# Visualizing the Dynamics of Neuroevolution with Genetic Distance Projections

Evan H. Patterson
ehp2095@rit.edu
Rochester Institute of Technology
Rochester, NY, USA

Joshua Karns
jak5763@rit.edu
Rochester Institute of Technology
Rochester, NY, USA

Zimeng Lyu
zimenglyu@mail.rit.edu
Rochester Institute of Technology
Rochester, NY, USA

Travis Desell
tjdvse@rit.edu
Rochester Institute of Technology
Rochester, NY, USA

## Abstract

Evolutionary algorithms have shown substantial progress in recent years, especially in neural architecture search applications, or neuroevolution. Despite their effectiveness, analyzing and understanding the evolutionary paths these algorithms traverse to reach solutions remains challenging. Often these algorithms involve distributed computing strategies, which can include subpopulations or islands, and they explore massive or even unbounded search spaces which can include both weights and architecture, in both continuous and non-continuous domains. Manually examining individual solutions to understand the evolutionary dynamics is often infeasible due to large population sizes, large genome sizes, and high generation counts. This work introduces a new methodology for visualizing neuroevolution population dynamics called genetic distance projections, along with a novel neural network based method for generating these representations. We evaluate this methodology empirically and find it performs better than other traditional methods in generating these representations. We further validate the usefulness of these visualizations using case studies from EXAMM, a long standing neuroevolution algorithm, in which one case study even led to finding and fixing a bug in EXAMM's algorithm.

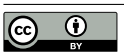
## CCS Concepts

• **Networks** → *Network dynamics*; • **Human-centered computing** → **Visualization design and evaluation methods**; • **Theory of computation** → **Evolutionary algorithms**.

## Keywords

Neuroevolution, Visualization, Evolutionary Dynamics

**ACM Reference Format:**
Evan H. Patterson, Joshua Karns, Zimeng Lyu, and Travis Desell. 2025. Visualizing the Dynamics of Neuroevolution with Genetic Distance Projections. In *Genetic and Evolutionary Computation Conference (GECCO '25), July 14–18, 2025, Malaga, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712256.3726457




*GECCO '25, Malaga, Spain*

ACM ISBN 979-8-4007-1465-8/2025/07
https://doi.org/10.1145/3712256.3726457

## 1 Introduction

Evolutionary algorithms have made significant advancements in recent years, particularly in applications involving Neural Architecture Search (NAS). In evolutionary NAS, neural networks are iteratively refined through processes such as mutations and crossovers within genome populations. Unlike other NAS methods that rely on predefined network optimization mechanisms, evolution-based NAS selects candidate genomes based on fitness evaluations. This selection process complicates the interpolation of evolutionary progress, especially when multiple evolutionary and reproductive rules are concurrently applied during the search. Throughout the evolutionary process and fitness evaluations, new genomes are generated through random mutations and crossover operations. However, the effectiveness of specific genes and their contributions to the optimal solution remain unclear. Visualizing the evolution process by mapping the two-dimensional relationships among all existing genomes can therefore be highly beneficial. Such visualization can illustrate the trajectory through which the global best genome emerges and enable tracing back the particular mutations and crossovers that contributed to its development.

In a seminal 2021 work, Ochoa *et al.* criticize the wide range of metaheuristics published with no focus on understanding the behavior – the manner in which the search space is actually traversed at runtime is not analyzed, and only algorithm performance is observed [9]. Their criticism is inspired by an argument made by Sörensen, in which he criticizes the flurry of new metaheuristics being released which are based on metaphors from nature without providing a critical analysis of the fundamental behavior of the search which separates it from existing work [14].

To address this, the Ochoa *et al.* proposed a novel strategy to visualize the search trajectory of metaheuristics – Search Trajectory Networks (STNs) [9]. Their method captures data at runtime and constructs a graph from solutions. Nodes correspond to solutions and edges correspond to ancestry – node A is connected to node B if B is a child solution to A. This method provides a robust way to visualize search trajectories, but fails to accurately represent the similarity of solutions from disparate lineages.

While the arguments posed by Ochoa and Sörensen do not map directly onto the field of neuroevolution, one key component does – there has been a wide variety of metaheuristics proposed in the last decade that, while unique in their mechanics, do not demonstrate novelty in the manner in which the search space is traversed. To

address this, this work proposes the genetic distance projection (GDP): a novel analysis and visualization strategy based on the work of Ochoa *et al.* [9], improving on their work by (1) applying a similar visualization strategy to an incredibly high-dimensional search space and (2) providing a visual representation of genetic distance in a 2D projection.

We evaluate multiple methodologies for generating a GDP, including traditional methods such as principal component analysis, singular value decomposition, and multidimensional scaling, and provide empirical results for how our novel neural network methodology provides better representations using smaller benchmark runs. We then further validate this methodology using three large-scale case studies using the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [11]. As a result of this visualization process, we even found and fixed an interesting bug raising from the interactions of EXAMM's island repopulation with crossover operations.

## 2 Related Work

This paper was partially inspired by Search Trajectory Networks (STNs) [9], which studied the visualization of metaheuristic behavior. The motivation behind this paper was primarily to develop tools for analyzing and visualizing the behavior of metaheuristic algorithms. Although this study did deliver as promised, it may have several limitations. The STN methodology uses partitioning, which creates grid-like divisions of the search space rather than dynamically capturing the landscape of local optima or inherent properties of solutions, meaning that it may be limited in its ability to convey the topology of the search space to a human user. It can convey the paths that solutions travel along but does not convey the relationships between the solutions. In other words, one cannot discern, by looking at the graph, how similar two solutions truly are. Additionally, STNs have not yet been demonstrated with larger dimensions than ten meaning that its usefulness and generalizability may be limited, especially in evolutionary algorithms that develop neural networks with thousands or millions of nodes and edges. Additionally, representative solutions don't necessarily convey anything about variance *between* solutions at a given time step or generation. There may be important differences between solutions in a population at a particular time step that are overlooked due to this. This is especially important when crossover is involved. It may not always be immediately apparent how two parents from different lineages may combine to produce a more optimal solution.

