NEUROEVOLUTION TRAJECTORY NETWORKS

illuminating the evolution of artificial neural networks

STEFANO SARTI

DEPARTMENT OF COMPUTING SCIENCE AND MATHEMATICS

Doctor of Philosophy

Division of Computing Science and Mathematics
University of Stirling

November 2023

DECLARATION

I hereby declare that this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text and bibliography.

I also declare that this dissertation (or any significant part of my dissertation) is not substantially the same as any that I have submitted, or that is being concurrently submitted, for a degree or diploma or other qualification at the University of Stirling or similar institution.

I was admitted as a research student in October 2019 and a candidate for the degree of Doctor of Philosophy in December 2020. This dissertation is a record of the work carried out at the University of Stirling between 2019 and 2023, under the supervision of Professor Gabriela Ochoa and Doctor Jason Adair.
I have read, and adhered to, the University's plagiarism policy, as detailed at:
`http://www.plagiarism.stir.ac.uk/`

Stirling, November 2023

Stefano Sarti

ABSTRACT

Neuroevolution is the discipline whereby Artificial Neural Networks (ANNs) are automatically generated using Evolutionary Computation (EC). This field began with the evolution of dense (shallow) neural networks for reinforcement learning task; neurocontrollers capable of evolving specific behaviours as required.

Since then, neuroevolution has been used to discover architectures and hyperparameters of Deep Neural Networks, in ways never before conceived by human experts, with many achieving state-of-the-art results. Similar to other types of Evolutionary Algorithms (EAs), there is a wide variety of neuroevolution algorithms constantly being introduced. However, there is a lack of effective tools to examine these systems and assess whether they share underlying principles.

This thesis proposes Neuroevolution Trajectory Networks (NTNs), an advanced visualisation tool that leverages complex networks to explore the intrinsic mechanisms inherent in the evolution of neural networks. In this research the tool was developed as a specialised version of Search Trajectory Networks, and it was particularly instantiated to illuminate the behaviour of algorithms navigating neuroevolution search spaces.

Throughout the progress, this technique has been progressively applied from systems of shallow network evolution, to deep neural networks. The examination has focused on explicit characteristics of neuroevolution system. Specifically, the learnings achieved highlighted the importance of understanding the role of recombination in neuroevolution, revealing critical inefficiencies that hinder overall algorithm performance. A relation between neurocontrollers' diversity and exploration exists, as topological structures can influence the behavioural characterisations and the diversity generation of different search strategies. Furthermore, our analytical tool has offered insights into the favoured dynamics of transfer learning paradigm in the deep neuroevolution of Convolutional Neural Networks; shedding light on promising avenues for further research and development.

All of the above have offered substantial evidence that this advanced tool can be regarded as a specialised observational technique to better understand the inner mechanics of neuroevolution and its specific components, beyond the assessment of accuracy and performance alone. This is done so that

collective efforts can be concentrated on aspects that can further enhance the evolution of neural networks.

Illuminating their search spaces can be seen as a first step to analysing neural network compositions.

ACKNOWLEDGMENTS

Embarking on a Ph.D. research is akin to setting sail into uncharted territories. Navigating this academic journey requires a reliable "crew", and the people chosen for this venture must be prepared to provide assistance throughout the arduous yet fulfilling process.

I am immensely grateful for my exceptional crew, whose unwavering support has been instrumental in making this challenging expedition feel like a success. I extend my heartfelt thanks to each of them for their contributions, transforming this academic endeavour into a truly memorable adventure.

Foremost, my sincere appreciation goes to my primary supervisor, Prof. Gabriela Ochoa. Their constant guidance, wealth of knowledge, and honest feedback have made me into the researcher I am today. I owe a significant portion of my development to their continual support from the very beginning.

I also extend my deepest gratitude to my second supervisor, Dr. Jason Adair, whose genuine advice and strong guidance have been indispensable in navigating the complexities of a research degree. Their assistance, defined by their kind and approachable manner, has profoundly influenced my academic path.

A special acknowledgment is reserved for my wife, Kate, whose constant support, kind advice, and enduring presence have shown strength throughout this lengthy project. Her care and understanding have made a significant difference, for which I express my deepest love and appreciation.

To my family, the unwavering harbour in a tempest and the radiant sunshine after, I send you heartfelt gratitude. Your constant presence during both joyous and challenging times has been a source of immense strength, and I am truly blessed to have you by my side. You have always believed in my potential, and never doubted me.

Lastly, my sincere thanks go to friends and colleagues who have added joy, laughter, constructive criticism, and countless happy memories to this Ph.D. journey. Your companionship has made the experience both fun and intellectually stimulating. This academic endeavour would not have been as enriching without your friendship.
I will treasure these memories for years to come.

to Laura and Massimo; for always believing in me.

LIST OF PUBLICATIONS

The chapters below are based on publications output during this Ph.D. research. Each row relates to the chapter and its relevant publication.

Chapter #	Publication
3	**Sarti, S.**, Ochoa, G., (2021), December. *A NEAT Visualisation of Neuroevolution Trajectories*. In: Castillo, P.A., Jiménez Laredo, J.L. (eds) Applications of Evolutionary Computation. EvoApplications 2021. Lecture Notes in Computer Science(), vol 12694. Springer, Cham. (*Best Paper Nomination, Best Student Paper Nomination*)
4	**Sarti, S.**, Adair, J., Ochoa, G. *Recombination and Novelty in Neuroevolution: A Visual Analysis*. SN COMPUT. SCI. **3**, 185 (2022)
5	**Sarti, S.**, Adair, J., Ochoa, G. (2022). *Neuroevolution Trajectory Networks of the Behaviour Space*. In: Jiménez Laredo, J.L., Hidalgo, J.I., Babaagba, K.O. (eds) Applications of Evolutionary Computation. EvoApplications 2022. Lecture Notes in Computer Science, vol 13224.
6	**Sarti, S.**, Lourenço, N., Adair, J., Machado, P., Ochoa, G. (2023). *Under the Hood of Transfer Learning for Deep Neuroevolution*. In: Correia, J., Smith, S., Qaddoura, R. (eds) Applications of Evolutionary Computation. EvoApplications 2023. Lecture Notes in Computer Science, vol 13989. Springer, Cham. (*Best Paper Award, Best Student Paper Award*)
6	**Stefano Sarti.** 2023. Neuroevolution Trajectory Networks: revealing the past of incrementally neuroevolved CNNs. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (GECCO '23 Companion). Association for Computing Machinery, New York, NY, USA, 41–42.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ANNs	Artificial Neural Networks	iii
EAs	Evolutionary Algorithms	iii
EC	Evolutionary Computation	iii
DPV	Double Pole Balancing with Velocities	xii
NTNs	Neuroevolution Trajectoty Networks	xi
STNs	Search Trajectory Networks	xi
GE	Grammatical Evolution	xi
NEAT	Neuroevolution of Augmenting Topologies	xi
CFG	Context-Free Grammar	xi
CNNs	Convolutional Neural Networks	3
DSGE	Dynamic Sturctured Grammatical Evolution	3
DNN	Deep Neural Networks	3
LONs	Local Optima Networks	4
BC	Behaviour Characterisation	6
Fast-DENSER	Fast-Deep Evolutionary Network Structured Representation	6
PSO	Particle Swarm Optimisation	10
DE	Differential Evolution	10
TSP	Travelling Salesman Problem	11
NAA	Network Architecture Analysis	15
TWEANN	Topology and Weight Evolving Artificial Neural Network	15
QD	Quality Diversity	20
DENSER	Deep Evolutionary Network Structured Representation	21
GP	Genetic Programming	21
SGE	Structured Grammatical Evolution	22
ES	Evolution Strategies	24
DAG	Directed Acyclic Graphs	25

MNIST	Modified National Institute of Standards and Technology database	26
SVHN	Street View House Numbers dataset	26
CIFAR10	Canadian Institute For Advanced Research dataset	26
Fashion-MNIST	Fashion Mixed National Institute of Standards and Technology	26
GAN	Generative Adversarial Networks	30
NSGA-II	Non-dominated Sorting Genetic Algorithm II	30
NSLC	Novelty Search with Local Competition	33
MAP-Elites	Multidimensional Archive of Phenotypic Elites	33
CPPNs	Compositional Pattern Producing Networks	35
HyperNEAT	Hypercube-based NEAT	35
CPPFs	Compositional Pattern Producing Functions	36
SGD	Stochastic Gradient Descent	36
sGLA	synaptic General Learning Abilities	38
sTLA	synaptic Transitive Learning Abilities	38
LSTM	Long Short-Term Memory	40
GA	Genetic Algorithm	46
PF	Partition Factor	54
XOR	eXclusive OR	58
RL	Reinforcement Learning	60
CSTNs	Compressed STNs	77
NAS	Neural Architecture Search	133

CHAPTER 1 — INTRODUCTION

1.1 OVERVIEW

1.1.1 *Modelling the world*

It is in our human instinct to have the tendency to visualise and model our world through various means; viewing it from different standpoints. In science, as well as in our daily lives, we use specific methods of observations to form the mental models of our understanding about the world that surrounds us; might this be the physical world or abstract dimensions. In cities, we have always developed panoramic viewpoints to grasp the landscape that lies beneath us. Geology and cartography use contour plots to model the terrain through the years. Meteorology uses radar mapping to understand and predict weather events in the atmosphere. Visualisation also happens in those very abstract domains that are invisible to the naked eye, like the particles crashing in the Large Hadron Collider of CERN (Geneva), reproduced by a 3-dimensional model.

Throughout history, we have seen this necessity of discovery through observation manifesting in many occurrences. Like the first glimpse of the Moon, Venus and the Sun with the invention of the telescope by Galileo Galilei in 1609, or Robert Hooke's discovery of biological cells in 1665 by looking at a slice of cork wood through his own compound microscope. More recently, in 1955, the first image of individual atoms by professor Erwin W. Müller and Kanwar Bahadur; the first image of deep space by the Hubble telescope in 2012; or the first glimpse of a distant black hole through the huge Event Horizon Telescope Collaboration in 2019 [13].

All the above are clear evidence that visualisations have been a form of scientific measurement tool, helping researchers since the beginning, even in the most abstract realms. It must have been an astonishing feeling when Benoit Mandelbrot was able to obtain the first high quality visualisation of the *Mandelbrot set* while working at IBM's Thomas J. Watson Research Center in 1980 (Figure 1.1). Something that was present in his mind and in mathematical formalism, actually took shape on paper, before his eyes, enabling this *set* to slowly change the world, finding applications in many fields [14, 15, 16].

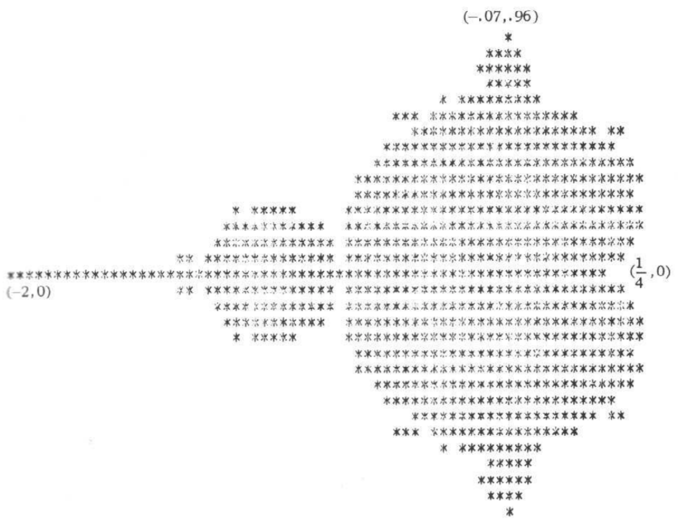


Figure 1.1: First computed visualisation of the Mandelbrot set [1].

