On Pros and Cons of Evolving Topologies with Novelty Search

Léni K. Le Goff[1], Emma Hart[1], Alexandre Coninx[2] and Stéphane Doncieux[2]

[1]Edinburgh Napier University, Scotland, UK
[2]Sorbonne Universite - ISIR/CNRS, Paris, France
leni_legoff@protonmail.com

Abstract

Novelty search was proposed as a means of circumventing deception and providing selective pressure towards novel behaviours to provide a path towards open-ended evolution. Initial implementations relied on neuro-evolution approaches which increased network complexity over time. However, although many studies have reported impressive results, it is still not clear whether the benefits of evolving topologies are outweighed by the overall complexity of the approach. Given that novelty search can also be combined with evolutionary methods that utilise fixed topologies, we undertake a systematic comparison of evolving topologies, using two types of fixed topology networks in conjunction with novelty search on two test-beds. We show that evolving topologies do not systematically help, and discuss the practical consequences of these results and the research perspectives opened up.

Introduction

The seminal work of Lehman and Stanley on Novelty Search (NS) (Lehman and Stanley, 2011) has created a significant shift of perspective within neuro-evolution (Stanley and Lehman, 2015). Although much work had been dedicated to understanding the influence of encodings before its advent (Stanley and Miikkulainen, 2002), the results obtained by NS suggest that selective pressures are of utmost importance to get the best from a particular encoding, leading to many works in which understanding the role of selection-pressure became the main focus (Doncieux and Mouret, 2014; Lehman and Miikkulainen, 2015). One of the most thought provoking experiments showed that a powerful encoding associated with a naïve goal-based selective pressure can get stuck in local optima, failing to consistently reproduce results they had found with other means (Woolley and Stanley, 2011). However, used judiciously, architectures automatically discovered by evolutionary approaches have proven to be competitive with hand-made architectures, even in deep neural networks (Miikkulainen et al., 2019), showing the potential of the method. But even now, besides observing that neuro-evolution methods that enable augmenting of topologies (e.g. NEAT (Stanley and Miikkulainen, 2002)) may work — and sometimes works extremely well — little is known about the inner workings of evolution of neural architectures, even in simple setups. When the focus is on open-ended evolution, its use seems to be a requirement (at least, this is suggested by its biological counterpart (Martin, 1999)), but what about other more general applications of evolution? What are the pros and cons of these approaches? Do the pros outweigh the cons and does NS really benefit from it? Although NS was first published more than ten years ago we still do not have a clear answer to these questions.

One of the initial motivations to add NS on top of NEAT was the idea of starting the exploration from simple behaviours and then providing the possibility for the algorithm to make their complexity grow (Lehman and Stanley, 2011). This could be a *bootstrap facilitator* (i.e. enabling the algorithm to get on track within the early iterations): too complex network architectures with random parameters could get stuck in degenerate behaviours that the evolutionary search has trouble escaping from. Starting from simple topologies that can be progressively complexified could avoid this.

A second potential advantage is to automatically find the most fit network structure without the need to handcraft it. Although it may seem appealing at first glance, this motivation needs to be justified. Neural networks have many standard structures that are known to fit different categories of problems: multi-layer perceptrons to approximate functions and Elman networks or fully recurrent networks when past values need to be taken into account, for instance. When facing a particular problem, knowing the kind of network structure that can solve it, or at least making a guess, is not a big issue. In very challenging applications, finding the right structure may make a difference, and in this case, network architecture search is clearly justified (Real et al., 2019). It is less clear in general if an architecture search algorithm can outweigh what a specialist can intuitively find within a very short time. In an open-ended evolution scenario, there may be no expert in the loop to find the structure for a particular problem that the system has to deal with, but if there is one, the question surely has to be raised.

Encodings with evolving topologies have a cost. There is no free lunch (Wolpert and Macready, 1997). The initial version of NS relies on NEAT (Stanley and Miikkulainen, 2002). Besides the network encoding per se, NEAT includes specific crossover and selection algorithms to respectively deal with the competing convention problem and to protect innovation. For this reason, it cannot be used with standard evolutionary algorithms, with the result that it cannot benefit from the the latest and most efficient evolutionary algorithms. It has been shown that some features of NEAT can be discarded to make it compatible with standard selection algorithms like NSGA-II (Mouret and Doncieux, 2012), which permits multiobjectivization (Doncieux and Mouret, 2014) in which novelty can combined with other objectives (Mouret, 2011), and can even result in increased performance (Mouret and Doncieux, 2012). However, an encoding with an unbounded augmenting topology capability *a priori* requires a genotype whose size can increase. This is a strong constraint that prevents the use of efficient algorithms with fixed size genotypes like CMA-ES (Hansen and Ostermeier, 2001).

