This is a post-peer-review, pre-copyedit version of a paper published in Castillo PA & Jiménez Laredo JL (eds.) *Applications of Evolutionary Computation*. Lecture Notes in Computer Science, 12694. 24th International Conference, EvoApplications 2021, Seville, Spain, 07.04.2021-09.04.2021. Cham, Switzerland: Springer, pp. 714-728. The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-72699-7_45

A NEAT Visualisation of Neuroevolution Trajectories

Stefano Sarti[0000−0002−1780−2259] and Gabriela Ochoa[0000−0001−7649−5669]

University of Stirling, Scotland, United Kingdom,
[stefano.sarti, gabriela.ochoa]@stir.ac.uk

Abstract. NeuroEvolution of Augmenting Topologies (NEAT) is a system for evolving neural network topologies along with weights that has proven highly effective and adaptable for solving challenging reinforcement learning tasks. This paper analyses NEAT through the lens of Search Trajectory Networks (STNs), a recently proposed visual approach to study the dynamics of evolutionary algorithms. Our goal is to improve the understanding of neuroevolution systems. We present a visual and statistical analysis contrasting the behaviour of NEAT, with and without using the crossover operator, when solving the two benchmark problems outlined in the original NEAT article: XOR and double-pole balancing. Contrary to what is reported in the original NEAT article, our experiments without crossover perform significantly better in both domains.

Keywords: Neuroevoltuion · NEAT · Search Trajectory Networks

1 Introduction

NeuroEvolution of Augmenting Topologies (NEAT) is an algorithmic system that belongs to the category of topology and weight evolving artificial neural networks. NEAT strengthens the analogy between genetic algorithms and natural evolution by both optimising and complexifying the solutions simultaneously. Starting from a population of simple neural networks (without hidden units), NEAT incrementally grows them to produce more complex structures, while preserving the simplest amongst the complexified configurations. Other features of NEAT include speciation to protect innovation, genome's historical markings to facilitate recombination, and fitness sharing to ensure diversity. NEAT outperformed the best fixed-topology neuroevolution methods on a challenging pole balancing task [12]. Thereafter, NEAT has proven to be an effective and adaptable system with several applications such as dynamically evolving agents and content for video games [4, 13], generating complex musical compositions [5], evolving reaction networks in synthetic biochemical systems [3], prediction in geosciences [14] and generating trading signals for financial markets [7].

Besides standard comparative performance studies, there is a lack of tools to analyse and explain the dynamic behaviour of neuroevolution systems and their variants. This article contributes to fill this gap by bringing a recent visualisation

and analysis tool, Search Trajectory Networks (STNs), to the realm of neuroevolution. The concept of STNs was proposed in a recent conference paper [9], where the authors modelled the dynamics of two population-based algorithms when solving synthetic continuous benchmark functions. STNs are a data-driven, graph-based model of search trajectories where nodes represent a given state of the search process and edges represent search progression between consecutive states. Once a system is modelled as a graph (network) it can be visualised and analysed with the plethora of powerful analytical and visualisation tools provided by the science of complex networks [8].

The main contributions of this article are to:

- Adapt STNs to model incremental and variable length genotypes such as those in NEAT
- Offer, for the first time, a network-based visual analysis of neuroevolution trajectories
- Revisit the issue of the role of crossover in neuroevolution systems

The rest of the article is organised as follows. Section 2 describes the STN model and how to adapt it to deal with NEAT trajectories. Section 3 describes the experimental setup, including the benchmark functions, algorithm variants and parameter values. Our results are presented and discussed in Section 4, while Section 5 summarises our main findings and suggestions for future work.

2 Search Trajectory Networks

We start with some relevant definitions and follow with our proposal to store NEAT genotypes so they can be used as nodes in the graph-based STN model. Moreover, we describe the sampling process to generate the models.

2.1 Definitions

In order to define a network model, we need to specify the nodes and edges. The relevant definitions are given below.

Representative solution. A solution to the neuroevolution task at a given time step that represents the status of the search algorithm. As NEAT is a population based algorithm, we selected the best solution in the population at the given iteration as the representative solution.

Location. A subset of solutions that results from a predefined partitioning of the search space. Each solution in the search space is an element of one and only one location. Each location is assigned a representative objective value. The distinction between *solutions* and *locations* is required because in continuous domains, such as neuroevolution, the number of candidate solutions is infinite in principle. Therefore we require a coarsening or partition of the search space, which is achieved by controlling the parameters' (weights, bias) numerical precision.