In our research endeavours, the primary intentions have been to leverage a specific type of visualisation to explore the complex, multidimensional domain of *neuroevolution*. The exploration tool, which will be discussed in details later, leverages *complex networks* [17]. Our exploration takes place in the *search space*. The space of allowed and searchable solutions. Throughout our exploration of the search space we have aimed at identifying salient characteristics that distinguish specific neuroevolution algorithms. Furthermore, our intent has been towards the methodical exploration of particular traits of said neuroevolution algorithms, to further characterise our understanding of such systems. Amongst others, the works of [18, 19] suggested that there is a known difficulty in determining whether a given meta-heuristic (or evolutionary algorithm in general) is truly original, simply from a *performance analysis* standpoint — that is, relying on a fitness metric. It is why we argue that the visualisation technique, named NTNs [11], which we are proposing as a novel contribution to knowledge, is useful and will be useful in deciphering the complex domain of neuroevolution, through the lens of the search space. The particular traits that have been successfully analysed during this research follow a systematic flow of analysis. We progressed from the examination of neuroevolution algorithms that develop shallow artificial neural networks such as NEAT [2] and Novelty Search [3] to those that aim at generating deep networks (i.e. CNNs) with layered-based neuroevolution such as DENSER [20] and Fast-DENSER [5, 21]. The traits that have been examined during this research, using this technique include: the role of recombination in neuroevolution, diversity of evolved solutions, novelty search mechanisms, the topological complexity of ANNs and transferring knowledge in the incremental development of Convolutional Neural Networks (CNNs) [7, 10, 11, 22].

Two main fields of study had to be brought together to make this research possible. The evolution of artificial neural networks and the analysis of the search space.

1.1.2 *The necessity to illuminate neuroevolution*

Neuroevolution is the use of evolutionary algorithms to create ANNs; often this class of algorithms will focus on optimising either the parameters, the topologies, the learning rules or a mixture of these properties. Since its inception, many algorithms focused on tackling single or multiple of the aforementioned properties.

Neuroevolution was originally created as an efficient way to automatically design neural networks, as doing this manually would require a lot of efforts, through long trial and error processes. Initially, these artificial neural networks were devised mainly as solutions to reinforcement learning problems, as this practice could be applied more widely than other algorithms which require large sets of valid input-output pairs.

Most commonly, neuroevolution was originally used in domains such as evolutionary robotics, artificial life, and games playing. Since then its applications have expanded, and the algorithms have evolved to incorporate more established techniques from other disciplines, such as traditional machine learning and deep reinforcement learning; to improve these practices and hybridising variants. For instance, most neuroevolution algorithms have been upgraded to include means of back-propagation/gradient descent techniques to improve their training [23, 24]. This is considered fundamentally different from the original approach, where evolution is seen as the only optimisation engine that would provide the necessary learning rules.

A further development of evolving neural networks has been successfully achieved using GE and Dynamic Sturctured Grammatical Evolution (DSGE) to evolve Deep Neural Networks (DNN) layers and learning parameters using human-readable Context-Free Grammar [20]. This method has shown comparable and outperforming results to hand designed DNNs.

In neuroevolution, a variant of the NEAT algorithm, which employed the *Novelty Search* strategy was introduced to overcome deceptive fitness landscapes [3, 25, 26]. Another similar research strands emerged, independent from NEAT, aimed at providing a highly diverse variety of best performing neurocontrollers, which can be applied to dynamic tasks where seamless adaptation is required, these are known as Quality-Diversity algorithms [26, 27, 28, 29].

Amongst other innovations, the neuroevolution paradigm has been further developed, analogously to the evolution of learning, which occurred in biological brains [30, 31]. Through evolution, biological brains have developed *neuroplasticity*, which is fundamental for meta-learning and adaptation.

Neuroevolution can be considered a promising field offering strong analogies to how human intelligence was generated through evolution. It is evident that a lot of effort has be placed into this field, developing algorithmic variants in line with its principles. Particularly, NEAT has been considered a successful and influential algorithm, which attracted attention in the evolutionary computation community. This led to substantial contributions, creating many alternative variations of this system, outlined in a comprehensive review by [32].

Due the aforementioned complexity of NEAT and neuroevolution in general, the increasing need was perceived to develop methods and techniques that better examine the differences between variants, as well as the mechanics of operators used. In light of this, the decision was made to begin our research focus on TWEANNs, NEAT in particular and potential variations.

As described earlier, in this research we leveraged the successes of a complex network modelling technique called STNs [19], which is closely related and inspired by Local Optima Networks (LONs) [33, 34]. LONs is a modelling technique that aims to visually represent the inherent characteristics of *fitness landscapes*. Similarly, using these procedures STNs are capable of mapping the optimisation process of a meta-heuristic algorithm and its trajectories in the search space. This has shown evidence of being successful in many domains [19, 35, 36].

Our evident contribution is the extension of this technique and its direct application to the examination of neuroevolution dynamics. The development of NTNs has demonstrated successful results. In particular, our analysis of NEAT has highlighted inefficiencies concerned with the recombination (crossover) operator [37]. Furthermore, the relationship between recombination and the novelty search strategy from the prospective of the solutions in the search space was, for the first time, examined [10]. Additionally NTNs were used to assess topological complexities and behavioural diversity in novelty search compared to random and fitness search strategies; this was achieved by illuminating the behavioural space, instead of the search space of genotypic solutions [11]. Finally, NTNs have been fundamental in identifying salient characteristics in the evolution of CNNs and highlight the transfer of knowledge occurring in the incremental development of DSGE powered neuroevolution.

These research accomplishments have highlighted the necessity for neuroevolution to be more carefully examined and assessed using non-conventional tools. Because from the perspective of performance, some of these algorithms can appear to behave similarly, while others may behave differently, and it is not a sufficient metric to establish accurate comparisons. We believe the proposed visualisation instrument offers an effective way of examining this domain and its dynamics, which helps to provide plausible — perhaps not exhaustive — explanations into processes that otherwise would remain hidden.

1.2 RESEARCH HYPOTHESIS

Our formulated research hypothesis states that:

The application of STNs will result in a more detailed and informative exploration of the highly-dimensional Search Space of Neuroevolution, compared to traditional fitness performance methods.

1.3 CONTRIBUTIONS

This research has proposed three fundamental contributions to our field.

- Extended STNs to generate NTNs: a visualisation tool aimed at examining the intrinsic dynamics and peculiarities of neuroevolution
- Questioned the usefulness of the recombination operator in neuroevolution
- Confirmed that fitness performance metrics are not sufficient to characterise the fitness landscape properties of evolutionary algorithms for neural network development

1.4 THESIS STRUCTURE

In order to provide a structurally sound volume of work that can both be read sequentially and used as reference material with the table of contents, this thesis will be structured as follows:

- Chapter 1 — Introducing the area of research, the formulated research hypothesis and the major contributions of this thesis.

- Chapter 2 — An outline of the necessary preliminaries, which include the background knowledge pertinent to the field of research.
- Chapter 3 — A review of the literature relative to the growth of neuroevolution and advanced visualisation techniques.
- Chapter 4 — Detailing the use of STNs to study recombination in *shallow networks* generation for control tasks, using NEAT.
- Chapter 5 — A further STNs exploration of the role of recombination in NEAT for both novelty and fitness-based search, in deceptive maze navigation domains.
- Chapter 6 — The formulation of the NTNs visualisation tool for the study of Behaviour Characterisation (BC) for different divergent search strategies.
- Chapter 7 — Using NTNs to study the transfer of knowledge in incrementally developed *deep networks*, neuroevolved using Fast-Deep Evolutionary Network Structured Representation (Fast-DENSER).
- Chapter 8 — A research synopsis, overarching reflections, and future directions.

2

CHAPTER 2 — PRELIMINARIES AND BACKGROUND KNOWLEDGE

The following chapter will outline the fundamental preliminaries of this research. These include notions and axioms necessary to appreciate the contents of this thesis. As previously introduced, the research outlined in this dissertation, is a result of two main areas of study coming together: the illumination of the search space and fitness landscapes; and the generation of artificial neural networks through evolutionary computing. The concepts described are illustrated in Figure 2.1. We explain them in a progressive manner; starting from the foundations and origins, all the way to the proposed NTNs methodology.

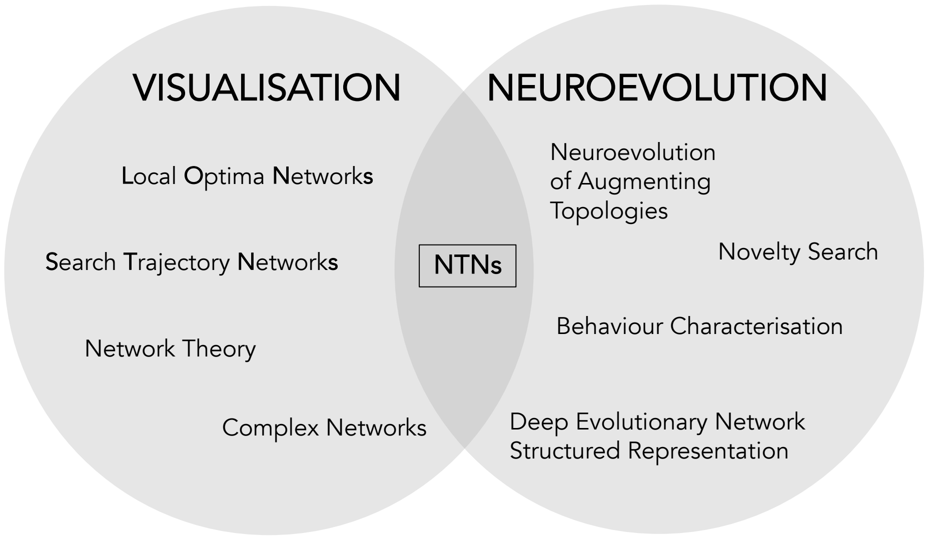

VISUALISATION
NEUROEVOLUTION
Local Optima Networks
Search Trajectory Networks
Network Theory
Complex Networks
NTNs
Neuroevolution of Augmenting Topologies
Novelty Search
Behaviour Characterisation
Deep Evolutionary Network Structured Representation

Figure 2.1: Venn diagram of NTNs methodology formation.

2.1 NETWORK THEORY

In mathematics and computer science, network theory belongs to the overarching discipline of *graph theory*. In a network graph, nodes are joined by edges and each of these components possess attributes. Network theory has many useful and diverse applications [38]. Networks lend themselves well to modelling various natural and artificial phenomena, allowing to form a simplified representation of the underlying structure of the modelled phenomena and propose a different perspective; which in turn may offer the tool to better understand them.

Graph theory originated from Leonhard Euler's paper on "Seven Bridges of Königsberg" which was published in 1736. In this manuscript Euler offers a solution to the Königsberg Bridge problem; a puzzle devised by the locals in which the goal was to walk around the city, crossing each of the seven bridges only once [39]. Although this was not strictly related to his mathematical fields of interest, Euler deemed this problem to be related to a topic that Gottfried Wilhelm Leibniz referred to as *geometria situs*: geometry of position. This "geometry of position" is what came to be known as graph theory, which Euler utilised for solving the problem.

The history of graph theory continued with the Hamiltonian cycles in Platonic graphs studied by William R. Hamilton and Thomas P. Kirkman [40]. Further studies and application were proposed by Gustav Kirchhoff, with the Kirchhoff's matrix tree theorem, where graphs are used to calculate spanning trees in electrical networks [41]. Graph theory has also been seen in the realms of chemistry, where Arthur Cayley and James Sylvester characterised chemical structures in mathematical terms by establishing that an isomorphism exists between the structures of individual chemical molecules, and mathematical graphs [42]. Worthy of mention is the work of Francis Guthrie and Auguste DeMorgan on "the four-color theorem", which was particularly influential in the establishment of this field. The work states that "*given any separation of a plane into contiguous regions, the regions can be coloured using at most four colours so that no two adjacent regions have the same colour*" [43].