Furthermore, the capacity to evolve topologies introduces new mecanisms to design and tune: a natural question is to understand whether they depend on the task or not. On tasks requiring a deep network, for instance, a classical mutation will have a too large impact on the network behaviour, suggesting to develop adapted mutation mechanisms (Lehman et al., 2018). If such adaptations are task specific, then the removal of the need to have task specific expertise to design the architecture will just be replaced by a requirement for a task specific expertise related to mechanisms design and tuning. The gain in this case is clearly not straightforward.

To sum up this work considers the following question: *does the complexification of an evolving topology encoding bring sufficient added value to novelty search?* We focus on two potential advantages for evolving topologies regarding novelty search: (1) bootstrap facilitation and (2) increasing the diversity of behaviours, while also recognising that there are two potential drawbacks: (1) the constraints on the genotype and (2) the potential difficulty to tune parameters. We use two domains (maze-navigation and a walking robot task) to systematically explore whether we observe the suggested advantages in practice, and whether or not they are significant enough to balance the constraints such a genotype imposes. As previous studies suggest that NS is not very sensitive to the encoding parameters (Gomes et al., 2015), we have chosen not to focus on this aspect here and to postpone further studies on this topic to future work.

Related Work

Recent works have shown that NS and other divergent algorithms can efficiently drive the optimisation of deep-neural networks that have a fixed structure (Conti et al., 2018; Gajewski et al., 2019), suggesting in a very challenging context, that evolving topologies may not be required in NS. Hence, if it is apparent that NS can efficiently work with a fixed structure, it is not clear whether the process would have bootstrapped faster using an encoding that enabled augmenting topologies, or if a more appropriate structure would have been found by an encoding that allows architecture search — in particular as architecture search has also shown its potential in challenging application domains (Real et al., 2019)).

It should be noted that NEAT was directly compared to fixed topologies when it was first introduced (Stanley and Miikkulainen, 2002). An ablation study was conducted to investigate the importance of the different elements of this approach. This comparison thus relied on the selection algorithm of NEAT, and it is well known that the selection pressure has a large impact on the performance of neuroevolution (Doncieux and Mouret, 2014; Lehman and Miikkulainen, 2015). Furthermore, although NS can be considered as a standard evolutionary algorithm in which the goal-oriented fitness function is simply replaced by a novelty criterion, in fact it has been shown to actually behave in a completely different manner, namely as a uniform sampling process in the behavioural space (Doncieux et al., 2019). This behaviour makes it difficult to transpose any results or design guidelines available with other evolutionary algorithms to NS. Hence, there is a need to study the impact of the encoding directly on NS.

The relationship between an encoding enabling architecture search and divergent algorithms has not been frequently explored in the literature. Mouret and Doncieux (2012) conducted an empirical study that included experiments using a direct but simple encoding enabling evolution of weights and topology loosely inspired by NEAT and a fixed Elman network. The comparison of these setups was not their goal, but it can be seen from their results that the evolving topologies setup has no clear advantage, at least not in every case. Likewise, Tarapore et al. (2016) have studied the impact of the encoding on the performance of MAP-Elites, another divergent search algorithm: their conclusion was more in favour of the fixed encoding they used. These results suggest that evolving structures with divergent search is not always beneficial, but these studies were not performed on NS and did not examine the dynamics of the search.

Gomes et al. (2015) performed one of the rare systematic studies of NS features. They compared a NS variant implemented with a genetic algorithm and a fixed Elman structure with a standard NS-NEAT. They examined many factors, including the influence of the mutation rate, concluding that, with an appropriate mutation rate, there was no significant difference between a fixed encoding, and NEAT. However, they only studied one network structure, only experimented with maze tasks, and did not look at the dynamics of the search process.

Background

Novelty Search

NS was introduced by Lehman and Stanley (2011). It is an evolutionary algorithm in which the goal-oriented fitness objective is replaced by a measure of the novelty of individuals. NS is particularly efficient when the rewards are deceptive or sparse, as it pushes the search towards exploration. In contrast to other evolutionary algorithms, individuals are not selected on their performance or fitness but with regard to their novelty with respect to previously evaluated individuals.

To be able to assess the novelty of an individual, we have to define a measurable space to compare individuals' behaviours. This is commonly called the *behavioural space* B. It is a smaller space than the full state space in which action and state trajectories of the agent can be fully described. Each individual has a behavioural descriptor defined in B. Generally, this descriptor is computed based on a trajectory[1] σ of the individual defined on the joint space of the robot states (S) and time (T) : S^T. Thus, we can define $F_B : S^T \rightarrow B$ a function that takes a trajectory as input and computes the corresponding behavioural descriptor.