Search trajectory. Given a sequence of representative solutions in the order in which they are encountered during the search process, a search trajectory is defined as a sequence of locations formed by replacing each solution with its corresponding location.

Node. A location in a search trajectory of the search process being modelled. The set of nodes is denoted by N.

Edges. Edges are directed and connect two consecutive locations in the search trajectory. Edges are weighted with the number of times a transition between two given nodes occurred during the process of sampling and constructing the STN. The set of edges is denoted by E.

Search Trajectory Network (STN) A STN is a directed graph STN = $G(N, E)$, with node set N, and edge set E as defined above.

2.2 Mapping NEAT Genotypes to Locations

Due to NEAT's dynamic nature, genotypes which encode both topologies and connections weights, can grow or shrink through generations. This makes the representation of NEAT search states hard to consistently map to the location signatures required for STNs modelling. STNs use said signatures as node identifiers to construct the aforementioned graph models. Therefore, it is important to capture and map all necessary information of a location in the NEAT search space.

To overcome this challenge, we propose using the Python object serialisation facilities, as provided by the `pickle` module function `dumps`. *Pickling* is the process whereby a Python object is converted into a byte stream. Our proposal is to serialise NEAT genomes and use the resulting byte streams as location signatures. Since the signatures are unique and contain all the genotypic information, they provide a faithful representation for the STNs nodes.

Figure 1 illustrates the details of the mapping process. NEAT genotypes encode both nodes and connections. Each node has an identifying id (key). Each connection can be either enabled or disabled. Connectionism tells us that for every node there is a bias and every connection has a weight. We extract this information from the genotype and separate them; focusing on the flow of information from nodes acted upon by other nodes. We then use this information to construct a pseudo-phenotypical vector representation (NN Representation in Fig. 1). The mapping is completed by passing this vector representation to the `pickle.dumps` function which enables us to create a flattened, compressed representation of the genotype as a byte stream.

Before the data is extracted and mapped, the numerical precision of the weights and bias values needs to be reduced. The goal is to partition the search space, and thus reduce the number of possible locations. This allows having manageable visual representations, as it was done for the continuous benchmark functions studied in [9] where a solution precision parameter was used to

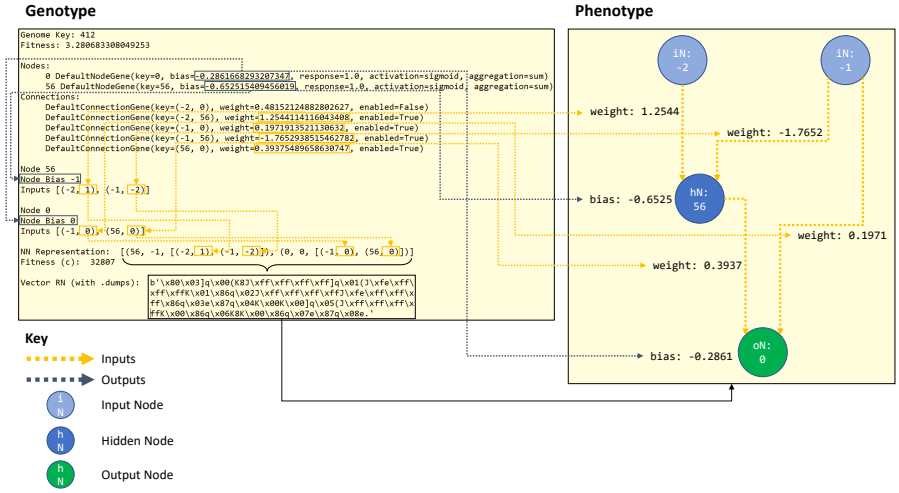

Fig. 1. Mapping NEAT genotypes to location signatures using Python object serialisation (`pickle.dumps`).

partition the continuous search space into equal-sized discrete portions. In the experiments reported in this paper, the partition is achieved by rounding off to $1e-0$ the numeric values in the genotype (weights and biases, as they are bounded in the range [-30,30], see Table 1), and to $1e-4$ for the fitness values.

2.3 Sampling and STN Model Construction

One strength of the STN model is that it does not require any additional sampling methods. Instead, the models are constructed from data gathered while running the evolutionary or metaheuristic algorithms under study. In this paper, the STNs were generated for two NEAT variants (with and without crossover) on two benchmark problems: XOR and double-pole balancing with velocities (DPV). For each problem, an STN was constructed by aggregating all the unique nodes and edges encountered across five independent runs of each NEAT variant. Section 3 gives details on the benchmark functions and parameter settings used.