Let us proceed to describe a graph. In mathematical formalism, a graph object can be defined as a set of *vertices* and *edges* as presented in Eq. 2.1.

$$G := (V, E, f) \tag{2.1}$$

Where V is a set of nodes or vertices. E denotes a set of elements which are known as edges or lines. While f is a function which maps the elements of E to and unordered pair of vertices V.

COMPLEX NETWORKS Complex networks are data-driven instantiations of graphs (networks) that present *non-trivial* structural features. These are considered features that do not occur in simple graphs (i.e. lattices). Often these are observable in networks that model real systems [17, 44]. Complex networks are found to be successfully applied in many fields spanning from biological, social and computer science.

Complex network share many of the characteristics explained earlier about graph theory. The elements used are nodes, defined as nodes pertaining a network object, $G(N)$, these are the vertices or points of the networks. Edges, on the other hand, are defined as $G(E)$ which are the lines or transitions between two or more vertices. Networks can be classed as *directed* if there is a flow, or orientation to the vertices, or otherwise undirected. Furthermore, other attributes of complex networks can be that edges are *weighted*, signifying the likelihood or occurrence of the connection. Vertices have similar attributes indicating the strength of incoming connections displayed by the proportional size of the nodes.

It is possible from the above definitions to appreciate how these generic and abstract properties can make the technique successfully applicable and utilisable for modelling various real-world and abstract systems, from the fields previously mentioned. Hence, why they are at the core of our proposed technique to illuminate neuroevolution.

One of the advantages of complex networks is that, aside from generating a visual illustration of a complex system or phenomena, which can incentivise the formulation of concrete mental models, this area of research has developed a multitude of *metrics* that can be used to generate a further comprehensive picture of the system in analysis. [44] provides a extensive taxonomy. During our research work, each line of investigation has utilised various different metrics, some have also been proposed by us as novel ways of analysing specific NTNs. These metrics will be discussed in details in each of the chapters. Amongst the most commonly used we find the *number of nodes*, the *number of edges*, and the *edge-to-node* ratio. The number of nodes metric goes to indicate how vast a network may be; often in our research this metric has been a powerful indicator of the diversity generated by a neuroevolution algorithm. The number of edges signify the degree of connectivity; while the edges-to-node metric offers a ratio on how many connection nodes may typically have. Another fundamental metric is the *degree* of nodes. The degree of a node is the number of incident edges, that is the number of connections that a node has to other nodes in the network. In a directed network, which is a network that has a specific orientation, there is an inbound (*in-degree*) or outbound (*out-degree*) nature to the edges of a node; these are often defined by an average across the whole network [45].

2.1.1 *Local Optima Networks*

It is important in this thesis to briefly detail LONs, which although different from STNs — the technique used to form NTNs — it is the predecessor of both of these techniques and a key piece of the foundations this research area.

Local Optima Networks (LONs) were successfully put forward in [46] where they were devised to study NK landscapes; a family of synthetic fitness functions which can be set to range from totally smooth to completely rugged. In their work, the authors found that through the perspective of LONs the landscapes exhibited "small worlds" traits, as the mean lengths of the network paths are seen to be short and scale logarithmically with the size of the networks.

Hence, we can define LONs to be a tool for the analysis of fitness landscapes, which is primarily dedicated at the detection of local optima and their connectivity patterns; this way formulating a visual model of the landscape. In the graph representing a LONs complex network, $LON = \{V, E\}$; the V denotes the *local optima*. Local optima and surrounding basins of attraction, are members that get transformed into the local optimum after local search. The E signifies the connectivity patterns between the V.

2.1.2 *Search Trajectory Networks*

While LONs are a tool dedicated to the detection and analysis of fitness landscapes and associated basins of attraction, STNs are devised to model the search process of meta-heuristic algorithms, occurring in the search space. STNs were initially proposed to model the optimisation search dynamics of population-based evolutionary algorithms such as Particle Swarm Optimisation (PSO) and Differential Evolution (DE) [47].

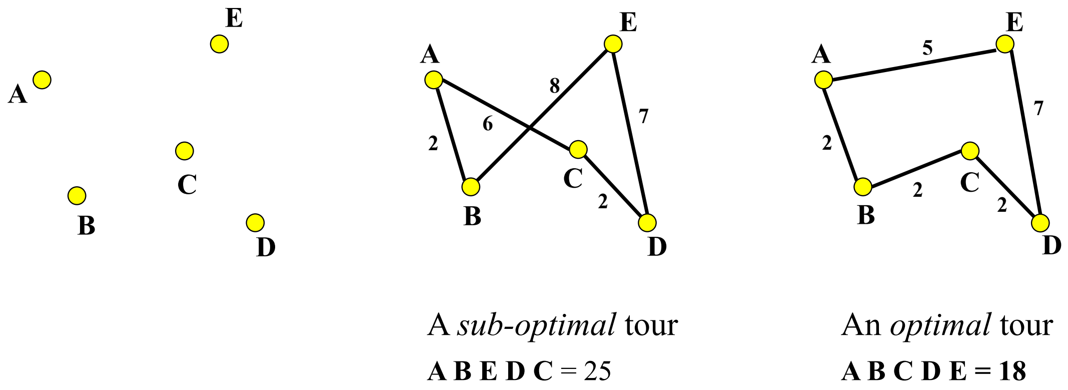


Figure 2.2: An example of an optimal and suboptimal solution for a small TSP instance.

Optimisation: To solve an optimisation problem, meta-heuristics operate by searching for objects in a large space of possibilities. These objects are referred to as candidate solutions, which are solutions that have the potential of solving the problem and the space is known as the *search space*. To give a concrete example, let us consider the Travelling Salesman Problem (TSP) problem. This rather abstract domain is useful in computer science as it has several practical applications (organising, routes planning, scheduling, and timetabling) [48]. The problem works by having a set of cities and the task is to find the shortest path (tour) to visit each city once. For an instance of the problem there are three components: inputs (the cities), the model (a formula) and outputs (the length of paths). In this case we are trying to optimise the path, to have the shortest tour around these cities. We want to search the space of possibilities (search space) to identify the candidate solution (a set of inputs) that produce a desired output; which is a sequence of cities with the optimal (minimal) path length. Therefore, this can be conceptualised as a minimisation problem. Figure 2.2 illustrates this problem well. We have a set of cities on the left, as our inputs, a suboptimal tour solution in the centre that yields a score of 25 and an optimal solution, which achieves a lower 18 tour length. The goal is therefore to identify the optimal solution amongst the space of possibilities.

Search Space: These desirable solutions reside in an enormous space of possibilities, which is known as the *search space*. Hence, problem-solving in this realm is seen as a systematic search process through a potentially vast space of plausible solutions. Mainly balancing between exploration and exploitation. These problems are also known as *search problems*. The definition of a search space provided by [49] is accurate and useful to elucidate this.

"[...] the collection of all objects of interest including the solution we are seeking."

This *collection of objects* is what it is used in the STNs approach to model the dynamics of an algorithm in analysis. Where vertices are the objects in the search space collection and edges are the transitions from these stages, forming the trajectories. Usually, if the algorithm in examination is a population-based algorithm, we select the best object of the collection found at each iteration, which is the best individual of the population at that stage of the search process.

The following are the definitions that constitute STNs [47].

REPRESENTATIVE SOLUTION is a solution from the collection of objects to the search problem, representing the status of the search algorithm. As discussed, population-based algorithms will have the best individual in the population as the representative for each iteration.

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, that is often the reduction in decimal points precision by an empirically chosen factor. The distinction between *solutions* and *locations* is required because in continuous (and discrete domains) the number of candidate solutions is, in principle, infinite. Therefore, we require a coarsening (partitioning) of the search space; essentially, clustering solutions to locations.

SEARCH TRAJECTORY Given a sequence of representative solutions in the order 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. This location is the unique signature that is used to construct the model.

NODE A location in a search trajectory of the search process being modelled. The vertices of the networks. The set of nodes is denoted by N. Nodes can be decorated based on specific attributes or computed *metrics*.

EDGES Edges are *directed* and connect two consecutive locations in the search trajectory. Edges can also be weighted and decorated like nodes (i.e. with the number of times a transition between two nodes occurred during the process). The set of edges is denoted by E.

SEARCH TRAJECTORY NETWORK Given the notions specified above, an STN is a *directed graph* defined as $\mathsf{STN} = G(N, E)$. The vertices are denoted by set N, and edges by set E.

EXEMPLIFICATION In the below illustration (Figure 2.3) we can see a practical example of what trajectories in the STN typically look like.

In all STNs trajectories there are nodes which are identified as starting locations; these coincide with the solution representative at the start of the search. We identify these as *starting nodes*. These are decorated with a specific colour (yellow in this case) and they are often represented by a different shape.

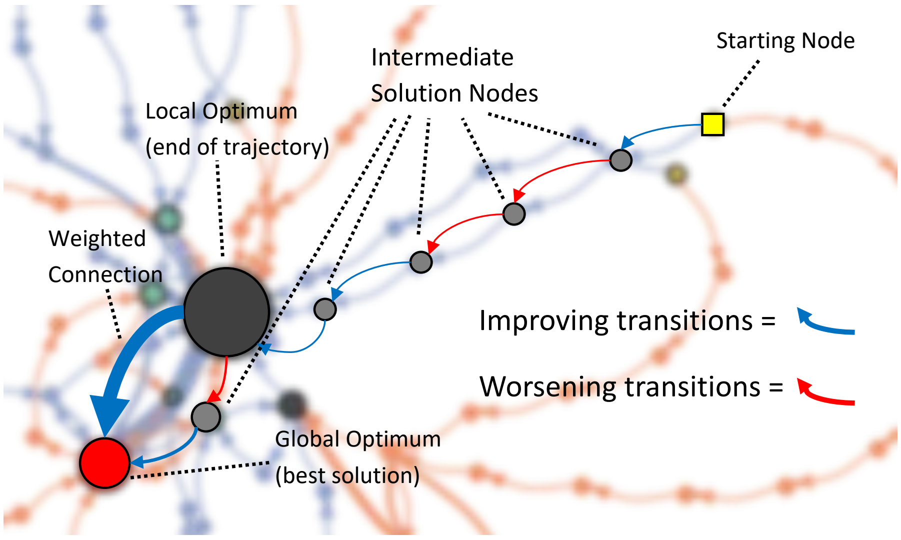


Figure 2.3: Exemplification of a typical trajectory in an STN formation.

When a representative solution becomes a location of interest in which the search process is, figuratively, “trapped”, this representative solution of interest could coincide with a local optima. This is often decorated with an enlargement of the vertex, proportional to the number of representative solutions that have visited the same signature (location) during the search process. Frequently, these are also found to be trajectories’ termination points.

From the illustration, we can see that edges can be *weighted* depending on the amount of transitions occurring from two locations (nodes). A further decorator could be a hue, given to signify certain attributes of the transitions. In this scenario, we see a colour is set for transitions that improve in score (i.e. fitness of solution) and another for those that worsen.

Often, multiple subsequent trajectories are expected to outbound a point of attraction, such as the case shown in the illustration. In other cases, (not illustrated here) there can be *loops*. A loop is an outbound trajectory that visits another location of the search space, only to return on the origin node. This is a typical behaviour where a solution is recurrently good but does not develop into a large trapping region — as defined previously.

Finally, in the illustration, (Figure 2.3) any nodes that are not part of the above classification are deemed intermediary or standard nodes. New standard nodes are recoded at every iteration of the algorithm, or at an empirically defined interval. These are unique progression stages visited during the algorithm’s traverse of the *space*. Furthermore, a node can be decorated of a specific colour to signify an attribute of the location, like in this case the red, which stands for a solution which meets the criteria of solving the problem, or reaching a satisfactory fitness threshold; in some cases this can coincide with the global optimum.

2.1.3 *Neuroevolution Trajectory Networks*

As the name of this technique suggests, this approach, which is the novel contribution proposed as the core of this thesis, is the union of two fields of study. The application of *Search Trajectory Networks* to the discipline of evolving artificial neural networks: *Neuroevolution*.