The novelty of an individual is computed with respect to the current population and an archive. The archive contains the behavioural descriptors (BD) of a sample of past individuals. The archive is filled throughout the generations. Then, the novelty of an individual is defined as the average of the distances between the BD of the individual and its K nearest neighbours in the archive and the population (equation 1).

$$\eta(F_B(\sigma)) = \frac{1}{K}\sum_{i=0}^{K} d_B(F_B(\sigma), F_B(\sigma_i)) \qquad (1)$$

Where d_B is a distance defined on the behavioural space B; σ is the trajectory of the current individual and σ_i the i-th closest individual in the archive and in the population.

Evolving Topologies

In the context of robotics, neural networks are often evolved as controllers to solve a specific task. A naïve approach would be to select a fixed topology and only optimising the weights. However, the topology of the network may be important in relation to the context and the task. In such cases, allowing evolution to discover the best structure seems a reasonable choice. Moreover, an approach that generates topologies should encourage the exploration process, thus releasing the full potential of NS (Lehman and Stanley, 2011). This has motivated the use of NEAT (Stanley and Miikkulainen, 2002). Alongside the weights, NEAT evolves network topologies of increasing complexity. It uses a diversity preservation mechanism based on speciation and a global innovation number for the crossover operator. However, these interesting and useful features prevent this encoding from being used with classical evolutionary algorithm such as NSGA-II as previously noted.

A simplification of this algorithm that we will refer to as Direct-encoding Evolving Topologies (DET) has been proposed by Mouret and Doncieux (Mouret and Doncieux, 2012). They use only mutation and no crossover. To evolve the structure and the weights, five mutation operators are used (selected at random): (1) Add a connection between two existing neurons (add_conn), (2) Remove an existing connection (rm_conn), (3) Add a neuron by splitting an existing connection in two. The weight is kept for the new connections (add_neu), (4) Remove a neuron and its connections (rm_neu), (5) Change the weights by polynomial mutation (ch_conn).

add_conn	rm_conn	add_neu	rm_neu	ch_conn
0.1	0.01	0.1	0.01	0.1

Table 1: Hyper-parameters of DET corresponding to the probability to apply these mutations. The values are those used in the study of Mouret and Doncieux (Mouret and Doncieux, 2012). Their are used in the experiments of this study too.

Assuming the parameters are chosen judiciously, neuro-evolution with DET therefore also generates augmenting topologies over generations (table 1). In this study, DET is chosen to investigate the benefits of augmenting topologies instead of NEAT because of its simplicity: it does not contain a speciation mechanism (which is unrelated to augmenting the topology) and therefore enables the study the effect of augmenting topologies mechanism alone on novelty search.

Experimental Protocol

We consider a neuro-evolution algorithm that is driven by the novelty only and assess the added value of an evolving topology encoding for novelty by comparing it with a fixed topology encoding. The goal of this evolutionary algorithm is to uniformly explore the behavioural space B (Doncieux et al., 2019). The question we seek to answer is: *What is the impact of evolving topology on the uniformity of exploration of B?* Fixed structure and evolved structure neural networks are compared based on their capacity to achieve uniform exploration. Two criteria are considered : (1) the uniformity of the exploration and (2) the speed of the bootstrap, i.e. the number of generation needed to converge.

Fixed Neural Network Structures

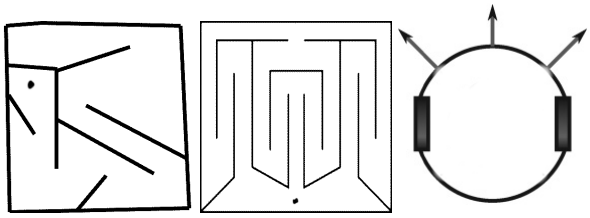
DET is compared to two types of fixed neural networks : a *fully connected feed-forward network* (FFNN) and an *Elman network* (RNN) which falls in the category of recurrent

[1] A list of states over time in which the agents have visited during an episode

neural networks. The *FFNN* is a three layered network composed of an input, a hidden layer, and an output layer. The input layer is fully connected with the hidden layer; the hidden layer is fully connected with the output layer. The Elman network is an extension of the *FFNN*. In addition to the other fully connected layers, a fourth layer, called context units, is fully connected with the hidden layer. Additionally, each context unit is connected to itself. For both types of network, we conduct experiments in which networks of increasing complexity (number of neurons and connections) are used.

Tasks

Experiments are conducted on two different environments. The first one is a simple 2D environment with a simulator based on Simple DirectMedia Layer (SDL)[2] with no physics engine. The second one is a 3D environment simulated with DART (Lee et al., 2018), a modern simulator. In both environments, the goal for the robot is to move away from its starting position. Thanks to novelty search, the end goal is to have a population of robotic controllers in which the end position of the robots uniformly covers the whole environment.