We also constructed the **merged STN** model of the two NEAT variants. The merged STN model for a given instance is obtained by the graph union of the two individual graphs for that instance. More formally, let $\text{STN}_\text{A} = G(N_\text{A}, E_\text{A})$ and $\text{STN}_\text{B} = G(N_\text{B}, E_\text{B})$ be the STNs of algorithm variants A and B for a given instance. We then construct STN_merged as the union of the two graphs. Specifically, $\text{STN}_\text{merged} = G(N_\text{A} \cup N_\text{B}, E_\text{A} \cup E_\text{B})$. The merged graph contains the nodes and edges that are present in at least one of the algorithm graphs. Attributes are kept for the nodes and edges indicating whether they were visited by both algorithms or by one of them only.

3 Experimental Setup

3.1 Benchmark Problems

We considered the two classic benchmark tasks outlined in the original NEAT article [12]; XOR and double pole balancing with velocities (DPV).

XOR. Because XOR is not linearly separable, a neural network requires hidden units to solve it. This structural requirement makes XOR suitable for testing NEAT's ability to evolve structure. The fitness function is measured as the complement of the sum of squared errors (Eq. 1).

$$\mathcal{F} = 4.0 - \sum_{i} (e_i - a_i)^2 \tag{1}$$

Here e_i and a_i are the expected and actual outputs, respectively. The maximum expected output is 4.0, as these are the four possible correct output for the XOR domain. Any values equating or exceeding a fitness threshold of 3.98 are considered to have solved the problem.

DPV. The pole balancing domain is well known in the reinforcement learning literature. We considered the double-pole balancing with velocity inputs, where two poles are connected to a moving cart by a hinge and the neural network must apply force to keep the poles balanced for as long as possible within the boundaries of the track. The Runge-Kutta fourth-order method is used to implement the system dynamics. The criteria for success on this task is keeping both poles balanced for 100,000 time steps (approx. 30 minutes of simulated time). Fitness is measured as the number of time steps that both poles remain balanced (Eq. 2).

$$\mathcal{F} = 1.0 - \frac{\log t_{max} - \log t_{eval}}{\log t_{max}} \tag{2}$$

Here t_{max} denotes the maximum expected number of time-steps (100,000 in our experiments), and t_{eval} the actual number of steps during which the controller was able to maintain a balanced state of the pole within the specified limits of $\pm 36°$, within the boundaries of ± 2.4 meters of the middle of track.

This particular implementation [10] uses logarithmic scales because most trials fail approximately in the initial 100 steps. Since the solving criteria is 100,000 steps, a logarithmic scale ensures a better distribution of scores. The second term in Eq. 2 is the loss function, in the range of [0,1]. The fitness function $\mathcal{F}$ is the complement of the loss score. Hence, fitness scores are in the [0,1] range, with best performing values tending to 1.0. Any values that equate or exceed 0.98 are considered to have solved the domain (fitness threshold).

3.2 Algorithms and Parameters

In terms of the search algorithm variants, we obtained inspiration from the *ablations* study conducted in [12] to identify how each of NEAT's components help to deliver its enhanced performance. The ablations isolate key properties of NEAT to assess whether removing them causes a significant decrease in performance. Four ablations were studied in [12], non-growth, random initialisation (instead of minimal initialisation), non-speciation and non-mating.

Here, we consider the non-mating ablation. That is, we measured the performance of NEAT with and without using the recombination (crossover) operator. The experiments use the same parameter values as in the original paper [12], the principal ones are listed in Table 1.

Table 1. NEAT parameter values used on each domain.

Parameter	XOR	DPV
Population size	150	1000
Total generations	100	1000
Fitness threshold	3.989	0.989
Bias range	[-30, 30]	[-30, 30]
Weight range	[-30, 30]	[-30, 30]
Input nodes	2	6
Output nodes	1	1

The experiments were conducted with the Python implementation of NEAT, `neat-python` [6]. For each domain and algorithm variant, 30 runs were conducted in order to correctly apply statistical tests.

4 Results

We start with a statistical analysis, supported by a non-parametric test (Mann-Whitney), contrasting the performance of the two NEAT variants (with and without crossover) when solving the two benchmark problems. This is followed by a detailed analysis contrasting the search dynamics using the STN model.