There are other similar visualizations tools that have been created and documented due to the need for understanding these evolutionary algorithm processes to improve their performance and find potential issues [8]. As such, there are a number of different methods and tools that have shown promise in this area, and provided evidence of the benefits of EA visualization [13, 3, 4, 12, 1, 9]. That being said, they leave gaps that this work aims to address. For example, GAVEL [3] was used to identify that a GA which was used to solve the long path problem was doing so through a fortuitous mutation, rather than by slowly building on an optimal solution. While GAVEL can offer practical benefits, it still requires pruning of the genome relations down to ancestors of the final best solution. EAVis [4] shows the exact set of genes of the global best for each generation, but this creates a natural limitation in the size of a genome that can be displayed, and the coverage maps created by Shine et al. [13] do not convey much about the topology of the search space or the differences between the genomes.

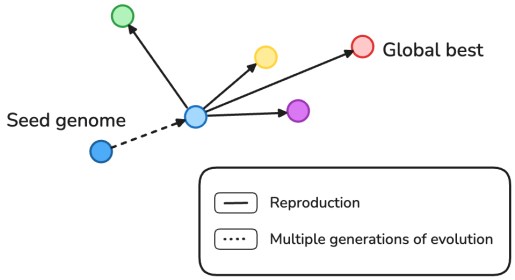



**Figure 1: A simple illustration of information that may be conveyed.**

## 3 Methodology

In Figure 1, a graph is shown containing nodes representing several genomes, forming a lineage. Evolution starts with a seed genome, which produces another genome (light blue), which may produce children (other colors), and a new global best genome. When looking at various genomes that have appeared throughout a neuroevolution run, it should be immediately evident to a human user how similar these genomes are to each other. If a population has been separated into multiple branches or lineages, the visualization should clarify whether two genomes belong to the same or different lineage and accurately represent the degree of similarity between lineages. It should allow users to easily trace the ancestry of the algorithm's final preferred solution. This type of information can be very beneficial when trying to improve or fine-tune evolutionary algorithms.

As we are working with evolutionary algorithms that evolve populations over time, the primary focus is on representing the progression of genomes through the search space and tracing persisting lineages. When distinct groups of solutions emerge, each exploring different optimization paths, our goal is to visualize these branches and convey their relative similarity to one another in a way that is understandable and interpretable from a human perspective. Ultimately, the purpose should be to create methods of analyzing the algorithms in question that allow us to identify ways that the algorithms can be made more effective and that will facilitate further algorithm development and refinement.

### 3.1 Definitions and Overview

This section provides an overview of the technique, named *Genetic Distance Projection* (GDP). Variations of this method (referred to later as NN-GDP, MDS-GDP, PCA-GDP, SVD-GDP, and SNN-GDP) use different methods of dimensionality reduction to reduce the contents of the genomes to two dimensions. In GDP, a population of $m$ genomes is encoded into vectors, each of size $n$, to form the matrix, $G = [g_1, g_2, \ldots g_m]$. We define a mapping function $f : \mathbb{R}^n \rightarrow \mathbb{R}^2$, which projects genomes into a two-dimensional space,

producing the position matrix $V$, where $V = [v_1, v_2, \ldots v_m]$, such that $f(g_i) = v_i$.

Given a matrix $G$ containing an encoded set of genomes, we compute the pairwise Euclidean distance matrix $D(G) = [d_{i,j}]$, where $d_{i,j} = \|g_i - g_j\|$, represents the distance between genome vectors $g_i$ and $g_j$. Similarly, for a matrix $V$ representing another set of encoded vectors, we define the corresponding pairwise Euclidean distance matrix as $D(V) = [d_{i,j}]$, where $d_{i,j} = \|v_i - v_j\|$ denotes the distance between vectors $v_i$ and $v_j$ in $V$. Given $\gamma \in \mathbb{R}$: An arbitrary constant used to scale distances, we try to perform a mapping of $f$ such that $D(G) \approx \gamma D(V)$. The resulting 2D positions are used to position the nodes of a directed graph, in which edges represent parent-offspring relations between the genomes. The graph is displayed to a user in the form of a figure, intended to convey how similar genomes are.

Finally, when testing the performance of these algorithms, $q$, and $u$ are calculated such that $q = \text{flatten}(D(G))$ and $u = \text{flatten}(D(V))$, and the closest distance is found with $\|q - \text{proj}_u q\|$.

## 3.2 Gene Data Preprocessing

Before we can map genomes to positions, the genomes must be vectorized. Each genome within the population is defined by two sets of genes: edge genes and node genes, which are stored as lists of unique identifiers. To make these lists usable in our processing pipeline, they are encoded into multi-hot encoded vectors. Each vector represents whether a genome contains a specific gene by setting corresponding entries to 1 if present, or 0 if absent. Figure 2 uses an example to illustrate this process.