(a) Hard maze (b) Multi maze (c) Wheeled robot

Figure 1: Picture of the two mazes and of the robot used for the mazes' experiments. The black dot indicates the starting position of the robot. The three arrows indicate the position of the proximity sensors and the two rectangles represent the wheels.

2D Mazes The first test case is a simple navigation task in a 2D maze. A wheeled robot has to move in the maze. It is equipped with two wheels and three range sensors on the front (see 1c). Two mazes with different features are used (see figure 1). The *hard maze* has wide corridors but a non-repetitive structure. It is named *hard maze* as in the study of Lehman and Stanley (2011). The *multi maze* has narrow corridors but a repetitive structure. It is named a *multi maze* as in the study of Gomes et al. (2015).

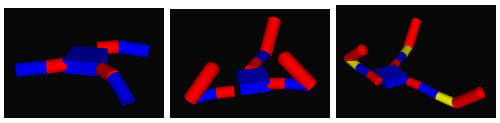
The behavioural descriptor represents the final position of the robot. The neural networks used to control movement have three inputs, two outputs, and one input bias. The number of neurons and connections of each fixed structure neural networks are listed in table 2. The evaluation of each controller is simulated using fastsim (Mouret and Doncieux, 2012), a simple 2D simulator based on SDL with no physics engine.

[2]https://www.libsdl.org/index.php

Name	Type	Hidden	Context	Neurons	Connections
ff_4	FFNN	4	0	10	24
rec_4	RNN	2	2	10	22
ff_16	FFNN	16	0	24	96
rec_16	RNN	8	8	24	120
rec_32	RNN	16	16	38	368
rec_128	RNN	64	64	134	4544

Table 2: Description of the fixed structure neural networks used for the 2D mazes experiments.

3D Walkers For this task, a three legged robot has to move in a square arena. Its starting position is in the centre of the arena. The experiments are conducted with three different robots which have either 2, 3, 4 degrees of freedom (DOF) on each leg (see figure 2). The behavioural descriptor, as in the maze experiment, is the final position of the robot. The position is calculated from the centre of the base on which the three legs are attached. The neural networks used to control the robot have as input all the joint angles of the legs and the position and orientation of the base, and as output the joint angles of the legs. Each neural network has in addition an input bias. The number of neurons and connections are listed in table 3. For this task, experiments conducted with feed-forward neural networks performed poorly and thus are of little interest. So, we chose not to include them. The evaluations are simulated using the DART simulator (Lee et al., 2018) with its provided physics engine and collision solver.



(a) 2 DOF (b) 3 DOF (c) 4 DOF

Figure 2: Picture of the three legged robots. Each degree-of-freedom (DOF) has alternating colours in order to be distinguishable.

Name	Type	Hidden	Context	Neurons	Connections
rec_32	RNN	16	16	51, 57, 63	576, 672, 768
rec_64	RNN	32	32	82, 88, 94	1664, 1856, 2048
rec_128	RNN	64	64	146, 152, 158	5376, 5760, 6144

Table 3: Description of the fixed structure neural networks used for the 3D walkers experiments. In the neurons and connections column, the numbers correspond in the following order to the 2, 3, and 4 DOF for each leg.

Exploration uniformity measure

To measure the exploration uniformity score of a run, we use the method used by Gomes et al. (2015). The environment is divided into $R = 12 \times 12$ cells and the number of individuals in each cell is counted. Then, the distance of this distribution of the population is compared with the ideal uniformly distributed population, measured using the *Jensen-Shannon divergence* (JSD) (Fuglede and Topsoe, 2004). JSD is based on the Kullback-Leibler divergence commonly used to measure the difference between two probability distributions. Hence, the exploration uniformity score is the opposite of the JSD between P_ϕ of the current population and the uniform distribution I (see equation 2). So, a score of 1 corresponds to a perfect uniformity of the distribution of the solutions' BD.

$$U(\phi) = 1 - JSD(P_\phi, I) \tag{2}$$

Where ϕ is the set of behavioural descriptors of a population; P_ϕ the distribution of this set in the behavioural space and I the uniform distribution in the same space.

Results

All the results shown in this section have been conducted with an evolutionary algorithm with tournament selection, no crossover and replacement of the whole population at each generation. The only objective is novelty. Ten replications of each experiment were conducted. Finally, the population is set to a size of 400. The source code used to obtain the results is available at this address : https://github.com/LeniLeGoff/novelty_neat.