The definitions of this network modelling technique are largely unchanged from the parent STNs approach. What is different is the encoding of the representative solutions and the space from which solution are modelled. Similarly to STNs, the formalisation of this technique can be expressed as $\mathsf{NTN} = G(N, E)$. An NTN is formed by nodes $N(G)$ which satisfy $N \in nS$ where nS is the *neuroevolution search space*. In essence, NTNs are graph models formed only by elements found during search in the neuroevolution spaces. These elements are modelled empirically, selecting the method that best represents them. Differently from a canonical search space like that of combinatorial optimisation, the representative solutions discovered from nS require for a more sophisticated encoding to be applied. This is because the solutions (genotypes) are primarily related to neural network formations or derivations of such, and are not solely solutions formed with vectors of real numbers. To effectively collapse an ANN representation to a unique signature, used to form the vertex of NTN, it is not a generalisable approach; as different neuroevolution algorithms have disparate characteristics and encoding mechanisms. Nevertheless, the researcher utilising NTNs, thorough preliminary knowledge about the algorithm under examination and pertinent empirical tests, should be able to identify the correct approach to encapsulate all fundamental information in the node-edge representation, including the attributes.

The expected outcomes of a NTN study, on a given neuroevolution algorithm, is successful if the following principals are satisfied:

- The fundamental neural network characteristics composing the solution should be universally captured and represented in a concise, unique and reproducible signature. A differentiations of the solutions should produce a new signature record.

- The high dimensionality related to the solution should be reduced in a lossless manner.

- Any information that are fundamental or worth for the purpose of the investigation should be captured and mapped correctly, becoming key attributes of the network models.
- Decorators, on N and E (and related emergent dynamics), must be used effectively with these attributes, to highlight the neruroevolving nature of the algorithm and what the investigation is aiming to highlight.
- A visualisation plot should directly convey a fundamental message related to the presumed underlying neuroevolutionary mechanics.

In future chapters, it will become apparent how these rules have been satisfied. How these helped to form the NTNs technique and the Network Architecture Analysis (NAA) realm of investigation, which will be described in the concluding remarks of Chapter 8.

2.2 ALGORITHMS AND SEARCH STRATEGIES

The algorithms and notions of neuroevolution that were examined in this research are detailed as follows. Outlined in a chronological progression, from our first analysis to the last. This largely coincides with the logic of our proposed publications outlined at the start if this dissertation.

2.2.1 *NEAT*

NEAT stands for **N**euro**E**volution of **A**ugmenting **T**opologies, it is an established algorithm, that was proposed in the early days of neuroevolution [2]. The focus of it is to evolve shallow (dense) neural networks, initially for simple classification and control tasks. This algorithm was revolutionary at that time for certain innovative characteristics. First, it not only evolved the *structure* (topologies) of the neural networks — this involves the placements of neurons and connections — but was engineered to evolve the *weights* of said neural networks, through evolution. Hence why the algorithm falls under the Topology and Weight Evolving Artificial Neural Network (TWEANN) categorisation. This means that weights are learned without any form of *gradient-descent* or advanced learning paradigm. The algorithm achieved this by deploying novel operators and functions which we proceed to succinctly describe.

COMPLEXIFICATION This algorithm is engineered to produce solutions starting from simple neural networks and increasing their complexity, as needed, based on the problem. The system will always select the minimum optimal structure that can solve the problem amongst complexified ones found. This incremental growth in complexity is beneficial as it prevents complex solution from saturating the population/search in early generations.

PERMUTATION PROBLEM This is also described by authors of NEAT as the *competing conventions* [2] problem. A known issue, when more than one representation exists for expressing a solution with a neural network to a weight optimisation problem [50]. Essentially, this refers to the issue in which standard crossover of neural network solution may lead to offspring that do not include all inherited genes. One of the biggest claim of NEAT is that the algorithm can overcome this obstacle by encoding solution that leverage *historical markings*. This is also referred to as a gene innovation counter mechanisms, which helps to keep track of the evolved genes during neuroevolution, informing what to select for neural networks recombination.

GENETIC ENCODING NEAT uses a genotype to phenotype direct encoding framework. Nodes and connections are explicitly dictated in two levels of the genotype. These genes carry additional information, useful for the operators and the architecture of networks. These include the knowledge about inbound and outbound connections. The node number and type, the weight of connections, whether the connection is enabled or disabled, and the innovation number.

HISTORICAL MARKINGS This is the innovation number which is used by the algorithm to establish how nodes and connections should be preserved during the perturbations, especially while recombining solutions. These historical markings are used to produce offspring that ensure the inclusion of all the high performing information from the parents' genotypes.

SPECIATION Another feature that this neuroevolution algorithm introduced is shown in Figure 2.4. In this we can observe how the population has been effectively separated into niches. To do this the authors proposed to leverage the historical markings to calculate the genotypic distance between solutions in terms of their genetic material. In Eq. 2.2 we outline how this is achieved.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W} \tag{2.2}$$

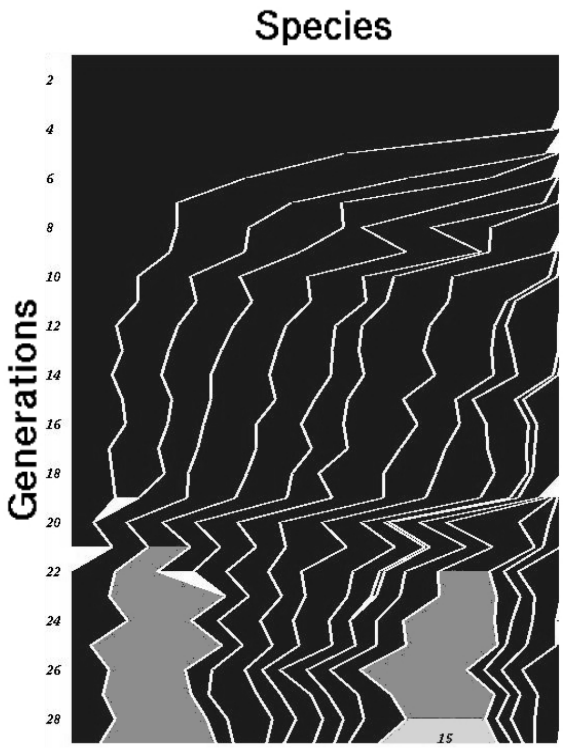


Figure 2.4: Speciation in NEAT. Sourced from [2].

The parameters c_1, c_2, c_3 are used empirically to adjust the importance of each factor in the equation: E, D, and W. These factors relate to the amount *matching* genes, which are those that are shared between two genotypes as per their innovation number. E denotes the *excess* genes. These are found outside the range of a given genotype (based on historical markings), therefore they are exceeding. On the other hand, D indicates those genes which occur within the range of a given genotype (based on historical markings) and therefore are deemed to be *disjoint*. W is an average measure of weight differences of matching genes; it is used to compute the final δ value as per the defined Eq. 2.2.

This value is used to separate solutions into niches. It is done to allow topologies of neural network to compete and survive long enough in their species, giving them the chance to fill their potential and optimise, in order to compete with stronger solutions. The authors engineered this, as it was assumed that some topological solution could be resilient stepping stones, which would lead to stronger and more optimised solutions, in further generations. The intuition arose from the biological world, in which diverse species tend to compete at different levels, hence allowing for interesting solutions to be protected from being eliminated by non-interspecies rivals [51].

It is evident from the many features outlined above that this algorithm has been packed with a multitude of operators and functions aimed at generating a system highly analogous to biology. Despite the efforts, all of these components will not work as cohesively and effectively as designed; an extensive systematic review of such algorithm has been proposed in [32]. This helps to identify potential inefficiencies. Some of which have been focus of this dissertation and that will be discussed in later sections.

2.2.2 *Novelty Search vs Objective Search*

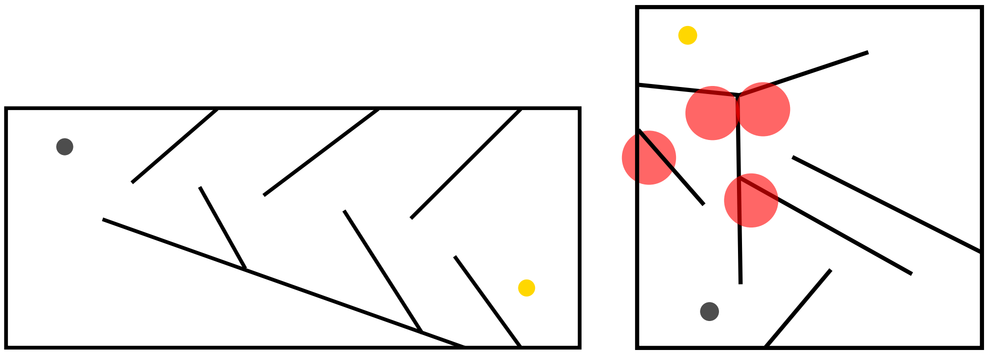

(a) Medium map (b) Hard map

Figure 2.5: The maze navigation problems used in Novelty Search [3].

Novelty Search is a search strategy first proposed in [3], which seeks to discover behaviours of the feature space that are novel and distant from those previously found during evolution. This was tailored to produce ANNs solutions that would be less prone to failing in deceptive fitness landscapes. The idea was initially applied to a maze domain (Figure 2.5), where a neuroevolved agent had to control its movements through the maze from a range of sensors, to reach the exit point.

The Medium and Hard Maze Problem: This reinforcement learning navigation domain has two types of mazes: a *medium maze* and a *hard maze*. Figure 2.5a illustrates the configuration of the medium maze. This is of low to intermediate difficulty as the map shows areas of low deception, which can be circumvented by the agent without great difficulty. The journey from the starting position (dark-grey dot) to the goal (yellow dot) is reasonably linear with a low chance of the agent getting trapped.

The hard maze has a deceiving structure. This is illustrated in Figure 2.5b where the map is visibly harder due the placement of the walls, which generate trapping regions (red shaded circles), capable of blocking the search progress of agents traversing the maze. These areas of high deception are what most challenge the neuroevolution search strategies.

OBJECTIVE SEARCH In this domain, the fitness function — also referred to as the objective function — measures the quality of an agent. This is calculated as its proximity to the goal at the end of the navigation task evaluation. Typically, evolutionary algorithms use a fitness function to guide the search. In [3] a variant of NEAT was guided by this fitness function. Let us describe it and compare this to the pursuit of Novelty.

$$\mathcal{L} = \sqrt{\sum_{i=1}^{2} (a_i - b_i)^2} \tag{2.3}$$

Equation 2.3 measures the Euclidean distance between the agent's final location in the maze and the exit point (the objective). Hence, $\mathcal{L}$ is the root-mean-squared error used for the evaluation. In the equation, a denotes the position of an agent at the end of the simulation and b the location of the goal — expressed as 2 dimensional spatial coordinates.

This is known as the *objective search strategy*. The objective is the sole driver of the search and dictates the neuroevolutionary process.

NOVELTY SEARCH This search strategy was proposed and discussed in various works [3, 25, 52, 53]. It was created to point out the benefits of abandoning objectives in the pursuit of novelty, specifically for deceiving domains, where the fitness gradient is not directly aligned with the notion of optimal performance. The idea behind this strategy is to adapt the objective function to leverage the novelty of an agent as a metric of performance. In NEAT — the neuroevolution engine used in this case — novelty can be implied as *behavioural* novelty, used in the seminal research of [3] or *structural* novelty, intended as the diversity of topological formations.

In Novelty Search, although it remains the goal of the maze, the intent is no longer that of aiming directly at achieving the closest proximity to the exit of the maze; it is swapped for the objective of having optimal agents that are able to exhibit diverse explorative behaviours. The scope is for the evolutionary process to reward those actions that yield agents' journeys to unexplored locations of the maze maps; to be capable of escaping the trapping regions and to ultimately reach the goal (yellow dot).

$$\text{dist}(x, \mu) = \frac{1}{n} \sum_{j=n}^{n} |x_j - \mu_j| \tag{2.4}$$

To achieve this, the performance of the neurocontrollers (agents) is calculated using a metric of sparseness. This is computed by the *k-nearest neighbours* algorithm. The implementation used for our analysis (Eq. 2.4) was derived from [9], which is faithful to the one deployed by [3].