4.1 Performance Analysis

For each algorithm variant and benchmark function, 30 runs were executed with the parameter settings outlined in Table 1. Table 2 summarises the success rate, average number of evaluations to reach a solution and average quality of solution reached for both domains. As results indicate, the no crossover NEAT variant has a slightly higher success rate, and is noticeably more efficient as it reaches a

solution with lower evaluations on average. Furthermore, the mean best fitness is calculated on successful runs at the stage when solution is reached. This is slightly higher for the variant without recombination, proving that this type of system produces better solutions on average. This overall observation is more marked for the DPV problem. This is in sharp contrast with the results reported in the original NEAT article [12], where the non-mating ablation study indicated that the use of crossover improves NEAT performance.

Table 2. Performance metrics.

	XOR		DPV	
	Crossover	No Crossover	Crossover	No Crossover
Evaluations (avg)	8,815.90	7,708.69	241,750.01	73,933.33
Evaluations (std)	2,760.28	2,199.09	227,395.80	74,380.52
Best fitness (avg)	3.9919	3.9924	0.99900	0.99947
Best fitness (std)	0.0060	0.0061	0.0030	0.0028
Success rate	73.3%	76.6%	93.3%	100%

In order to observe the dynamic behaviour of the two NEAT variants, Figures 2 and 3 show the average performance (best fitness) curves with error-bands (standard deviation), on the two benchmark problems respectively. The oscillating behaviour in the average fitness across generations, particularly observable in Figure 3 is expected, as for our experiments elitism was not considered. This means that at each generation, the best-performing individual may differ from the best-performing individual at the previous generation. The algorithm, however, tracks the best overall solution obtained across the run, which is then returned at the end of the run.

Given the aforementioned results, we proceed to assess for significance using the Mann-Whitney test; setting a *p-value* of 0.05. The system of hypothesis was formulated as follows.

- H_0: NEAT without crossover has similar distributions as the system with crossover.
- H_1: The two NEAT variants have significantly different distributions. Hence NEAT without crossover performs significantly better.

Table 3 displays the results related to the quality of the solution reached (effectiveness) both at midpoint (test 1) and endpoint (test 2). Moreover, we test for the evaluations required to reach solution (efficiency). As results indicate, H_0 is rejected in most cases as tested distributions differ, producing a p-value lower than 0.05, with the exception of XOR, tested at endpoint which shows no significance.

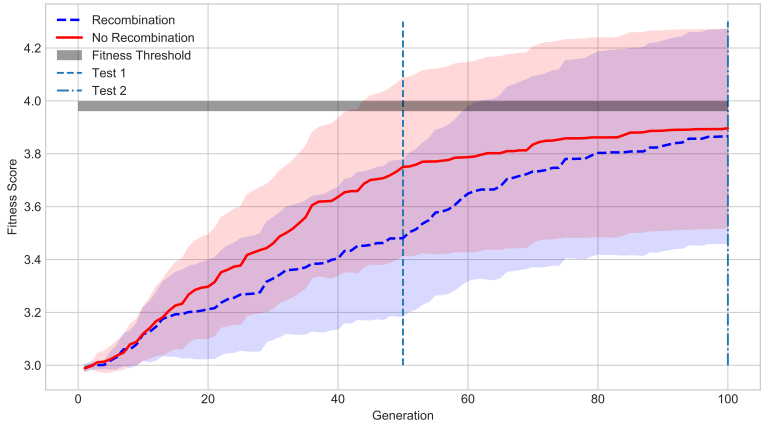

Fig. 2. Average best fitness with standard deviations across generations for the two NEAT variants on the XOR domain.

Table 3. Significance testing.

	XOR	DPV
Effectiveness midpoint test	$p = 0.00178$	$p = 0.00502$
Effectiveness endpoint test	$p = 0.32071$	$p = 0.00360$
Efficiency test	$p = 0.03911$	$p = 0.00007$

In Figure 4 and 5 these findings are substantiated by visualising the distribution of the number of evaluations to reach a solution (left plot) and the best fitness sampled at run midpoint (right plot), on the two domains respectively.

In Figure 4 we observe that in the system without crossover, the distribution of values for efficiency testing (evaluations required to reach a solution) are concentrated much lower, than its counterpart.