2D Mazes

Figures 3 and 4 show the exploration uniformity score over the generations for the two mazes. Experiments with DET reach an exploration score at convergence that is less than or equal to the best runs from RNNs. Indeed, rec_16 (see table 2) has a better score than the DET on the hard maze, and the DET reaches a score equivalent to rec_32 on the multi maze. Moreover, on both mazes, experiments with DET have a slower bootstrap than almost all the runs with a fixed structure network.

However, it is worth noting that DET achieves a high coverage and an exploration capability close to uniformity with simpler network structures than its fixed network challengers. As seen in figure 5, the maximum number of neurons and connections in the DET generated networks are respectively, around 15 and 40. By comparison, rec_16 has 24 neurons and 120 connections and rec_32 has 38 neurons and 368 connections (see table 2). Thus, neuro-evolution with DET generates simple structures which achieve at least same results than more complex recurrent networks.

Interestingly, increasing the complexity of the RNN (i.e. number of neurons and connections) does not necessarily

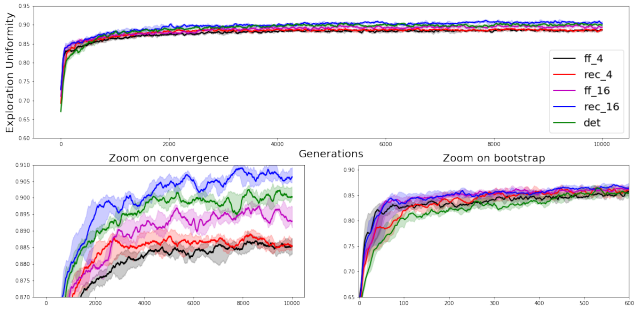


Figure 3: Exploration uniformity score over the generations of experiments conducted on the hard maze. The solid bold line corresponds to the median and the transparent areas above and under the curves correspond to the first and third quartile. Network structure parameters are in Table 2.

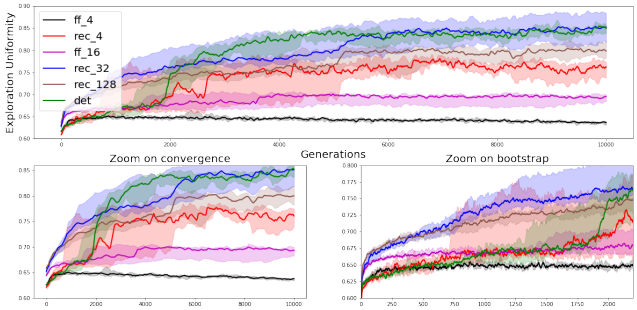
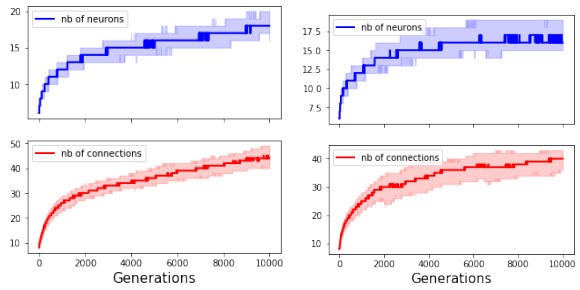


Figure 4: Exploration uniformity score over the generations of experiments conducted on the multi maze. The solid bold line corresponds to the median and the transparent areas above and under the curves correspond to the first and third quartile. Network structure parameters are in Table 2.

achieve a more uniform exploration. Experiments conducted with rec_128 which has 134 neurons and 4544 connections (see table 2), reach an exploration uniformity score below those conducted with rec_32 (see figures 4 and 6). Also, Elman networks lead to a better exploration than feed forward networks for a similar complexity. Experiments with rec_4 reach largely higher scores than ff_4 on the multi maze (see 4). The multi maze seems to present a greater challenge than the hard maze which is likely due to the narrow corridors and the length of the paths.

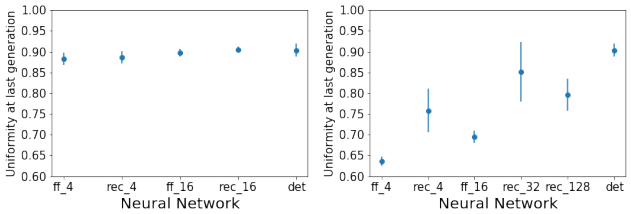
3D Walkers

As shown in figures 7, 8, and 9, with all the robots, DETs does not lead to an exploration score as high as that obtained using RNNs. Unlike on the maze environment, experiments with DET are not systematically slower to bootstrap than with the RNNs, but neither are they faster (see figures 8 and 9). On all the experiments, rec_32 and rec_64 are the best structures for exploration, apart from the 3 DOF legged



(a) Hard maze (b) Multi maze

Figure 5: Number of neurons and connections of the DETs throughout the generations on the hard maze and on the multi maze. For comparison : *ff_4* has 10 neurons and 24 connections, *rec_4* has 10 neurons and 22 connections, *ff_16* has 24 neurons and 96 connections, *rec_16* has 24 neurons and 120 connections, *rec_32* has 38 neurons and 368 connections, and *rec_128* has 134 neurons and 4544 connections. Refer to table 2 for more information about the fixed networks structures.