Increased sparseness is therefore equivalent to higher *Novelty*. This is obtained by neurocontrollers with exploratory behaviours that lead to unexplored locations of the maze (in terms of Cartesian coordinates). The distance between the two trajectory vectors, one for each compared agent, is used to compute the metric. To do these comparisons correctly, historical novelty must be logged in a *novelty archive*. The items in the current population are compared to the novelty archive. This evaluation allows for novelty to be assessed, and subsequently introduced in said archive. A set of parameters control the criteria for acceptance and inclusion of items in a novelty archive.

In the Eq. 2.4 $\mathrm{dist}(x, \mu)$ is the novelty score achieved from evaluating the behavioural difference between two agents. This is calculated as the absolute distance between the two trajectory vectors x and μ. Trajectory vectors are traced by agents; bi-dimensional maze coordinates of size n. x_j and μ_j are the values of the compared vectors (x and μ) at position j. In [11], to simplify the calculation, only the agent's trial end coordinates ($j = n$) were considered as the coordinates of interest.

To sum up, Novelty Search is intended to generate behaviours that diverge from the gradient of fitness. As shown in the hard maze of Figure 2.5b, the regions of local optima highlighted in red would deceive the search process of an agent using only fitness as the strategy. Novelty in this case is designed to bypass this issue of deception, effectively circumventing these traps.

BEHAVIOUR CHARACTERISATION Behaviour characterisation will be used in the discussion of Chapter 6. The chapter outlines our research intentions to detect BCs and be able to appropriately evaluate these, based on rigorous testing. Our particular attention was pointed to a topological analysis. The complexity of a topology can often dictate the behaviour of an evolved agent. Our research aimed at elucidating what roles topologies play in BC and diversification.

In [54] we find a comprehensive review of Quality Diversity (QD) algorithms which focus on behavioural characterisation. The authors define BC to be the notion by which behaviours are classified. The classification are found to be either *aligned* or *unaligned* to the notion of fitness quality, that is determined by the assessment domain. In their research, nothing in between these two characterisations is proposed. They additionally identified that not all behaviours are equally important. A noteworthy finding that emerged from their study is that BCs which are aligned to the *objective* of the assessment domain lead to favourable diversity and high exploitative qualities to reach the domain goal.

Nevertheless, the counterargument [25, 26, 27] is that the evolution of diversity should stem from BC which are not directly aligned to quality itself. The non-directed explorations would in fact point towards divergent ways of finding larger varieties of optimal behaviours capabilities [55]. Recent works have also focused on allowing for an automated search and definition of BCs — also more widely known as novelty descriptors. This ensure for the algorithmic system to have an independent selection of the descriptors often unrelated to agents' behaviours [56].

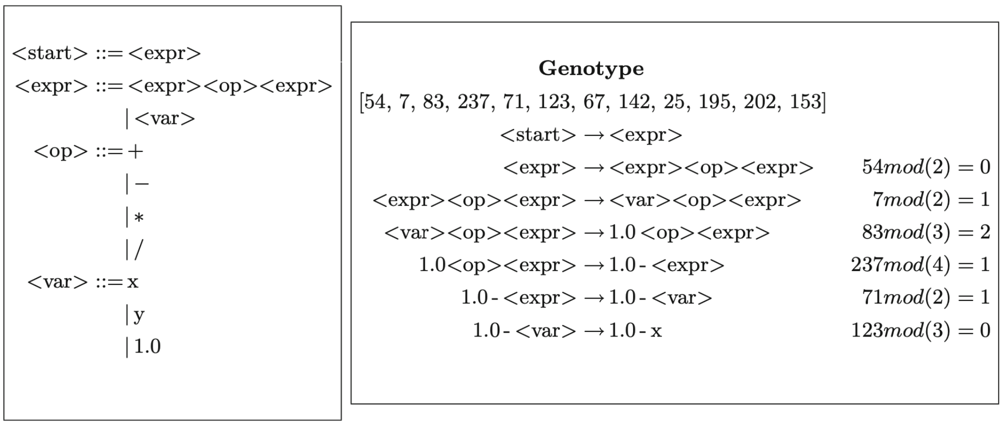
2.2.3 *DSGE-Powered Neuroevolution*

Presented next are a new set of neuroevolution algorithms, integral to the latest stages of this thesis investigation. We will outline two algorithms which were developed in close succession: Deep Evolutionary Network Structured Representation (DENSER) [57] and Fast-DENSER [21]. Both of these leverage a form of Grammatical Evolution (GE) used to effectively evolve ANNs.

Grammatical Evolution: GE is a type of evolutionary algorithm, specifically a grammar based Genetic Programming (GP), first proposed by [58]. In [4] we find a succinct and accurate definition, which goes as follows.

In GE individuals are presented as a variable length string of integers. An executable program is created by mapping the individual's genotype (string of integers) to a phenotype (program) through the production rules specified by the CFG. A grammar can be formalised as a tuple $G = (NT, T, S, P)$. NT and T denote the non-empty set of *Non-Terminal* (NT) and *Terminal* (T) symbols. S is the starting symbol; an element of NT called the axiom. P is the set of production rules, in the form $A ::= \alpha$, with $A \in NT$ and $\alpha \in (NT \cup T)^*$. A language $L(G) = \left\{ w : S \overset{*}{\Rightarrow} w, w \in T^* \right\}$ is defined by each grammar. The set of all sequences of T^* derivable from the axiom.

In the genotype to phenotype mapping of the original GE, selecting a derivation rule should replace the *Non-Terminals*. This is done by using the modulo operator. Figure 2.6 helps to elucidate this concept. The genotype comprises randomly generated values between 0 and 255. The process begins at $< start >$. This case shows that the only possible derivation is $< expr >$. The expansion possibilities of $< expr >$ are 2. To expand this we select the first value in the genotype, which is 54 and find the remainder of the division



```
<start> ::= <expr>
<expr> ::= <expr><op><expr>
         | <var>
<op> ::= +
       | −
       | *
       | /
<var> ::= x
        | y
        | 1.0
```

Genotype

[54, 7, 83, 237, 71, 123, 67, 142, 25, 195, 202, 153]

<start>	→ <expr>	
<expr>	→ <expr><op><expr>	$54 mod(2) = 0$
<expr><op><expr>	→ <var><op><expr>	$7 mod(2) = 1$
<var><op><expr>	→ 1.0 <op><expr>	$83 mod(3) = 2$
1.0<op><expr>	→ 1.0 - <expr>	$237 mod(4) = 1$
1.0 - <expr>	→ 1.0 - <var>	$71 mod(2) = 1$
1.0 - <var>	→ 1.0 - x	$123 mod(3) = 0$

Figure 2.6: Explanation of Grammatical Evolution (GE) with a simple grammar. Sourced from [4].

using the *modulo* operator. $54 \text{mod}(2) = 0$ is the result, which indicates that the first choice is selected: $< expr >< op >< expr >$. This process continues until all NT symbols are exhausted or there are no more integers in the genotype. The latter will force a wrapping mechanism to be executed were the genotype will be reused, until a valid phenotype is generated or the maximum wraps parameter is reached. If this is reached the solution will be declared invalid and a new genotype will be selected.

Dynamic Structured Grammatical Evolution: is a powerful subsequent form of the original algorithm, which aims to solve the known problems of low locality and high redundancy found in GE [59, 60]. Low locality, meaning that when a genotype undergoes one mutation, several phenotypic units are changed, resulting in great locality disruptions. High redundancy, in the sense that several genotypes correspond to the same phenotype.

DSGE derives from Structured Grammatical Evolution (SGE) [61]. The aim is to improve the genotypic representation, requiring no modulo operator to dictate the expansion possibilities, hence avoiding the redundancy associated with it. To achieve this, each gene is associated to a specific NT and an integers list is used to select the expansion option. Creating the list requires computing the maximum possible number of expansions of each Non-Terminal (NT). The bounded list of possibilities limits the number of changes that can occur at the phenotypic level.

The *Dynamic* SGE approach was devised mainly to address the criticism of the recursion problem. To prevent runaway bloat resulting from excessive tree size, the algorithm requires a maximum tree size of each Non-Terminals to be specified a priori. In this version, the solution is that the intermediate grammar derivation rules are created to mimic the recursion process. Additionally, this allows for the entirety of a genotype to be used, as this can grow as needed, for just the number of required derivation. This is achieved dynamically during evolution. Nonetheless, an upper limit has to be posed on the genotype size for this not to grow indefinitely, this is in the form of a maximum tree-depth parameter.

DENSER An initial neuroevolution proof was first approached in [62] and then fully formalised in [57] with the inception of DENSER. This algorithm specifically aims to evolve Deep Neural Networks, and it works by using two evolutionary algorithms (EAs) simultaneously, at different levels. In the outer-level we find a vanilla Genetics Algorithm (GA). While in the inner-level is where DSGE is deployed. In DENSER, the GA level encodes the macrostructure of the networks, which requires structures' allowance to be predefined. If we look at the development of CNNs, these require a specific CFG to be defined in Backus-Naur form.

To give an example of the allowance for a CNN, consider the following. An allowed GA structure would be:

$$[(\mathtt{features}, 1, 10), (\mathtt{classification}, 1, 2), (\mathtt{softmax}, 1, 1), (\mathtt{learning}, 1, 1)] \tag{2.5}$$

In the formalism of (2.5) each tuple is a valid *starting symbols* and the *minimum* and *maximum* number of times these can be used. Hence, if these ranges are added up, it would result that the GA is allowed to generate and search for structures of up to 10 convolutional/pooling layers (see features), followed by up to 2 fully-connected layers (see classification), and 1 classification layer (see softmax). Finally, the learning tuple is used to specify parameters related to the training and optimisation of the network in evolution.

On the other hand, the DSGE is leveraged to search the parameters inherent to the layers evolved from the macrostructure apparatus. Parameters and allowed values are preconfigured by the user in the grammar [21]. In Chapter 7 this particular research will be discussed further. In Figure 2.7 an example of the grammar used for the experiment is provided.

Using the GA, which encapsulates the genetic information for the layers, makes it easy for the variation operators to be applied (mutation and recombination). Most importantly, having the DSGE enables this technique to be generalisable to other problem domains. That is, by altering the grammar construct, one can explicitly redirect the search towards different network types.

This multi-EAs combination showed remarkable results for neuroevolution, all based on a grammar encoding [57].

```
        <features> ::= <convolution> | <convolution>
                       | <pooling> | <pooling>
                       | <dropout> | <batch-norm>
     <convolution> ::= layer:conv [num-filters,int,1,32,256] [filter-shape,int,1,2,5]
                       [stride,int,1,1,3] <padding> <activation> <bias>
      <batch-norm> ::=layer:batch-norm
         <pooling> ::= <pool-type> [kernel-size,int,1,2,5]
                       [stride,int,1,1,3] <padding>
       <pool-type> ::= layer:pool-avg | layer:pool-max
         <padding> ::= padding:same | padding:valid
  <classification> ::= <fully-connected> | <dropout>
 <fully-connected> ::= layer:fc <activation>
                       [num-units,int,1,128,2048 <bias>
         <dropout> ::=layer:dropput [rate,float,1,0,0.7]
      <activation> ::= act:linear | act:relu | act:sigmoid
            <bias> ::= bias:True | bias:False
         <softmax> ::= layer:fc act:softmax num-units:10 bias:True
        <learning> ::= <bp> <early-stop> [batch_size,int,1,50,500]
                       | <rmsprop> <early-stop> [batch_size,int,1,50,500]
                       | <adam> <early-stop> [batch_size,int,1,50,500]
              <bp> ::= learning:gradient-descent [lr,float,1,0.0001,0.1]
                       [momentum,float,1,0.68,0.99]
                       [decay,float,1,0.000001,0.001] <nesterov>
        <nesterov> ::= nesterov:True | nesterov:False
            <adam> ::= learning:adam [lr,float,1,0.0001,0.1] [beta1,float,1,0.5,1]
                       [beta2,float,1,0.5,1] [decay,float,1,0.000001,0.001]
         <rmsprop> ::= learning:rmsprop [lr,float,1,0.0001,0.1]
                       [rho,float,1,0.5,1] [decay,float,1,0.000001,0.001]
      <early-stop> ::= [early_stop,int,1,5,20]
```

Figure 2.7: An example of the CFG used in the research detailed in Chapter 7. Sourced from [5].