The second plot on the right, represents the distributions of fitness values halfway through the runs. Visibly, the system without crossover, exhibits a bimodal distribution with some values concentrating between 3.4 and 3.5, similarly as the variant with recombination. Although, in the system without crossover, greater density can be observed higher towards the upper whisker (between 3.85 and 4.0).

Comparably, Figure 5 depicts tested values for the DPV domain. In relation to the plotted efficiency (left plot), we observe that in the no crossover system, a greater and narrower density of values resides much lower in the evaluations. The distribution of values in the crossover system is similar to its counterpart, yet it shows wider variance with greater upper and lower bounds.

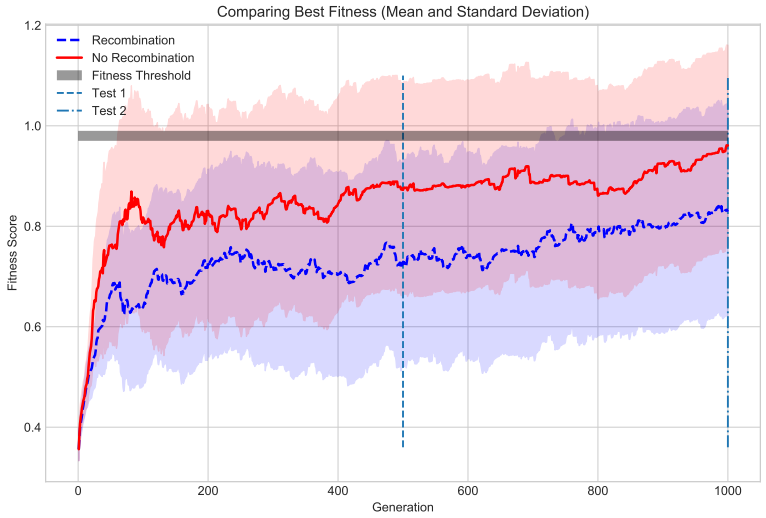

Fig. 3. Average best fitness with standard deviations across generations for the two NEAT variants on the DPV domain.

Distributions of fitness values tested at midpoint show similar spreads between the two systems. Bi-modality can be observed in both systems, in similar ways, although less accentuated in the no crossover system. Higher density can be witnessed in the crossover system at lower values, between 0.5 and 0.7. For the system without recombination, a much higher concentration is visible around higher values, between 0.9 and 1.0. Further confirming the improved performance of this system.

4.2 STNs Analysis

When networks are of moderate size, visualisation is a powerful tool allowing us to appreciate structural features which can be difficult to infer studying only network metrics. Node-edge diagrams, used here, are the most common visual representation of a network. Node-edge diagrams assign nodes to points in the 2-dimensional Euclidean space, and connect adjacent nodes by lines. If the graphs are directed, arrowheads are used to indicate the direction of connections. Nodes are then drawn on top of the edges using simple geometric shapes (such as circles or squares). Typically, the most important attributes of nodes and edges are assigned to visual properties (such as size and colour) of the shapes and lines; for instance, the area of a circle can be made proportional to the degree of the node in order to highlight hubs (i.e. highly connected nodes).

The graph visualisations in this paper were produced with the `igraph` library [2] of the R programming language. We visualised the merged STN mod-

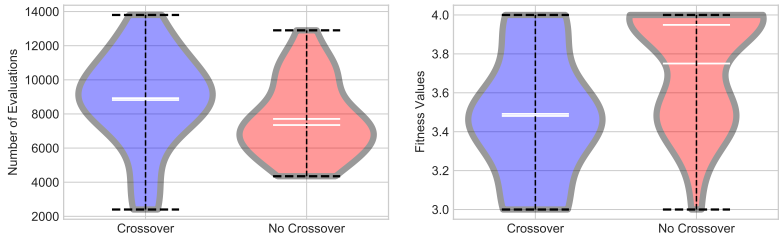

Fig. 4. Distribution (across 30 runs) of the number of evaluations to reach a solution (left plot) and the best genomes fitness values at the middle of the run for the two NEAT variants on the XOR domain.