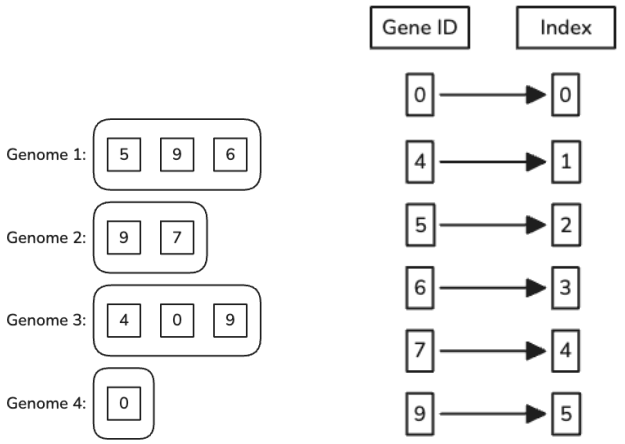
For graph based neuroevolution techniques such as EXAMM and NEAT [15], edges and nodes in a network can both be represented as genes. When the set of genes encoding edges is converted into this format, it becomes a matrix of shape $(m,\ n_e)$, where $n_e$ is the total number of edge genes that a genome can possess. Likewise, when the set of genes encoding graph nodes is converted to $(m,\ n_n)$, where $n_n$ is the number of possible node genes. For this implementation, the two matrices were concatenated into a matrix of shape $(m,\ n_e + n_n)$ or $(m, n)$. This matrix would then be used to generate positions for the genomes by performing a mapping of $n$ dimensions to 2.

To accurately represent each genome's full structure, we combine matrices derived from edge and node genes, allowing for a more comprehensive representation of genome family relations and evolution. In addition to the positioning that is done here, nodes may also be colored to make the graphs that are shown more visually informative. These methods are designed to allow human users to easily understand and conceptualize how genomes change over time.

## 3.3 Problem Statement

The goal of this methodology is to visualize the optimization process in such a way that, given a pair of genomes, $g_i$ and $g_j$, the visualization makes it apparent to a human observer that $\|g_i - g_j\|$ is based on $\|f(g_i) - f(g_j)\|$.

For this problem, the family tree represents the ancestry and succession of the genomes in the population. The positions of the nodes in the graph (which correspond to genomes) are intended to be represented by 2D positions such that the distances shown




**(a) Gene lists of several genomes.**

**(b) How genome IDs are mapped to vector indices.**

| | | | | | | |
|---|---|---|---|---|---|---|
| Genome 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| Genome 2 | 0 | 0 | 0 | 0 | 1 | 1 |
| Genome 3 | 1 | 1 | 0 | 0 | 0 | 1 |
| Genome 4 | 1 | 0 | 0 | 0 | 0 | 0 |

**(c) The resulting binary vectors.**

**Figure 2: An illustration of the genome vectorization process.**

(at least roughly) convey the distances between the corresponding genomes. More formally: given some function, $f\colon \mathbb{R}^n \to \mathbb{R}^2$, we should ideally be able to perform a transformation from the genome size to two dimensions, such that the differences between outputs are essentially scaled versions of the differences between inputs.

$$\gamma\|f(g_i) - f(g_j)\| = \|g_i - g_j\| \tag{1}$$

This ideal is not always possible because the transformation may either be lossy or show results that aren't distinct, as it is from an arbitrarily large number of dimensions to only two. To frame the problem in a way that is solvable, we aim to minimize the expression:

$$(\|g_i - g_j\| - \gamma\|f(g_i) - f(g_j)\|)^2 \tag{2}$$

## 3.4 Traditional Distance Projection Methods

To attempt to capture the relationships between the genome values and positions simply, two linear transformation based methods were attempted, including principal component analysis (PCA), singular value decomposition (SVD), as well as multidimensional scaling (MDS), which is not strictly linear. Lastly, we propose a novel neural network based approach for more effectively capturing these relationships.

*3.4.1 Principal Component Analysis and SVD-Based GDP (PCA-GDP, SVD-GDP).* One very straightforward approach is to perform

a linear transformation designed to minimize the above expression. PCA is a common method used for applications of this type. If we allow ourselves to be content with a comparison of inner products rather than distances, we can substitute $g_i \cdot g_j = f(g_i) \cdot f(g_j)$ for Equation 1, and PCA may be used, while also allowing $f$ to be a linear transformation. If we want better numerical stability than with PCA, SVD is another commonly used method.

*3.4.2 Multidimensional Scaling-Based GDP (MDS-GDP).* The main issue with the aforementioned methods is that the space being traversed is nonlinear. MDS is one method that may be capable of overcoming this limitation. If we want to assume that the distance between the inputs and outputs of $f$ are equal, as in Equation 1, we can set the distance matrices of $G$ and $V$ equal to each other:

$$D(G) = D(V)$$

Multidimensional scaling (MDS) may then be used to derive a possible matrix $V$, from $D(V)$.

## 3.5 Neural Network-Based GDP (NN-GDP)

In addition to the previous methods, we present a novel methodology which utilizes a neural network to capture relationships that are not limited to linear mappings. This involves using a neural network with two hidden layers to map $n$ dimensions to 2 output values. The model uses two linear layers, and a `leaky_relu` activation function for both hidden layers, with no activation function used in the output layer. The first hidden layer in the model used here was of size 256, and the second was of size 128. The Adam optimizer was used to train the neural network, with a learning rate of 0.001, and a mean squared error loss function.

During the training loop, two genomes, $g_i$ and $g_{i+1}$, are passed jointly into the model for each forward and backward pass. If the mapping performed by the model is represented by $f$, we are minimizing expression 2, mentioned earlier.