FAST-DENSER DENSER's long evolution and training process, due to its large population (100 individuals), made this technique unfeasible, despite its successes. In response, the authors decided to devise Fast-DENSER [21]; a faster algorithm, which is comparable in fitness performance to DENSER. This neuroevolution algorithm offered a 20x speed-up compared to its predecessor.

To achieve this, a $1 + \lambda$ Evolution Strategies (ES) was introduced in place of the GA. λ is set to generate four offspring from a single best parent at each generation. Doing so, the population was effectively reduced by 95% compared to DENSER. This drastic reduction corresponded to lower training and evaluation times, less computational resources. The running time was reduced to a fraction compared to DENSER.

Moreover, Fast-DENSER improves the flexibility of DENSER by allowing *skip connections*; layers of the evolved networks can connect to any previous layers. This is similar to the principle of Directed Acyclic Graphs (DAG). It also introduced limits on the number of previous layers that can be used as inputs (known as levels back constraint). This allows to avoid disjoint graphs, which could arise if the rule is not specified that layers always have to be connected to previous layers.

The initialisation of Fast-DENSER solutions is achieved in a gradual manner. A lower upper bound on the maximum number of layers is enforced (on the outer-level encapsulation) for the initial population. This ensures that the network is initially simple, deepening further through generations. This method, which is similarly used also in NEAT, is effective as it prevents excessively deep architectures form saturating the training progress early on.

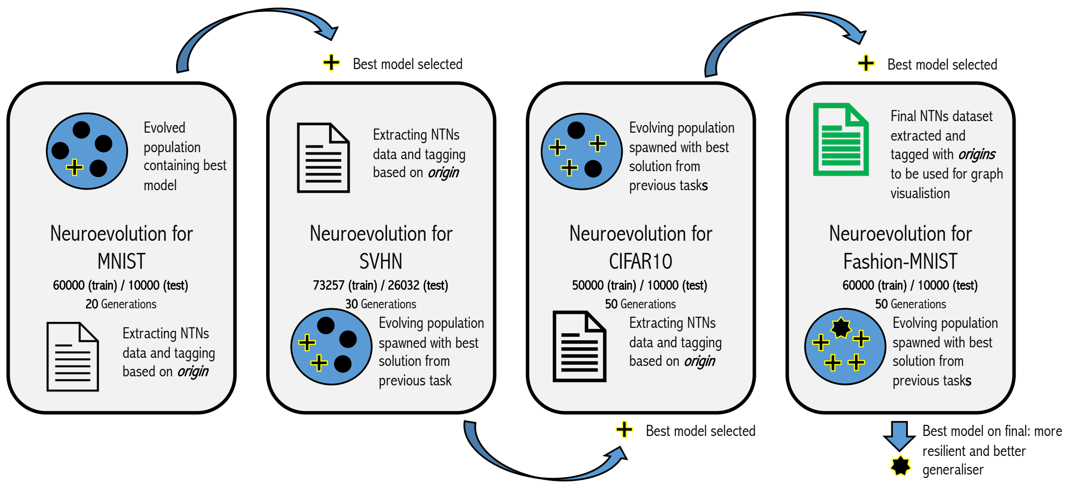
In Fast-DENSER, several types of genetic operators are included; notably *recombination* is not included in this version. This observation served to form part of our analysis, which will be outlined in Chapter 6. The operators used in the outer encapsulation level are: **add evolutionary unit**, **remove evolutionary unit**, **add connection**, **remove connection**. The inner-level mutation operators which are described in details in [57], alter real values/integers related to the grammatical expansion possibilities, as it is custom in grammatical evolution.

INCREMENTAL DEVELOPMENT/TRANSFER LEARNING For our analysis of Chapter 6 a specific variant was used known as incremental development, which is conceived to facilitate the transfer of knowledge acquired by the algorithm during neuroevolution. The concept of incremental development, specific to Fast-DENSER was introduced in [5]. The initial goal behind their research was to observe whether significant improvements could be gained from transferring evolutionary knowledge between DNNs trained on related classification problems. This was also aimed at reducing the number of generations necessary to reach comparative fitness values as those populations that were randomly initialised.

Furthermore, the authors were interested in examining if the knowledge gained from previous datasets worked, not just at initialisation but during other evolutionary stages. The consequent intention was demonstrating that incrementally developed solutions could be more resilient and better generalisers than solutions, which were not evolved incrementally on similar classification problems.

In [5] — the work on which our research examination of Chapter 6 was based on — the neuroevolution algorithm was configured with a specific grammar for the construction of CNNs; network architectures that have been found to be highly successful in image classification/object detection [63].

This variant of Fast-DENSER works such that at the end of each run, the best individual is selected from the final population, this is then used to spawn the new population for a subsequent similar benchmark problem. Four benchmarks for image classification/object detection were used in the following order: MNIST [64], SVHN [65], CIFAR10 [66] and Fashion-MNIST [67]. Modified National Institute of Standards and Technology database (MNIST) is a dataset of handwritten digits, which has 10 different classes of images (0-9), comprised of 60,000 training samples and 10,000 testing ones. Street View House Numbers dataset (SVHN) is a real-world image dataset of house numbers obtained from Google Street View, it incorporates over 600,000 digit images of 10 different classes (0-9); 73,257 digits for training, 26,032 for testing, and 531,131 extra ones. Canadian Institute For Advanced Research dataset (CIFAR10) consists of 60,000 tiny colour images split in 10 classes, with 6,000 images for each class; 50,000 are for training and 10,000 are for testing. Finally, The Fashion Mixed National Institute of Standards and Technology (Fashion-MNIST) dataset is comprised of 10 classes of greyscale images about fashion items, generated by the Zalando Research team. It is comprised of 60,000 training and 10,000 testing samples. The authors found that this incremental approach, where the knowledge about network architectures could be reused in later datasets, was able to produce faster convergence to the optimal fitness, due to this incorporation of prior learnings.



Best model selected
Best model selected
Evolved population containing best model
Neuroevolution for MNIST
60000 (train) / 10000 (test)
20 Generations
Extracting NTNs data and tagging based on *origin*
Extracting NTNs data and tagging based on *origin*
Neuroevolution for SVHN
73257 (train) / 26032 (test)
30 Generations
Evolving population spawned with best solution from previous task
Evolving population spawned with best solution from previous tasks
Neuroevolution for CIFAR10
50000 (train) / 10000 (test)
50 Generations
Extracting NTNs data and tagging based on *origin*
Final NTNs dataset extracted and tagged with *origins* to be used for graph visualistion
Neuroevolution for Fashion-MNIST
60000 (train) / 10000 (test)
50 Generations
Evolving population spawned with best solution from previous tasks
Best model selected
Best model on final: more resilient and better generaliser

Figure 2.8: Diagram illustrating the workings of Fast-DENSER with incremental development [6].

Figure 2.8 illustrates the Fast-DENSER workflow in details, a diagram extracted from the published research [6].

After a population is evolved through solving the first benchmark MNIST (a simple classification task), Fast-DENSER begins a new evolutionary search for a new domain SVHN. This new search is done by incorporating the best found model in the previous dataset (MNIST); the process continues for all datasets in the pipeline, until the final target dataset (Fashion-MNIST). Knowledge is incorporated at population inception but also during the intermediate search stages.

At inception, to generate individuals, the following four operations are allowed:

(i) include all feature extraction layers but randomly generate the classification layers

(ii) random generation of feature extraction layers but port classification layers from previous best models

(iii) transfer knowledge of learning evolutionary units but generate all other components from scratch

(iv) randomly initialise all evolutionary units at random, without incorporating any previous knowledge.

Perturbing solutions via mutation can leverage previous knowledge too. The operators are unchanged from those in [5], these are: addition, removal, and/or duplication of any evolutionary unit. In incremental development, this operator can also decide to adopt information of the best models from previous datasets or simply discard it and use the knowledge present in the evolution for the current dataset.

3

Chapter 3 — Literature Review

Similarly to Chapter 2, the content of this chapter will be structured by first reviewing the literature related to Neuroevolution, and a variety of the most prominent related forms that have risen during the years. Then the focus will shift to the literature relevant to the visualisation of fitness landscapes and search processes.

3.1 Early days of neuroevolution

The field of neuroevolution began to gain traction in the early 1980s where groups of researchers in evolutionary computing began to question if artificial brains, abstractions of biological ones, could be effectively generated through evolution, as occurs in nature [68, 69]. The motivation were bio-inspired and emerged from the remarkable phenomenon of the evolved human intelligence; alongside attempting to discover alternatives for canonic back-propagation approaches.

One of the first examples of neuroevolution algorithms emerged in 1994 as a simple optimiser of network weights [69]. Following this, the attention quickly shifted to developments that were not only able to evolve parameters but also the structure of these neural networks [70]. This was soon followed by a progression towards encodings that would allow to manipulate networks as graphs [71]. Other types of encodings were also considered [72, 73, 74, 75], namely *indirect encoding* was a technique by which a function expresses the network to be generated, rather then a direct description of weights.

Transitioning away from just evolving fixed topology neural networks was followed by novel requirements as well as challenges. Effectively crossing over different topologies and preserving information was critical. Alongside protecting neural network structures from extinction, long enough so that the weights could get updated, expressing their maximum capabilities. One of the early most successful approaches was NEAT [2]; an evolutionary algorithm based on direct encoding that explicitly mapped genotypes to phenotypes, labelling genes through historical markings and protecting variation of solution using a speciation operator, hence solving the aforementioned problems.

Amongst the successes of this early algorithm we find: dynamically evolving agents and content for video games [76, 77], generating complex musical compositions [78], evolving reaction networks in synthetic biochemical systems [79], prediction in geosciences [80], generating trading signals for financial markets [81]; even estimating the measurement of the top quark from the Tevatron particle collider [82]. A comprehensive and well-structured review of the advancement of NEAT-derived algorithm was composed by [32], offering a meticulous categorisation and a detailed comparison of each variant.

3.2 RECOMBINATION IN NEAT

Recombination has been a primary focus of the research outlined in this thesis. Specifically, the recombination operator proposed by NEAT, which has a particular characteristic that has often been deemed interesting and capable of evolving increasingly complex topologies. Thanks to the introduction of historical markings, the crossover operator generates valid offspring by comparing identifiable regions of compatibility or incompatibility, much like masking. All components and operators in NEAT were extensively examined via ablation testing in [2] and each was deemed essential for the performance of NEAT [83]. These tests generated ablated versions of the original algorithm, a variant without complexification capabilities (no-growth), another with no speciation, a further one that initialised genotypes at random, and the one we paid attention to is the non-mating variant. From these tests, the main insight was that this last version, without crossover, was able to reach a solution and converge to a high fitness threshold value, significantly quicker than other operators removed through the ablation tests. Suggesting that recombination contributes comparatively less than other operators to the performance of the algorithm as a whole.

The ablation tests were also reproduced on a derivative version of NEAT known as odNEAT [84], which is a decentralised variant created for online learning within groups of autonomous robots. These experiments showed that deactivating crossover resulted in an insignificant reduction in the number of successful runs. Additionally, the average number of evaluations of each robot increased by 18.9%. This finding was in support of recombination for NEAT. In [85], further results derived from odNEAT, on crossover, have shown that ablating this operator would have a worse impact than removing speciation.