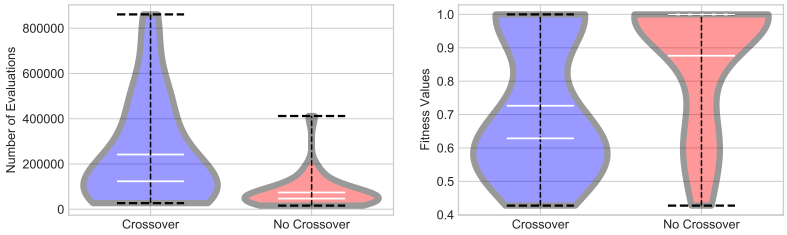

Fig. 5. Distribution (across 30 runs) of the number of evaluations to reach a solution (left plot) and the best genomes fitness values at the middle of the run for the two NEAT variants on the DPV domain.

els using the Reingold-Tilford [11] layout algorithm, which is specially suited for drawing trees (graphs without layout cycles). It generates a layout where vertices are organised into layers based on their geodesic distance (path length) from a chosen root vertex, which in our case are the start of trajectories. The algorithm also strives to minimise the number of edge crossings and to make the layout as narrow as possible.

Figures 6 and 7 illustrate the merged STN models for XOR and DPV, respectively. Nodes and edges are decorated to highlight relevant features of the search dynamics. Node sizes are proportional to their incoming weighted degree, which indicates to what extend nodes are revisited and thus attract the search process. The no crossover NEAT variant is visualised in red while the crossover variant in blue. Five trajectories were used to generate the STN for each algorithm variant, which start from the same five random seeds. This can be appreciated by the five start nodes highlighted in yellow and of larger size. All the visualised trajectories end in a different solution (fitness value above the respective threshold for each domain), which is visualised with the dark grey enlarged nodes. For both domains, trajectories from start to solution nodes are clearly shorter on average

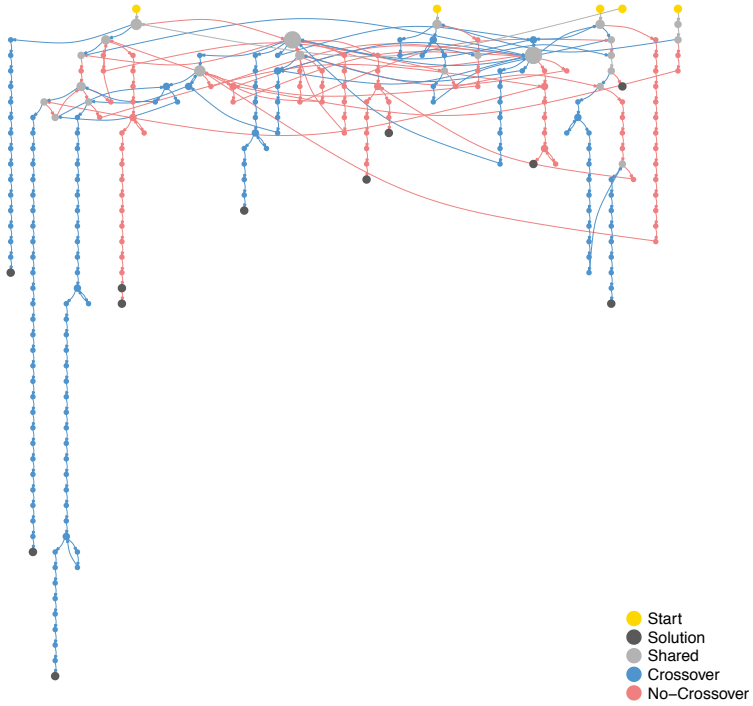

Fig. 6. Merged STN for XOR. The nodes and edges visited by each NEAT variant are identified with different colours. Light grey nodes indicate locations visited by both variants. Node sizes are proportional to their incoming degree. The start of trajectories and the nodes achieving the fitness threshold (solutions) are also highlighted in different colours and of slightly larger size.

for the no crossover variant. This is more noticeable for the DPV domain (Fig 7) where the blue trajectories (NEAT with crossover), are consistently larger.

The XOR merged STN of Figure 6, shows a larger number of nodes where the two algorithm variants overlap (visualised in light grey). The five initial nodes in yellow are shared by both algorithms, after this some grey shared nodes occur at the early stages of the search process (first steps of the search trajectory). This is consistent with the incremental grow of the NEAT genotypes, the early structures are more likely to be similar. The number of shared nodes is smaller for the more complex DPV domain (Figure 7), which can be explained by the larger genotypes for this task. Interestingly, we can observe a large shared node in the DPV domain (grey node around the top middle of the image), which is visited by the 3rd no crossover (red) trajectory and also traversed by three of the crossover (blue) trajectories. Recall that the size of nodes is proportional to their incoming degree. We hypothesise that this shared node is a good but sub-optimal configuration that tends to attract the search process with crossover.