*3.5.1 Neural Network: Scaling the Norms.* Before training, we need to determine a scalar value to scale the outputs with, so $\|\gamma f(g_i) - \gamma f(g_j)\| \approx \|g_i - g_j\|$. This is important because we are training the neural network to match the distances between input vectors. If we translate and rotate like above, the matrix is forced to produce output vectors with magnitudes in a range that may not match the magnitudes of the genomes. This is not desirable because we are directly minimizing the differences between the output and input vectors. We will determine an initial value of the output by getting the average value of $\frac{\|g_i - g_j\|}{\|v_i - v_j\|}$. When a forward pass is performed through the neural network, we will multiply the resulting vector by $\gamma$ before returning it.

*3.5.2 Neural Network Training.* The neural network is trained to minimize Expression 2 for all $i$ and $j$, which would, in theory, imply each pair of genomes in the dataset would be passed through the model during each training epoch.

Because the number of genes in total increases as the number of generations increases, $n$ is actually proportional to $m$. Since the number of forward passes as a function of $n$ is quadratic, and $n$ is proportional to $m$, the overall, worst-case time complexity of one epoch of the training process would be quartic (exponent of four).

Given the significant computational expense of this approach, a slightly different strategy was used. With this new strategy, each element of the dataset is randomly paired with another, and a single pass is done through all pairs of elements. When used with mini-batch gradient descent, the data is shuffled, each batch is split in half, and the corresponding elements of the two halves are paired with one another. The neural network is then be used to predict the positions of paired elements. If the two batch elements are $g_i$ and $g_j$, the training loss is then calculated based on Expression 2. With this process used instead, the resulting time complexity is cubic.

This process can be shown in Figure 3.

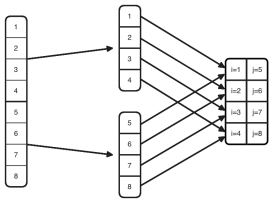



**Figure 3: How a batch of size 8 would be split up.**

*3.5.3 Simple Neural Network-Based GDP (SNN-GDP).* Because the rationale for this method is that neural networks, being nonlinear, can approximate functions that linear methods cannot, it may be useful to test this by removing the nonlinearities from the neural network described above to see if the advantage is lost. To test our assumption, we use the neural network described earlier exactly the same way, except we modify it to only use one linear layer and no activation. To be clear, this neural network uses the same loss function, optimizer, number of epochs, and data, but does not have any hidden layers, and does not use any activation functions, making it equivalent to a single linear transformation.

## 3.6 Post-Processing of Positions

The figures provided by these distance projections may be easier to read if the graph is presented in a way that shows progress in a particular direction because, roughly stated, the user will know what to expect. In the case of this algorithm, the graph will always be rotated to show the root member of the family tree on the position $\begin{pmatrix}0\\0\end{pmatrix}$ while the final best is shown on $\begin{pmatrix}1\\0\end{pmatrix}$. This can be done simply with translations and rotations after the positions have already been calculated.

To center the graph on the origin, we subtract everything from the root member's position. We then find $\theta = \arctan2(v_{\text{best}})$, and multiply all positions by the matrix, $R$, if

$$R = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

The positions of the nodes, $\dot{V}$ that are shown in each figure are thus calculated as:

$$\dot{V} = R(V - v_{\text{root}})$$

## 3.7 Coloring

The graph contained in any figures generated will have nodes that need to be colored. This is another opportunity to include

information about the neuroevolutionary process. Insofar as this is done, there are three different ways of coloring the nodes of the graph. These include mapping the genome's contents to colors in a similar vein to the position calculations already mentioned, using the fitness of each node to determine coloring, and using timing to determine coloring. Multiple approaches are used to give an idea of how each attribute individually varies across the solution landscape. When representing the contents of genomes using colors, the computations used for positioning may also be used for coloring, the only two differences being that the number of target dimensions is three instead of two, and that there must be a normalization to the range $[0, 1]$.

To translate the losses to colors, the best approach appeared to be to convert the fitnesses to a normalized percentile ranking from 0 to 1. The fitness of a genome becomes the proportion of genomes in the population that would rank below it in terms of loss. The reason for this approach was mainly that loss curves would vary widely enough between runs that normalizing using usual methods, such as min-max normalization would not work. It is very possible that more sophisticated techniques would offer benefits here, but that is not the focus of this paper. It may also be worth noting that in the case of EXAMM, mean squared error training loss is the metric that is used for sorting genomes, so the algorithm is minimizing, which means that lower "fitness" values in this case are considered more optimal.

It should also be noted that EXAMM has distinct groups of genomes, or "islands," that genomes evolve on in parallel [7]. Due to this, another method of coloring included coloring nodes based on the island they belonged to.

## 4 Results

Initial results utilized the EXA-STAR framework, which is a new Python and PyTorch implementation of the Evolutionary eXploration of Augmenting Memory Models (EXAMM) [10] and Evolutionary eXploration of Augmenting Convolutional Topologies (EXACT) [2] algorithms. As EXACT and EXAMM have been implemented in C++, EXA-STAR is being developed to provide a more easy-to-use and extensible framework for designing neuroevolution algorithms. As EXA-STAR is currently under development, it does not yet fully implement EXAMM, this work also utilizes the full EXAMM algorithm for more representative real world case studies, as it provides a number of state-of-the-art methods to improve the performance of neuroevolution, such as Lamarckian inheritance of weights to reduce the amount of training epochs required [6], multiple island populations with extinction and repopulation events to prevent premature convergence [7], and co-evolution of training hyperparameters [5, 16], of which only Lamarckian weight inheritance has been ported to EXA-STAR. For this work, both EXA-STAR and EXAMM were updated to provide genome traces consisting of JSON outputs containing the information required to create the genome vectorizations presented in Section 3.2. The latest version of the code used to generate the visualizations seen in this paper has been made publicly available on GitHub[1].