Further studies, compared NEAT to a neuroevolution system based on reinforcement learning (EANT2) [86], where authors use neuroevolution to evolve and generate the structures of ANNs through reinforcement learning, applying it to a specific visual surveying scenario. EANT2 is not directly derived from NEAT, although the operators are inspired by it. In this research, the authors were not capable of developing a successful variant of crossover that would provide additional performance to the algorithm.

Another algorithm was proposed in [87] to evolve and optimise the architectures of CNNs. The architecture of the CNNs are encoded as a graph with each of the vertices representing a rank-3 tensor. Two of them are used for the spatial coordinates of the underlying image, and the third for the number of channels. Similarly as in [86] the operators are NEAT inspired. In this system the types of mutations were proved to be highly successful, on the other hand several variants of recombination were unable to improve it and were since then abandoned.

Recently, NEAT was adapted to be used to evolve DNNs [88]. The proposed algorithm (COEGAN) was derived from the NEAT extension known as DeepNEAT [89]. This algorithm is generated from a combination of neuroevolution and co-evolution for the training of Generative Adversarial Networks (GAN). In the study the authors used both sexual and asexual reproduction to generate valid offspring, and discovered that recombination led to an immediate saturation of the number of layers and premature convergence. This result discouraged further use of this operator.

In [27] a variation of Non-dominated Sorting Genetic Algorithm II (NSGA-II) [90] was proposed in which the direct encoding of NEAT is adapted for evolving neural network topologies. The crossover operator was excluded without any impact on performance, achieving instead state-of-the-art results. The extensive systematic review compiled by [32] on the progressions and algorithmic variations derived from NEAT, highlighted the limited benefits of certain operators such as crossover; the authors advocate revisiting those original ablation tests proposed in [83]. As observed there are strong contrasting views on recombination for neuroevolution. Our novel intuition is that the proposed thesis sheds light on this operator, offering some plausible truth on its usefulness or not.

3.3 SCALING UP TO DEEP LEARNING

Since its inception, neuroevolution has greatly evolved due to the benefits brought by improved hardware (parallelisation and GPU acceleration). When largely the same principles created in those days are coupled with the computational resources available now, we witness speed-ups, improved accuracy and vast scaling possibilities, not just in neuroevolution but also other disciplines of the larger AI field [91]. Therefore, it was a logical consequence for neuroevolution to benefit from this, by adapting the paradigms to deal with scaling and benefiting from vast parallelisation. This transition gives neuroevolution the chance to compete with DNNs for reinforcement learning, as well as architecture search, as an alternative to gradient-descent, or advanced learning optimisers. Neuroevolution is particularly suited to reinforcement learning problems; domains where reward is infrequently assigned, and the evolved agents have to progressively learn from sparse feedback.

An evolutionary algorithm developed by [92] was in fact able to produce comparative results as those of deep learning. This algorithm was able to directly evolve the weights of a DNN of the same size as the one presented in [93, 94]. The success of this evolutionary approach was remarkable due to the high dimensional parameter spaces to be searched. Although in this particular case, it was not conclusive to the argument that non-gradient based evolutionary algorithm could compete at the DNN level, as this algorithm does in fact follow a gradient, after estimating it.

Nonetheless, in [95] true *deep neuroevolution* had its early successes when a simple genetic algorithm (gradient free) was deemed a competitive alternative to deep reinforcement learning. Furthermore, this genetic algorithm was progressed to even outperform Rainbow [96]; an experiment where six extensions of the DQN algorithm were empirically studied alongside their combination. This provided state-of-the-art results on the Atari 2600 reinforcement learning benchmark, in terms of data efficiency and final performance. Ablation studies were also conducted to see which component of the ensemble approach contributed the most to overall performance. As the GAs developed in [92, 95] are population based, given enough computational power, they are extensively parallelisable, offering far greater performance in terms of speed to solution (in terms of wall clock).

Generating hybrid algorithms, by combining gradient based deep learning and neuroevolution, gained interest as a viable source of high performing models. An early example of this was presented in [97] where the authors proposed *safe mutations* (through output gradients). An algorithm for deep and recurrent neural networks based on the intuition that it is often expensive to run a simulation to evaluate a policy, either in a video game or engineering/physics context; on the other hand, it is fairly simple to instead evaluate the outputs of a neural network by performing propagations on reference instances. In this approach, neuroevolution allows to perform random mutations to the mapping from inputs to outputs of a DNN (policy). These mutations can have either a huge or minimal effect to the behaviour produced. By keeping a state and behaviours referencing system of the policies, safe mutations allows, through a gradient-based mechanism to scale the magnitude of the mutations hence making meaningful changes to the policy, towards the required direction.

Other successful forms of hybridisation include; executing variants of gradient-based reinforcement learning for crossover and mutation operators, used in a neuroevolution algorithm [98]. Furthermore, in [20, 21] the authors demonstrated a novel neuroevolution method to evolve DNNs based on GE, namely DSGE. In this approach the automated design is derived from a multilevel representation, an outer level encodes the general structure of the network (for instance the sequence of layers) and the inner level encodes the parameters associated with each layer. Layers and hyper-parameters range values are defined and expressed through a predefined *human-readable Context-Free Grammar*. These hybrid techniques for architecture and parameter search have shown promising, comparative, as well as state-of-the-art results on classification benchmarks such as MNIST, Fashion-MNIST, and CIFAR-100 (described in Chapter 2.2.3). The authors were also able to extend the Fast-DENSER algorithm to include transfer learning via an incremental development method [5] (also described in Chapter 2.2.3).

3.4 THE SEARCH FOR NOVELTY AND INCREASED DIVERSITY

It is widely appreciated that diversity is a driver for spontaneous innovation [91, 99, 100]. This could be the same innovation engine that allowed human-level intelligence to arise from evolution. Relying on convergent search strategies is often the cause of stifling diversity. Optimisation tends to exclude sub-performing solutions in favour of following a gradient towards the global best. Often global optima are hard to discover as the fitness landscape can be deceptively configured depending on the domain; in addition to using a defective objective function.

The vast majority of neuroevolution algorithms are population-based, this is an advantageous trait for diversity preservation. Operating on a population of solutions makes this field innately predisposed to parallel exploration of diverse solutions. Generally, techniques try to push the search away from attraction basins; for instance in NEAT, *explicit fitness sharing* is used to partition the population into species based on the genetic distance but also to penalise the number of individuals belonging to the fitness cluster [2]. This is known as the genetic space diversity or parameter space diversity. Despite the efforts, the diversity generated by these operators is often insufficient to produce vastly different behaviours. This is because there are multiple ways in which a neural network can be configured, which would instantiate the same behaviours. This creates an issue where genetic diversity is insufficient as certain problem domains are deceiving in their fitness landscape and will require behavioural diversity [25, 27, 101]. Aside from being successful in small evolved networks; it has been shown that this approach is also well performing on high-dimensional reinforcement learning problems [95, 102].

Due to this, a lot of attention has been placed to create algorithms capable of evolving solutions producing high behavioural diversity. Several desirable properties have emerged from these kind of search strategies. Amongst some we find surprise search, evolvability search, intra-life and across-training novelty [103, 104, 105]. This research further expands into developing neuroevolution systems that strike a trade off between high diversity and high fitness. These are known as *quality-diversity* algorithms. This concept is inspired by the notion that evolution in biology is not an optimiser that favours a particular solution or configuration but instead provides high performing alternatives which possess their own characteristics. One of the earlier exemplars, based on NEAT, was Novelty Search with Local Competition (NSLC). This strategy leverages the Pareto front of the multi-objective NSGAII algorithm to determine how to generate, via novelty search [25], solutions that are both highly performing and highly diverse [26]. Similarly, another algorithm of this category is Multidimensional Archive of Phenotypic Elites (MAP-Elites); this is a simple and powerful algorithm which discretises the behaviour space and allocates the highest performing solution to these niches. Competition can happen only within these niches, whereas a perturbation in one niche can generate a new champion in another niche. This principle shows that a high performing solution in one discrete section of the behaviour space can be generated as a consequence of the stepping stones in other niches of the space. This simple yet extremely powerful diversity enforcing algorithm produced remarkable results on a robot locomotion task, which required adaptation after being damaged, making the cover of Nature journal [106].

Aside from these remarkable results, quality-diversity is a research topic that continues to attract interest and where new discoveries are constantly being made. For instance, encouraging modularity in both the genotype and phenotype are an important step towards solving large-scale multimodal problems [107]. The Innovation Engine, which built on top of Novelty Search replaces the human-dictated behavioural distances with a Deep Neural Network capable of recognising interesting differences between phenotypes [108]. Moreover, extending Novelty Search to run in parallel on multiple islands, enabling to scale the neuroevolution of diversity to much larger populations and more diverse runs; doing so to harness much more computing power [109]. Another work was inspired by the simple notion of open-endedness in biology. This work investigates the extent to which interactions between two co-evolving populations — subject to respective constraints — is able to produce functional and diverse results, without any behaviour characterisation nor novelty archive [110]. Furthermore, other evidence of this is provided by the investigation which confirms that BCs which are created from hand-designed features are limited by human expertise and cannot exploit domain nuances. The same work shows that this can be addressed by recording generic behaviours on several evolutionary training tasks, and that new BCs can be learned from that. This forces and distils evolution towards successful behaviours on further tasks from similar domains [55].

The research conducted by [111] offers interesting insights and critiques of this discipline. The authors emphasise the significant challenge: evolving topologies for Novelty Search may pose complexities disproportionate to the achievable outcomes. Their studies followed the intuitive decision to test whether evolving topologies, compared to fixed topologies, can help or hinder the performance of this algorithm. These were tested on two benchmarks: 2D mazes, akin to those discussed in Chapter 2.2.2, and 3D walkers. Their result demonstrated that evolving topologies does not consistently help. They concluded that this approach failed to facilitate search initialisation or identify the optimal topological structure efficiently. Ultimately, their findings suggest that Novelty Search might not benefit significantly from integration with neuroevolution.

3.5 THE INDIRECT ENCODING PARADIGM

A research theme in neuroevolution that has gained a lot of traction and interest from the research community, concerns the mapping between genotypes and phenotypes, known as *encoding*. As with most aspects of evolutionary computing, indirect encoding is inspired by how information are expressed in our DNA. With *direct* encoding each bit of information in the genome expresses a specific individual characteristic in the resulting phenotype (making it a one to one mapping). On the other hand, similarly as it happens in our genetic code, *indirect* encoding allows for a piece of information to be expressed in multiple ways, in the phenotype. In such way offering a compressed and scalable method of determining placement, conformation and parameters of evolved neural networks, in essence, reusing information multiple times.

This notion is crucial, as a computational approach capable of evolving and reaching solutions up to the magnitude of our biological brain, requires an encoding mechanism that is efficient and computationally inexpensive. Hence, information reusability may allow for exponential growth and complexification from a simple encoding mechanism.

Indirect encoding not only fosters scalable ANN production through compressed information, its inherent mechanisms is able to generate patterns of regularity in the weights composition. Alan Turing was fascinated about this concept, described as *morphogenesis*, in his work on reaction-diffusion equations in patterns generation [112]. This technique allows for genes — which are information parameters — to indirectly map to units of structures in the final realised entity: the *phenotype*.

We see that this concept of regularities in ANNs and DNNs have been successful, especially in the notion of *convolution*; this is the regular connectivity pattern, where a detector of features is located in multiple areas of the same layer. The concept of evolvable indirect encodings to promote regularities has been studied profusely [113, 114, 115].

A successful indirect encoder was developed in [116]; this method uses Compositional Pattern Producing Networks (CPPNs) and it is founded on the morphogenesis principles mentioned before. These can be explained as simple networks of function compositions, represented as graphs. Said graphs are evolved, traditionally using NEAT [2], to form the Hypercube-based NEAT (HyperNEAT) [117]. The simple NEAT algorithm evolved CPPNs (encoders) that can vary in dimensions and complexity; these graphs are comprised of functions yielding complex patters. The array of functions used span from Gaussian for symmetric patterns, sigmoid for asymmetric ones, and sine wave to produce periodical segmentation. Achieving complex patterns of regularities is already possible by composing simple networks comprised of these functions. In [118] it has been shown that this compressed encoding mechanism can efficiently generate ANNs with massive connectivity patterns from very minimal CPPNs encoders.