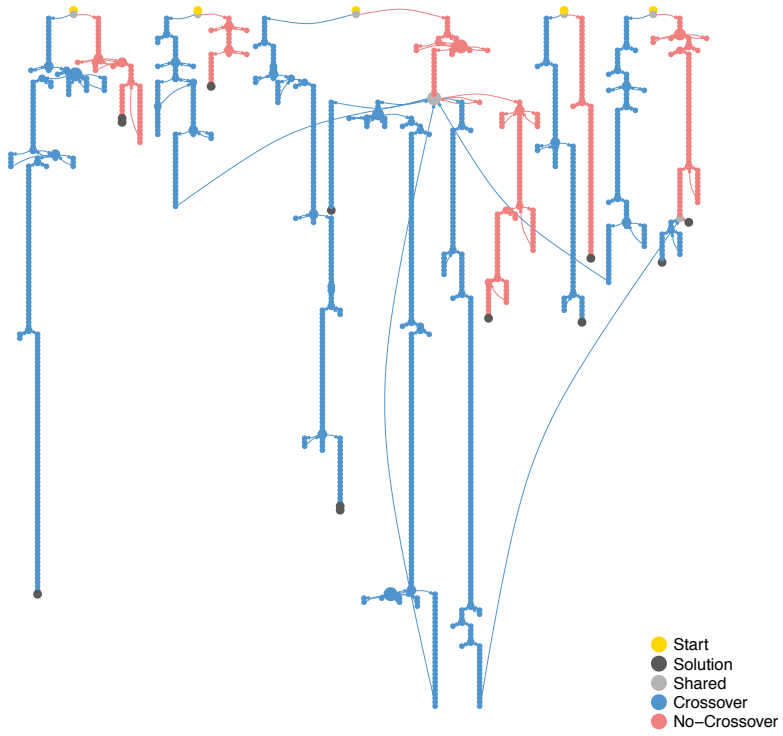

Fig. 7. Merged STN for DPV. The nodes and edges visited by each NEAT variant are identified with different colours. Light grey nodes indicate locations visited by both variants. Node sizes are proportional to their incoming degree. The start of trajectories and the nodes achieving the fitness threshold (solutions) are also highlighted in different colours and of slightly larger size.

Table 4. STN structural metrics.

	XOR		DPV	
	Crossover	No Crossover	Crossover	No Crossover
Nodes	173	112	954	320
Edges	186	121	1018	345
Path length (avg)	34.0	20	137.38	44.34
Path length (std)	13.72	8.59	58.56	23.08
Shared nodes	27		12	
Shared edges	7		5	

In order to support the visual STN analysis, Table 4 reports the following network metrics for each domain:

- Nodes: The number of nodes, which corresponds to the number of unique locations visited by each variant.
- Edges: The number of edges, which corresponds to the number of unique search transitions between locations.
- Path length: Average path length (with standard deviation) from start to solution nodes. The length of a path is the number of edges it contains.
- Shared nodes: The number of shared nodes in the STN model, which corresponds to the number of locations visited by both NEAT variants.
- Shared edges: The number of shared edges, which corresponds to the number of search transitions traversed by both NEAT variants.

The metrics in Table 4, confirm that the no crossover NEAT variant produces much shorter trajectories and thus more efficient search on both domains. The larger number of unique nodes visited by the crossover variant on both domains, indicate that crossover brings increased diversity to the evolutionary process. However, in the two domains explored, this diversity does not translate into a more efficient or more effective search. For both domains and algorithm variants, the number of edges is of similar magnitude to the number of nodes, indicating little trajectory overlap across runs; that is, only a few nodes are revisited by different runs. Instead, each trajectory follows mostly a different path, specially after the few early stages of the search process, and all trajectories end up in a successful different configuration with fitness value above the threshold. This observation can be confirmed by looking at the STN plots (Figs. 6 and 7) where all the trajectories end in different dark nodes (solutions). This occurs because numerous differing neural network topologies and weight combinations can produce similar good performance.

In terms of the number of nodes visited by both algorithm variants (shared nodes), this is relatively small in both domains (in comparison to the total number of nodes), being smaller for the more complex pole balancing domain. This is primarily due to the large variety of topologies being produced by the NEAT algorithm.