[1] TheDeepDaemon/genetic-distance-projection

### 4.1 Evaluating Visualization Quality

To make an empirical evaluation of the methods presented in this work, a direct comparison between the similarities between genomes and the similarities between their corresponding positions in the visualization is required. For this, two metrics were used to quantify each method's performance. The first was a correlation-based metric and the second was a distance-based metric.

To calculate these two metrics of performance, two vectors are used: $q$, which represents the distances between all combinations of the distances between pairs of genomes; and $u$, which represents all combinations of distances between all combinations of positions of genomes in the visualization. These were calculated by flattening their two respective distance matrices, and they ultimately include all comparisons between vectors of interest. The vector $q$ is defined as:

$$q = \text{flatten}\left([\|g_i - g_j\|]_{1 \le i < j \le m}\right) = [\|g_i - g_j\|]_{1 \le i < j \le m}$$

and similarly for $u$:

$$u = [\|v_i - v_j\|]_{1 \le i < j \le m}$$

The different visualization generation methods were compared using the Pearson correlation between $q$ and $u$, and also the closest possible distance between the two vectors.

The EXA-STAR algorithm can be run for a predetermined number of generations, and for these tests, that ranged between 10 to 100 generations. The population size was set to 10, meaning that up to 10 genomes could exist per generation. For testing, the number of generations was varied, and the results were collected and averaged for each number of generations. Each data point is the mean value achieved over 10 runs of EXA-STAR.

*4.1.1 Correlation-Based Metric.* Correlations between $q$ and $u$ are displayed in Figure 4. Each point marks the mean correlation between vectors $q$ and $u$ given some number of generations. The whiskers above and below each point show the highest and lowest correlations given some method at each number of generations. Because we are measuring the correlations between $q$ and $u$ in this graph, methods that are more effective at generating positions that fit our criteria will tend to give higher values. NN-GDP has the highest mean at all numbers of generations above 20. This is an indication that NN-GDP may produce positions that most accurately reflect the similarities between the genomes.

*4.1.2 Distance-Based Metric.* To compare different methods of positioning, the distance between two genomes was compared to the distance between their computed spatial distances in the visualization. However, if we simply measure the distance between $q$ and $u$, it is possible to end up with misleadingly inaccurate results. This is due to a significant difference in the magnitudes of the distances calculated, which can result from the significant differences in dimensionality between the genome distance and the spatial visualization distance. In Section 3.5.1, a $\gamma$ value was used to fix this problem during the optimization process. A related strategy is used here for the purpose of comparison. To compare $q$ and $u$ accurately, we get $\|q - \text{proj}_u q\|$. This results in a comparison of the closest possible distances rather than the absolute distances.

Figure 5 presents a graph of the distances between genomes and positions over all numbers of generations tested. The whiskers

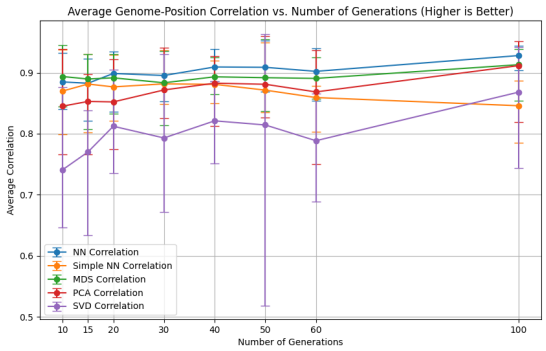



**Figure 4: Graph of the correlations between $q$ and $u$.**

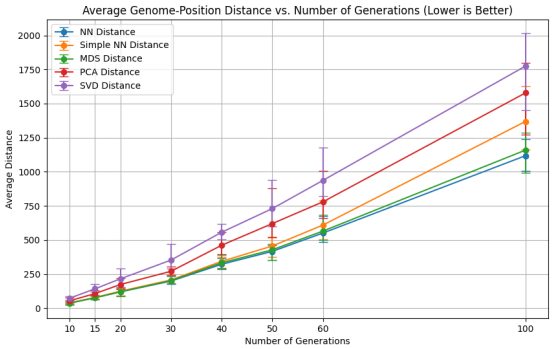
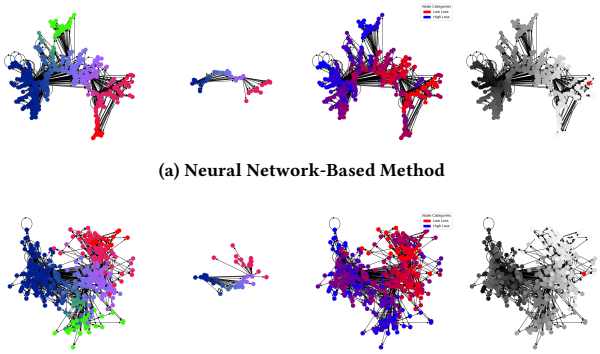



**Figure 5: Graph of the distances between $q$ and $u$.**

above and below each point show the highest and lowest values for each method. This plot shows a clearer picture than the graph of correlations, as the distances increase as the number of generations is increased, which is a consequence of the fact that longer runs result in more varied and complex genomes, with larger numbers of genes, which leads to increased dimensionality. By this metric, MDS and the neural network-based method outperform the other methods significantly. Upon closer inspection, the neural network performs slightly better than MDS, and the differences become more pronounced as the scale increases.