(a) Hard maze (b) Multi maze

Figure 6: Plot of the exploration uniformity at the last generation for each neural network topology and DET.

robots in which there is no significant difference between the different RNNs. As on the maze environment, the best RNN is not the most complex one. On the experiments with the robots with 2 and 4 DOF on each leg, rec_128 reaches an exploration score below those of rec_32 and rec_64 (see figure 11).

The number of neurons and connections increase with a similar dynamic as the experiments on the maze (see figure 10). This is expected as the hyperparameters of DET are the same in both tasks. Also, as in the maze task, DET generates networks with lower complexity than the RNN. The maximum number of neurons and connections for the robots with 2, 3, and 4 DOF on each leg are, respectively, around 25 neurons and 85 connections; 32 neurons and 150 connections; and 40 neurons and 230 connections. By comparison, *rec_32* has around 50 neurons and between 500 and 800 connections, *rec_64* has around 90 neurons and between 1500 and 2000 connections, and *rec_128* has around 150 neurons and between 5000 and 6000 connections (see table 3).

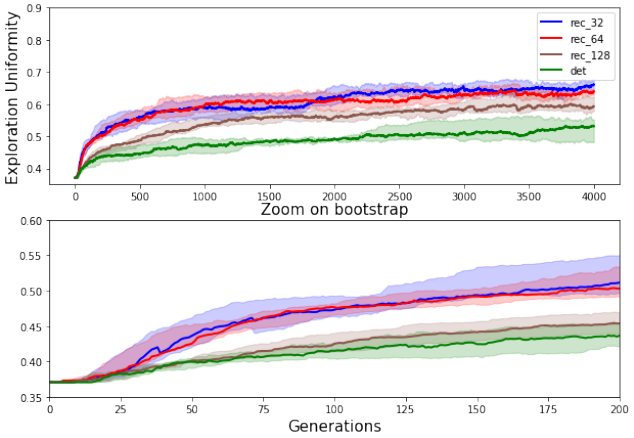


Figure 7: Exploration uniformity score over the generations of experiments conducted with the three legged with 2 DOF on each. The solid bold line corresponds to the median and the transparent areas above and under the curve correspond to the first and third quartile. Refer to the table 3 to have information of the fix networks structures

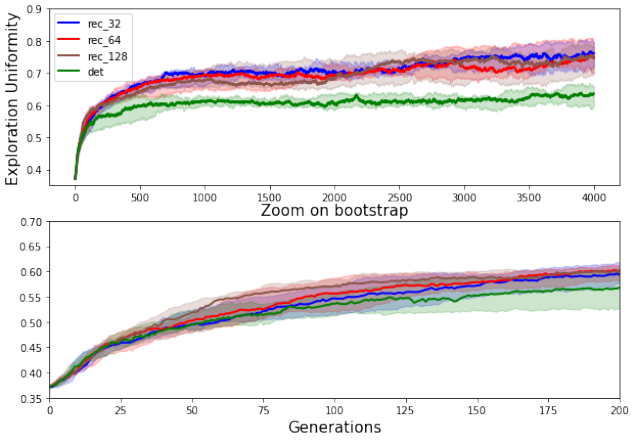


Figure 8: Exploration uniformity score over the generations of experiments conducted with the three legged with 3 DOF on each. The solid bold line corresponds to the median and the transparent areas above and under the curve correspond to the first and third quartile. Refer to the table 3 to have information of the fix networks structures

Overall, the results on this walking task are consistent with the results obtained on the maze task. Evolving topologies neuro-evolution using DET does not systematically lead to a better exploration uniformity than with fixed topology neuro-evolution. Regarding bootstrap, DET does not guarantee a fast bootstrap: in fact, the opposite is observed on the maze task. However, the networks discovered by DET are simpler than the fixed structures that challenge it.