5 Conclusions

We have adapted a recent graph-based optimisation analysis and visualisation tool, Search Trajectory Networks (STNs) [9], to study the behaviour of a neuroevolution system (NEAT) [12]. To the best of our knowledge, this is the first time a neuroevolution system is visualised using this technique. Our proposal considers object serialisation in Python, combined with partitioning (rounding off) the genotypic numerical search space, in order to manage the complexity of NEAT growing genotypes.

We revisited the issue of the usefulness of crossover in neuroevolution systems. Contrary to what was reported in the original NEAT article [12] and to our surprise, the results indicate that crossover slows down the evolutionary process. We observe this, both in a standard statistical performance analysis of the NEAT variants with and without crossover, and with the recent STN analysis.

The NEAT trajectories with crossover, explored the search space more widely, thus bringing diversity to the search process. This diversity, however, does not translate into improved performance. Without crossover, NEAT trajectories towards a solution are shorter and direct; thus the process is more efficient. NEAT without crossover is also more effective, producing higher best fitness values in our experiments. This finding is consistent with an earlier observation by Angeline et al. [1] who demonstrated that neuroevolution of topologies does not need crossover to work, and indeed suggested that crossover does more harm than good.

When contrasting the structure of the STN models of NEAT against those of the classical optimisation benchmark functions studied in [9], we observed that NEAT search spaces contain multiple alternative nodes that achieved the desired fitness threshold. Therefore, there is little overlap of the neuroevolution trajectories when progressing towards good solutions. This happens because numerous differing neural network topologies and weight combinations can produce similar behaviours. This is in contrast with the search trajectories of synthetic optimisation benchmark functions, which tend to converge towards the portion of the search space containing the global optimum, or towards a small number of sub-optimal solutions attracting the search process.

Future work will explore the role of crossover in other neuroevolution settings. For example, is crossover useful when solving more challenging tasks using NEAT? What about the role of crossover in other neuroevolution systems? Furthermore, we plan to augment the STN modelling technique with additional metrics and visual decorators, which may offer supplementary instruments to analyse and understand more complex systems. In this research, object serialisation has proven to be a powerful mapping technique, that has extended the potential reach of STNs. We hope that bringing such tools to neuroevolution and metaheuristcs, will contribute to their understanding and explainability, as well as guiding the way towards improving their performance. The hope is that increasing STN adoption will further improve its analytical powers.

References

1. Angeline, P.J., Saunders, G.M., Pollack, J.B.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks **5**(1), 54–65 (1994)
2. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal **Complex Systems**, 1695 (2006)
3. Dinh, H., Aubert, N., Noman, N., Fujii, T., Rondelez, Y., Iba, H.: An effective method for evolving reaction networks in synthetic biochemical systems. IEEE Transactions on Evolutionary Computation **19**(3), 374–386 (2015)
4. Hastings, E., Guha, R., Stanley, K.: Automatic content generation in the galactic arms race video game. IEEE Transactions on Computational Intelligence and AI in Games **1**(4), 245–263 (2009)
5. Hoover, A., Stanley, K.: Exploiting functional relationships in musical composition. Connection Science **21**(2-3), 227–251 (2009)

6. McIntyre, A., Kallada, M., Miguel, C.G., da Silva, C.F.: neat-python. https://github.com/CodeReclaimers/neat-python
7. Nadkarni, J., Ferreira Neves, R.: Combining neuroevolution and principal component analysis to trade in the financial markets. Expert Systems with Applications **103**, 184–195 (2018)
8. Newman, M.E.J.: Networks: an introduction. Oxford University Press, Oxford; New York (2010)
9. Ochoa, G., Malan, K.M., Blum, C.: Search trajectory networks of population-based algorithms in continuous spaces. In: Applications of Evolutionary Computation, EvoApps. Lecture Notes in Computer Science, vol. 12104, pp. 70–85. Springer (2020)
10. Omelianenko, I.: Hands-On Neuroevolution with Python. Packt Publishing, Limited (2019)
11. Reingold, E.M., Tilford, J.S.: Tidier Drawings of Trees. IEEE Transactions on Software Engineering **SE-7**(2), 223–228 (1981)
12. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10**(2), 99–127 (2002)
13. Stanley, K., Bryant, B., Miikkulainen, R.: Real-time neuroevolution in the nero video game. IEEE Transactions on Evolutionary Computation **9**(6), 653–668 (2005)
14. Wang, G., Cheng, G., Carr, T.: The application of improved neuroevolution of augmenting topologies neural network in marcellus shale lithofacies prediction. Computers and Geosciences **54**, 50–65 (2013)