## 4.2 Visualization Methodology Effectiveness

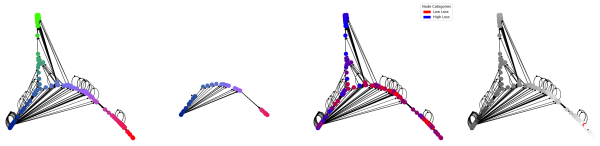
Figure 6 presents example visualizations of the methods described in Section 3 on the previous EXA-STAR test cases after 100 generations. PCA-MDS and SVD-MDS did not convey much important information, validating previous empirical results. On the other hand, the neural network-based method used here and MDS offered significant benefits over the others. Because PCA and SVD are designed to maximize variance, they are not the optimal choices for algorithms that produce positions that accurately convey how similar or different the genomes in question are. Consequently, when used for positioning nodes in some of the graphs included below, the positions do not vary in a way that is very informative. On the other hand, when used for determining the coloring of nodes, the



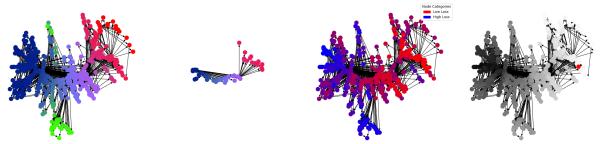
**(a) Neural Network-Based Method**



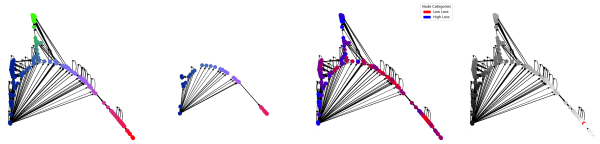
**(b) No Layer Linear Activation Neural Network-Based Method**



**(c) PCA-Based Method**



**(d) MDS-Based Method**



**(e) SVD-Based Method**

**Figure 6: Visualizations of the various genome placement methods. From left to right, the first image shows genomes with colors derived using PCA. The second shows only the global best genomes. The third displays genomes with colors that are derived based on fitness (lower losses are red, higher losses are blue). The last shows the graph of family relations.**

graphs tend to appear variegated. Due to this, PCA is used as one of the options for coloring graphs, in addition to fitness and timing.

The primary reason that NN-GDP and MDS-GDP performed better to such a significant degree appeared to be that they were capable of performing mappings that were nonlinear. The neural network with nonlinearities included was measurably more accurate, and it produced more visually informative graphs. This seems to indicate that the nonlinearities were essential for producing graphs according to the criteria laid out.

Additionally, according to these metrics, NN-GDP is slightly better than MDS-GDP. That said, the NN-GDP is more computationally expensive than the MDS-GDP. The graphs produced by the