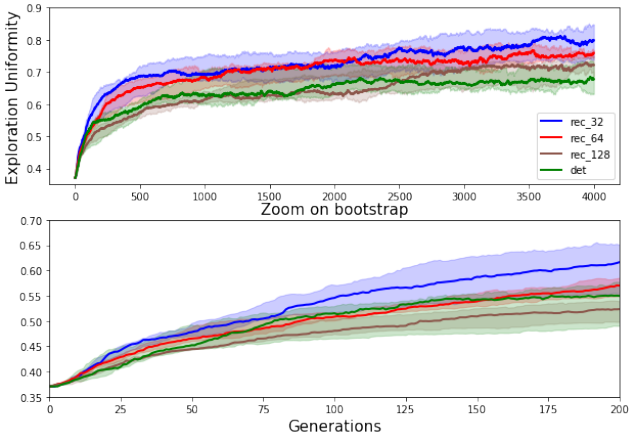
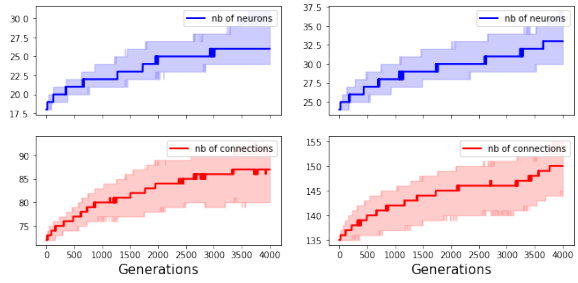
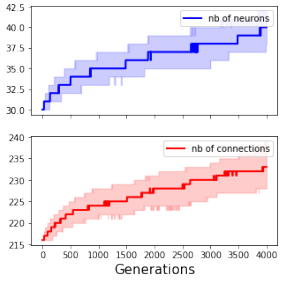


Figure 9: Exploration uniformity score over the generations of experiments conducted with the three legged with 4 DOF on each. The solid bold line corresponds to the median and the transparent areas above and under the curve correspond to the first and third quartile. Refer to the table 3 to have information of the fix networks structures

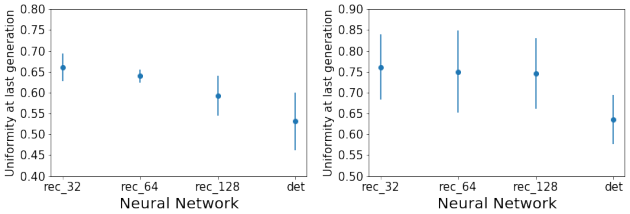


(a) 2 dof legs (b) 3 dof legs

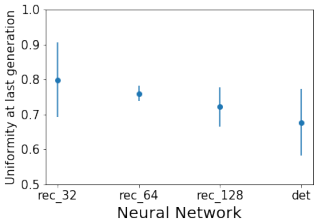


(c) 4 dof legs

Figure 10: Number of neurons and connections of the DETs throughout the generations with the legged robots and on the multi maze. For comparison : respectively for 2, 3, 4 dof legs, *rec_32* has 51, 57, 63 neurons and 576, 672, 768 connections, *rec_64* has 82, 88, 94 neurons and 1664, 1856, 2048 connections, and *rec_128* has 146, 152, 158 neurons and 5376, 5760, 6144 connections. Refer to table 3 for more information about the fix networks



(a) 2 DOF (b) 3 DOF



(c) 4 DOF

Figure 11: Plot of the exploration uniformity at the last generation for each neural network topology and DET. Beware of the y axis scales which are different for each plot.

Discussion

On both tasks studied, the fixed structures chosen for the experiment lead to better exploration. On the maze task, they also lead to a faster bootstrap time than the augmenting topology encoding. The slower bootstrap observed with the DET could be due to the time needed to find appropriate network structures: this is not needed with a fixed structure network encoding. Also, within the available generation budget, neuro-evolution with DET does not generate networks which are as complex as the biggest RNN and FFNN used in this study. This is an interesting feature of augmenting topology neuro-evolution, i.e. that it can achieve equivalent results with less complex networks than with hand-made neural network structures. However, this is also a possible explanation of why this method has lower exploration capability than the most complex of the fixed structure networks used in this study. Indeed, if a task needs a high level of complexity in its policy representation, augmenting topology may need a high number of generations to find this level, or at worst, may never converge to this level of complexity.

Moreover, the results on the maze task show that RNN performs better than FFNN for equivalent complexity, suggesting that network structure has to be chosen according to the task. Indeed, an Elman network (being a recurrent network) has the capacity of computing sequences (Elman, 1990) which is useful for a task such as navigation which has an obvious temporal component. It would be better to use a central pattern generator (CPG) (Shan et al., 2000; Liu et al., 2013) in the walking tasks given that CPG facilitates the emergence of cyclical patterns. In this respect, interesting studies have been conducted to assess the importance of

the network architecture over the values of the weights by randomly assigning values to the weights (He et al., 2016), achieving similar performance to trained networks. It is unclear so far whether augmenting topology neuro-evolution algorithms are capable of generating such structures. However, a recent work proposed a method to evolve weight-agnostic neural networks with the aim of evolving networks able to solve tasks with random weights. This is a significant attempt to shift emphasis towards the structure of networks.

