced Robotic Systems **13**(5), 1729881416669349 (2016). https://doi.org/10.1177/1729881416669349
3. Di Matteo, J.M., Weissl, O., Eiben, A.: Fertility during learning in evolutionary robot systems. In: Proceedings of the Genetic and Evolutionary Computation Con-

ference. p. 131–139. GECCO '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3712256.3726382
4. Di Matteo, J., Grigoriadis, I.: Ariel (2025), https://github.com/ci-group/ariel
5. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference. pp. 11 – 15. Pasadena, CA USA (2008)
6. Koza, J., Poli, R.: Genetic Programming, pp. 127–164 (01 2005). https://doi.org/10.1007/0-387-28356-0_5
7. Kurth, W.: Specification of morphological models with l-systems and relational growth grammars. IMAGE. Zeitschrift für interdisziplinäre Bildwissenschaft **3**(1), 50–79 (2007). https://doi.org/http://dx.doi.org/10.25969/mediarep/1674
8. Luo, J., Zeeuwe, D., Eiben, A.E.: Gait-learning with morphologically evolving robots generated by l-system. In: 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET. pp. 1–9 (2024). https://doi.org/10.1109/ICECET61485.2024.10698145
9. Miras, K., Haasdijk, E., Glette, K., Eiben, A.E.: Search Space Analysis of Evolvable Robot Morphologies. In: Sim, K., Kaufmann, P. (eds.) Applications of Evolutionary Computation. pp. 703–718. Springer International Publishing, Cham (2018)
10. Singh, R.: Evolutionary ai for morphological and locomotor design of resilient exploration robots (Oct 2025). https://doi.org/10.36227/techrxiv.176116522.21490062/v1
11. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic Programming and Evolvable Machines **8**, 131–162 (2007), https://api.semanticscholar.org/CorpusID:2535195
12. Tim Henderson, Stephen Johnson, G.S., Fonseca, E.R.: zss library (zhang-shasha implementation) (2020), https://github.com/timtadh/zhang-shasha
13. Veenstra, F., Hart, E., Buchanan, E., Li, W., De Carlo, M., Eiben, A.E.: Comparing encodings for performance and phenotypic exploration in evolving modular robots. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 127–128. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3322054
14. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. **18**, 1245–1262 (12 1989). https://doi.org/10.1137/0218082