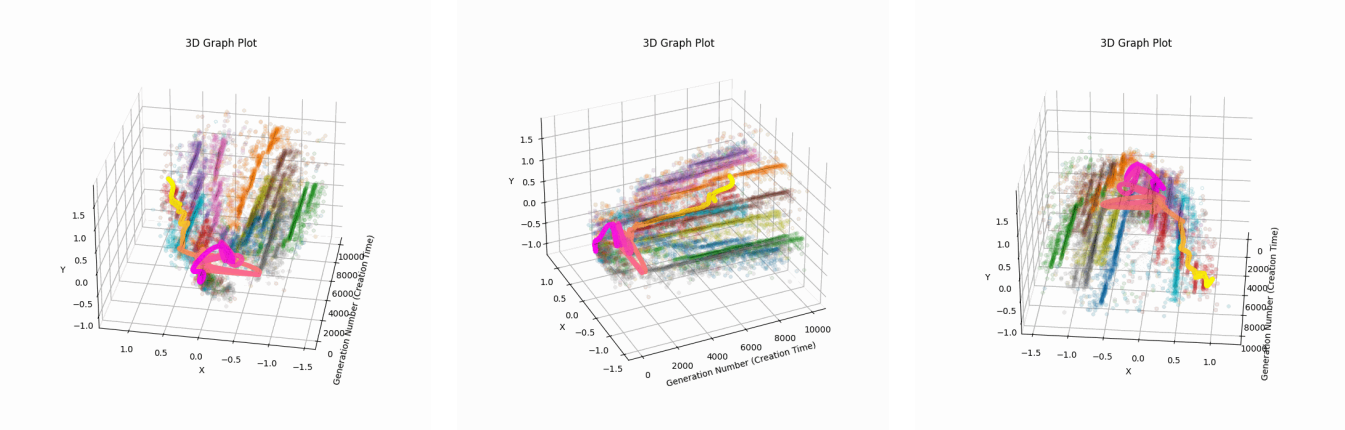



**Figure 7: Visualizations of the EXAMM run with no island repopuation.**

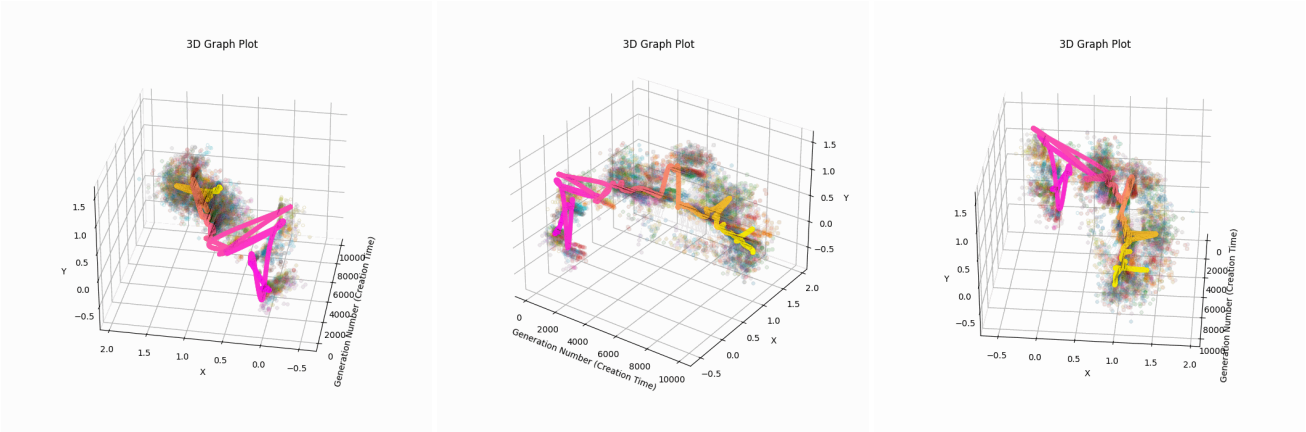



**Figure 8: Visualizations of the EXAMM run with repopulation and low crossover.**

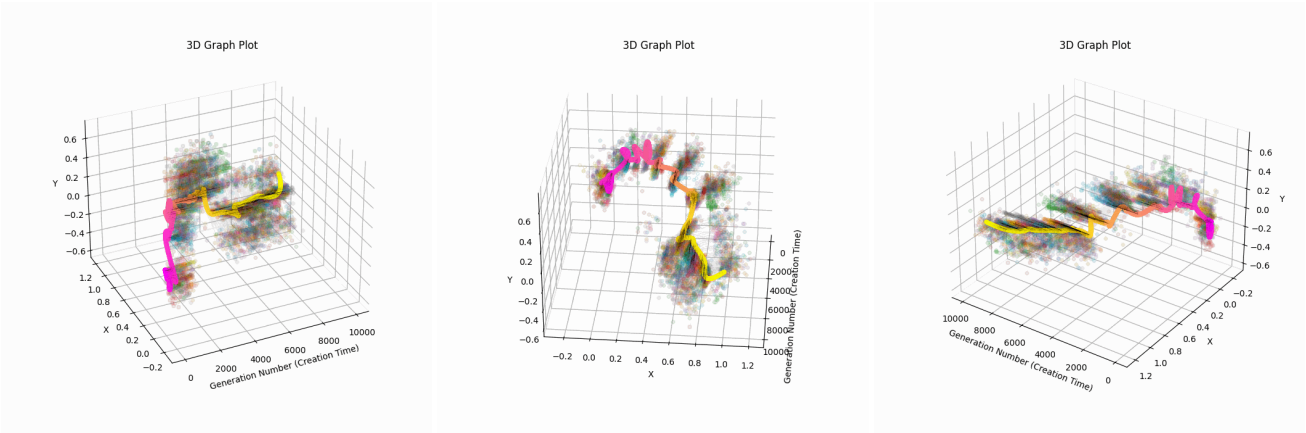



**Figure 9: Visualizations of the EXAMM run with repopulation and high crossover.**

neural network-based method also tend to be quite visually messy, while the MDS-based method tends to be less so. This may mean that the MDS-based method has the strength that a human using the graph will more easily read and interpret the results. This, in combination with the faster computation time, may mean that the MDS-based method is better in at least some ways for practical purposes despite producing graphs that are slightly less precise in conveying genome similarity.

## 4.3 EXAMM Case Studies

In addition to the previous experimental EXA-STAR runs, three different large-scale runs were performed in EXAMM to better highlight the capabilities of this visualization methodology. In each case study, EXAMM was run for 10,000 evolved neural networks, utilizing its standard suite of mutation and crossover operations, along with 10 island populations. Each of these three runs had variations to the algorithm which present themselves visually using GDP, and in the case of the second case study, it even led to an improvement in the algorithm. For higher resolution and more viewpoints, rotating animations of the visualizations are provided in the supplementary material.

*4.3.1 Case Study 1: No Repopulation.* EXAMM provides a methodology that allows for the extinction of poorly performing islands, followed by a repopulation from other well-performing islands, which has been shown to significantly improve the performance of the algorithm [7]. Figure 7 presents 3D visualizations of the NN-GDP method where the Z axis is each genome's generation time. A colored line representing each new best genome is shown (earliest best genome in pink which transitions to the final best genome in orange). Genomes are otherwise colored by island, and we can see that genomes from each island stay in fairly distinct regions in the visualization and many islands do not converge to the global best (which remains near only within a single island).

*4.3.2 Case Study 2: Low Crossover.* To highlight the effects repopulation adds to EXAMM's neuroevolution process, Figure 8 provides an example of a run where repopulation is turned on and every 150 generated genomes an extinction event occurs where the worst performing island is repopulated with mutations and crossovers from the best performing. In contrast with the no repopulation run, we see a much greater level of interaction between islands, which progresses over time as they reach the final global best genome.

Most interestingly, while visualizing this experiment we saw that crossover (which should be occurring 20% of the time, and is shown by having arrows from two parent genomes go to a child genome) was happening significantly less than expected. On investigating EXAMM's code, we found that EXAMM only performed crossover when all islands were full, so during the repopulation process, crossover was turned off, which ended up being most of the time due to the frequent repopulation events. Consequently, we implemented an update to EXAMM where crossover could occur within any island as long as its population was full, and between islands if at least 2 islands had full populations. This finding and fix led us to case study 3.