In this study, the impact of the parameter settings for DETs has not been explored. For instance, on the walking task, it would be interesting to tune the parameters in order to accelerate the augmentation of complexity to find larger networks in fewer generations. However, if the parameters of the encoding have to be fine-tuned according to the task domain, then the expertise required to design network structure is simply replaced by the need for task-specific expertise, so there is no significant gain from this point of view.

Experiments have been conducted in different environments and using robots of varying complexity to challenge the complexity of the networks. For the walking task, the same fixed structure leads to the best exploration behaviours, which suggests that it is possible to find one structure with enough generalisation capability to be used on a large variety of environments in the same domain. However, the results also suggest that a critical point is reached in the complexity of the network, after which the quality of exploration decreases with increasing complexity. Furthermore, this critical point seems to be different for each task.

Augmenting topology neuro-evolution is one way to avoid the difficult search for a suitable network. According to the task, the type of structure and its size have to be chosen carefully, which is not a trivial task. However the results of this study show that the use of augmenting topology neuro-evolution is certainly not a panacea. Further research directed towards finding a method that balances the pros and cons of both fixed topology methods and of augmenting topology neuro-evolution is required.

Conclusion

Evolving topologies may have two different impacts: (1) the approach may help to bootstrap the search and (2) it may discover an appropriate structure without the need for an expert to define it. Our experiments show that a simple encoding using an augmenting topologies feature neither helped bootstrap the experiments nor did it converge towards the performance of the most efficient network structure. On the other hand, these experiments converged to structures that are simpler than their fixed structure equivalent. Therefore the conclusion from this study is that, as for now, evolving topology encodings do not outweigh their cost. In line with the principle of Occam's Razor, the evidence presented here suggests that NS should be used with a fixed structure neural network. Nevertheless, architecture search clearly has great potential. Our results show that different network complexities generate different performances and the most complex network is not necessarily the best. It also suggests that the neural networks do not need to start from the simplest structures possible, and that there is no bootstrap problem when starting from complex structures, at least on the tasks we have considered. New encodings which aim to develop methods that rapidly converge towards appropriate structures should take these factors into account.

References

Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K., and Clune, J. (2018). Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems*, pages 5027–5038.

Doncieux, S., Laflaquière, A., and Coninx, A. (2019). Novelty search: a theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 99–106.

Doncieux, S. and Mouret, J.-B. (2014). Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, 7(2):71–93.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Fuglede, B. and Topsoe, F. (2004). Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE.

Gajewski, A., Clune, J., Stanley, K. O., and Lehman, J. (2019). Evolvability es: scalable and direct optimization of evolvability. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 107–115.

Gomes, J., Mariano, P., and Christensen, A. L. (2015). Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 943–950. ACM.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195.

He, K., Wang, Y., and Hopcroft, J. (2016). A powerful generative model using random weights for the deep image representation. In *Advances in Neural Information Processing Systems*, pages 631–639.

Lee, J., Grey, M., Ha, S., Kunz, T., Jain, S., Ye, Y., Srinivasa, S., Stilman, M., and Liu, C. (2018). Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3(22):500.

Lehman, J., Chen, J., Clune, J., and Stanley, K. O. (2018). Safe mutations for deep and recurrent neural networks through output gradients. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 117–124.

Lehman, J. and Miikkulainen, R. (2015). Extinction events can accelerate evolution. *PloS one*, 10(8).

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223.

Liu, C., Wang, D., and Chen, Q. (2013). Central pattern generator inspired control for adaptive walking of biped robots. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5):1206–1215.

Martin, A. P. (1999). Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):111–128.

Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzyan, A., Duffy, N., et al. (2019). Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier.

Mouret, J.-B. (2011). Novelty-based multiobjectivization. In *New horizons in evolutionary robotics*, pages 139–154. Springer.

Mouret, J.-B. and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary computation*, 20(1):91–133.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Shan, J., Junshi, C., and Jiapin, C. (2000). Design of central pattern generator for humanoid robot walking based on multi-objective ga. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, volume 3, pages 1930–1935. IEEE.

Stanley, K. O. and Lehman, J. (2015). *Why greatness cannot be planned: The myth of the objective*. Springer.

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

Tarapore, D., Clune, J., Cully, A., and Mouret, J.-B. (2016). How do different encodings influence the performance of the map-elites algorithm? In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 173–180. ACM.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.

Woolley, B. G. and Stanley, K. O. (2011). On the deleterious effects of a priori objectives on evolution and representation. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 957–964.