*4.3.3 Case Study 3: High Crossover.* Figure 9 shows the last EXAMM run which incorporates the fix to crossover frequency found from visualizing case study 2. We can see in these visualizations a stepwise pattern due to increased crossover where the islands more quickly converge to new best found genomes and then hop to a new position when a new best found genome with a more different architecture is found. This stands in contrast to case study 2 where there is a more gradual transition due to slower convergence from the EXAMM run utilizing mostly mutation. It should be noted that in some cases this stepwise behavior could indicate a problem, however after implementing the fix it was found that the higher crossover runs tended to find better genomes more quickly.

## 5 Conclusion

This work introduces genetic distance projection (GDP), a new methodology for visualizing the evolutionary dynamics of neuroevolution or neural architecture search algorithms. It operates by vectorizing genome representations generated as trace information over the execution of a run, and mapping these representations to a 2D space such that the distance of any two genomes in the 2D space is representative of their distance in the high dimensional original space. This representation can then be visualized with varying color schemes, e.g., by fitness value, distance, or even other data like island population, with other important genomes (e.g., global best genomes) highlighted. It can also be shown using a third dimension as generation time to get an idea of how a search progresses over time. GDP, in practice, allows visualization of connections between genomes over time, what groups or populations of genomes are most closely related, and whether groups are sectioned off into different populations or species.

This work evaluated a variety of methods for performing GDP and found that our novel method based on training neural networks provided the best visual information when empirically compared to other methods. Further, three different runs from the EXAMM neuroevolution algorithm with different settings were visualized as case studies of how this visualization can be useful in understanding evolutionary dynamics. In particular, in performing one of these case studies we found that EXAMM's crossover operation was not working as expected, leading to an improvement in the algorithm.

GDP provides a powerful new way to understand and visualize the dynamics of neuroevolution algorithms, and this work also has multiple avenues for improvement and extension. In particular, the vectorized genome representations only incorporated the presence or absence of an edge or node in the genome, but not actual weight values in the neural network. Extending the methodology to include these values for even better distance information could provide further insights into the evolutionary process. Further, many neuroevolution algorithms are memetic, where backpropagation is done for a period of epochs whenever a genome is generated. It would be highly interesting to visualize the starting and ending points of genomes before and after backpropagation to see its effect on how genomes move through the space during backpropagation, and how much this leads to new better-performing genomes in the population.

## References


[1] Junghoon Chae, Catherine D. Schuman, Steven R. Young, J. Travis Johnston, Derek C. Rose, Robert M. Patton, and Thomas E. Potok. 2019. Visualization system for evolutionary neural networks for deep learning. In *2019 IEEE International Conference on Big Data (Big Data)*, 4498–4502. doi:10.1109/BigData47090.2019.9006470.

[2] Travis Desell. 2017. Large scale evolution of convolutional neural networks using volunteer computing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 127–128.

[3] E. Hart and P. Ross. 2001. Gavel - a new tool for genetic algorithm visualization. *IEEE Transactions on Evolutionary Computation*, 5, 4, 335–348. doi:10.1109/4235.942528.

[4] A. Kerren and T. Egger. 2005. Eavis: a visualization tool for evolutionary algorithms. In *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)*, 299–301. doi:10.1109/VLHCC.2005.33.

[5] Amit Dilip Kini, Swaraj Sambhaji Yadav, Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Co-evolving recurrent neural networks and their hyperparameters with simplex hyperparameter optimization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 1639–1647.

[6] Zimeng Lyu, AbdElRahman ElSaid, Joshua Karns, Mohamed Mkaouer, and Travis Desell. 2021. An experimental study of weight initialization and lamarckian inheritance on neuroevolution. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 584–600.

[7] Zimeng Lyu, Joshua Karns, AbdElRahman ElSaid, Mohamed Mkaouer, and Travis Desell. 2021. Improving distributed neuroevolution using island extinction and repopulation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 568–583.

[8] Aruanda Meiguins, Yuri Santos, Diego Santos, Bianchi Meiguins, and Jefferson Morais. 2019. Visual analysis scenarios for understanding evolutionary computational techniques' behavior. *Information*, 10, 3, 88. doi:10.3390/info10030088.

[9] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 2021. Search trajectory networks: a tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, 109, 107492. doi:10.1016/j.asoc.2021.107492.

[10] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO '19). ACM, Prague, Czech Republic, 446–455. ISBN: 978-1-4503-6111-8. doi:10.1145/3321707.3321795.

[11] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating recurrent neural network memory structures using neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, Prague, Czech Republic, 4–10. doi:10.1145/3321707.3321795.

[12] Hartmut Pohlheim. 2006. Multidimensional scaling for evolutionary algorithms— visualization of the path through search space and solution space using sammon mapping. *Artificial Life*, 12, 2, (Apr. 2006), 203–209. eprint: https://direct.mit.edu/artl/article-pdf/12/2/203/1662293/artl.2006.12.2.203.pdf. doi:10.1162/artl.2006.12.2.203.

[13] W.B. Shine and C.F. Eick. 1997. Visualizing the evolution of genetic algorithm search processes. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, 367–372. doi:10.1109/ICEC.1997.592337.

[14] Kenneth Sörensen. 2015. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22, 1, 3–18.

[15] Kenneth O. Stanley and Risto Miikkulainen. 2001. Evolving Neural Networks through Augmenting Topologies. Tech. rep. TR-AI-01-290. University of Texas at Austin, Department of Computer Sciences, Austin, TX, USA, (June 2001).

[16] Aditya Shankar Thakur, Akshar Bajrang Awari, Zimeng Lyu, and Travis Desell. 2023. Efficient neuroevolution using island repopulation and simplex hyperparameter optimization. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1837–1842.