In HyperNEAT [118] the evolved CPPNs encoders are used to generate the weights belonging to a fixed neural network. They do so by querying the spatial coordinates of a *substrate*, which can represent the given task or the structure of the agent for said task. This signifies that the output from CPPNs inherently incorporates weight patters, geometrical information about the problem domain. This inclusion is important in problems such as robot locomotion, which requires strong affinities between inputs and outputs signals of the agent. The success of this has been shown in [119] where the positioning of the sensors and actuators of a quadruped can be enhanced by evolving regular gait patterns. Similarly, [120] shows the possibility of developing ANNs of modular and repeated regular architectures.

In the work by [121] the utilisation of the HyperNEAT paradigm is seen successfully deployed for evolutionary robotics. Based on evidence showing that it is feasible to autonomously generate evolved designs in the physical domain; the authors specifically highlight the challenges posed by autonomous manufacturing and assembly processes, which introduce limits not observed during simulation. They successfully apply the paradigm to create repertoires of diverse and feasible robot designs.These serve as seeds within evolutionary loops, facilitating the evolution of designs and controllers capable of successfully solving maze-navigation tasks. Incorporating prior knowledge of a diverse and manufacturable population in this search has helped in faster convergence on specific tasks while ensuring the validity of manufactured designs [122].

This technique has been studied at length and has produced successful results; especially when used for image generation. It has been shown that these creations, aside from being remarkable on an artistic level, they are able to trick even highly accurate DNNs [123]. Furthermore, they have been used in DNNs to improve some of the limitations of convolution, by providing them with inputs from a space of coordinates [124]. Through evolution, they helped to shape the research on understanding engines for innovation to automate creativity [108].

A variant of this algorithm was proposed in [125] where the authors demonstrate that Compositional Pattern Producing Functions (CPPFs) can efficiently be evolved, instead of CPPNs, by a vanilla GP algorithm. The research offers remarkable results that outperform those of CPPNs. HyperNEAT has also been extended in [24] by an algorithm known as *hypernetworks*. In this approach, the original algorithm that evolves CPPNs is adapted to train the evolved weights by Stochastic Gradient Descent (SGD).

The increasing availability of high computing power is giving traction to this field by enabling efficient encoding of highly complex, as well as deep architectures. This include variants of HyperNEAT such as ES-HyperNEAT [126], where the original algorithm is augmented to not only evolve CPPNs capable of indirectly encoding patterns of weights but to also evolve the substrate topologies. This eliminates the manual component of creating a substrate structure and therefore, the possible human inference associated with it. This algorithm despite evolving the location of every neuron, it can also constitute regions of varying density, increasing substrate resolution holistically over iterations.

As it will be presented in the following section, most of these indirect encoding algorithms involving CPPNs, have been adapted to encode similar neuroplasticity rules, as seen in biological neural networks.

3.6 NEUROPLASTICITY AND META-LEARNING

The study of plasticity rules encoding has its application primarily in the pursuit of adaptable systems. These are resilient algorithms that attempt to generalise to different domains, after minimal training instances. The principal application witnessed in the literature up to this date is related to controllers in robotics, such as robotic locomotion tasks.

The idea of meta-learning or "learning to learn" is seen as a natural step towards Artificial General Intelligence. In biological neural system the properties of plasticity are considered essential for the learning and development of the forms of life we know. Adaptable ANNs are seen as dynamic and have the potential generalisation qualities that allow their application to span to a variety of domains. A recent survey by [31] offers great details on this area of research.

Specifically, the following sources use CPPNs for indirect encoding. Adaptive HyperNEAT [30] extends HyperNEAT, in which CPPNs are adapted to include, not only spatial coordinates of the substrate, but also pre and post-synaptic activity signals. Making it a system of plasticity patterns encoding. In [30] the authors test indirect encoding of plasticity rules through an operant conditioning task. An agent has to navigate a T-shaped maze to find high rewards located at either side of the maze. When the high reward switches location, the neural network encoded with plasticity rules is expected to adapt and explore the other side of the maze.

An extended version of this is Adaptive ES-HyperNEAT [127]. This algorithmic system augments ES-HyperNEAT to include further information during neural network encoding. It holistically evolves the topologies of the substrate based on geometrical information derived from the substrates, and it introduces regions of the neural network populated with standard and neuromodulated synapses, (weights) dependent on the outputs derived from the single instance queried CPPN. In the research of [127], the adaptive behaviour gained by indirectly encoding plasticity rules, is extended by using ES-HyperNEAT. A system that helps to encode the placement, density, and connectivity of neurons. Neuromodulated plasticity is encoded using this algorithmic system and performance is once again tested on the same navigation task, the T-maze domain [30].

The following research studies relate to further applications and algorithms, formulated on the basis of neuroplasticity encoding. The early work of [128, 129] investigated the possibility to evolve local learning rules, so that the weights of connections in the ANN would be modulated by the activation signals from both source and target nodes. This approach attempts to mimic the properties of the Hebbian rule in biological brains [130].

In [131] we can witness the initial studies of indirect encoding through NEON. A system which adopts NEAT to incrementally encode characteristics of artificial ontogeny, at different phases of the evolutionary search process. In this approach, the algorithm is tested against memory and control benchmark domains.

The work of [132] attempts to further investigate the workings of the aforementioned systems by isolating the adaptive component and testing the effectiveness in simpler domains. This analysis helped to understand these system better and the limitations they have in more complex environments.

The research of [133] explore how different plasticity encoding mechanisms are responsible for higher and more adaptable learning abilities. The authors report that by offering mechanisms of regularity, developmental (indirect) encoding is the reason for improved performance in simple operant conditioning tasks. The same authors in [134] highlight the importance of evolving adaptable plastic neural networks. This research explores key concepts necessary to understand synaptic General Learning Abilities (sGLA) and synaptic Transitive Learning Abilities (sTLA). This work offers fundamental definitions and metrics for being able to assess generalisation capabilities in evolved ANNs of replicated neuroplasticity.

Research efforts by [135] attempt to solve plasticity-stability problems by implementing an external memory bank trained to preserve previously learned and newly acquired information. This approach introduces Evolvable Neural Turing Machine, which makes it a simpler algorithm reported to be a better generaliser, with minimal memory access required. This is tested against the continue double T-Maze, proving that external memory can offer greater adaptability. This research work, on continual and one-shot learning, has been brought forward by [136], improving this algorithm by introducing a new property to the ENTM, a default jump mechanism that identifies portions of unused memory, which are then utilised to favour the evolution of continuous adaptive learning.

Another similar research on memory-like mechanism, used for ANNs adaptive properties, was conducted by [137]. In their approach, using a mechanism called adaptive synaptic delay has been identified to facilitate stability in the network's dynamics, achieving brain-like predictive functions. This was used to solve a complex dynamic control task using a bespoke sensorimotor controller.

The work of [138] shows that the approach of combining neuromodulation with functional modularity in neural networks such as [139], helps to prevent the overriding of previously acquired skills, known as catastrophic forgetting, as different modules of the ANNs are used to solve different tasks.

Another successful approach on solving this problem was proposed in [140], where the concept of diffusion-based neuromodulation was proposed. This technique allowed for plasticity to increase or decrease in different regions of the network, through evolution. Using this method enabled to largely eliminate catastrophic forgetting in simple ANNs.

Research conducted by [141] has provided evidence that Spike-Timing-Depended Plasticity can be evolved using NEAT. Neuroevolution has proven to be better at selecting network parameters that result in the best accuracy, depending on configurations. This resulted in networks more sensitive to their input encoding parameters.

Other more current work include; multi-objective evolution of network weight and topologies using NSGA-II and NEAT, augmented by the use of neuromodulation. This approach tests neurocontrollers with conflicting objectives. Interestingly, results of this research show that speciation is not necessary for neuromodulation as this approach already preserves innovation [142].

In [143] synaptic plasticity in the form of spiking controllers are deployed when adapting simulation learned controllers to reality for Unmanned Aerial Vehicles.

An aspect of neuroevolution which is closely related to meta-learning is known as architecture search. Designing neural network is an important step in the "learning to learn" approach. Deep neural networks are becoming increasingly more difficult to conceive for human experts, hence the necessity for neuroevolution to step in and offer interesting alternatives. Most recent research has been focusing on developing large scale deep neural networks [20, 89, 144, 145]. A lot of the methods that aim at complexifying architectures, by adding layers instead of neurons, have proven successful in image classification, detection and multitask learning.

Amongst these methods, highly successful approaches are found such as [87] where, similarly as NEAT, DNNs are evolved from simple networks, complexifying them through mutations that add entire layers. A further variation of this is proposed in [144], which achieved even greater performance by building modules of small neural networks that are used multiple times in a larger hand-designed neural network composition. This was inspired by the successes of stacking multiple layer modules to form DNNs [146, 147].

Another successful approach in architecture search can be found in [20, 148] where the authors develop DENSER, a neuroevolution algorithm for multitask learning. In this approach the layers and parameters of the DNN are evolved using DSGE. This was proven to be an extremely successful and a novel approach for discovering counter-intuitive architectures that would have not otherwise been designed.

Differently from the task of object detection and image recognition, which are found in computer vision, where the most successful approaches are based on CNNs, in language modelling and processing the most common architecture is the Long Short-Term Memory (LSTM). In this area it was also found that it is possible to generate successful designs automatically through neuroevolution [89, 149].

3.7 THE MODERN DAYS OF NEUROEVOLUTION

Since its inception, evolutionary computation discipline has gained a lot of interest and the progress made in later years is remarkable. For instance, in modern days, neuroevolution has been successfully utilised for the development of a fully automated approach to iterative game testing [150].

Another similar research is proposed by [151]. Here the authors highlight that modern automated environments for GUI testing are unable to exhaustively explore the entire state space of software, nor process the input space of graphical user interfaces, as they are highly dimensional. In this study, they propose a novel approach to address these issues. As neuroevolution can offer a scalable alternative to deep reinforcement learning methods. The method offers a higher robustness to parameter influences and a better handling over sparse rewards. The evolved agents in this work are trained to identify errors in software and are consequently rewarded for high test coverage.

In [152] neuroevolution is parallelised to perform a complexity analysis of the neural network and their performance, in terms of accuracy and time taken, compared to back-propagation, in order to identify sustainable ways of evolving competitive neural networks.

Further modern work [153] on transfer learning has provided an approach to identify specific neurons that are necessary in a neural network composition, which should be consequently transferred. The Hot neural components are transferred from the source ANN, to assist the learning at of the target ANN.

In [154] the authors highlighted the inefficiencies and disruptiveness of crossover in neural networks, due to strong functional dependencies of connection weights. They proposed a modularity-based linkage model at the weight level, preserving the functionally dependent building blocks of neural networks during crossover.

Additional evidence of successful neuroevolution work is presented in [155] where the authors try to address an issues of Topology and Weight Evolving ANNs, related to the design of genetic recombination, and the exploration of huge search spaces; especially in large-scale problems, which lead to impractical runtimes. In response, they develop a novel evolutionary framework for the neuroevolution of ensemble learners, using a specialised divide-and-conquer manner. Doing so, they address the problem of genetic recombination, as the search space for this operator is restricted to suitable solutions in the ensemble methods.

The work of [156] highlights a gap in the application of deep neuroevolution techniques to imitation learning. In this research they propose an assessment of whether neuroevolution can successfully be deployed for behaviour imitation on known simulation environments. A co-evolutionary adversarial generation framework is engineered, and it is evaluated by evolving standard deep recurrent networks, for the imitation of pre-trained agents, on eight state-based control tasks from Gymnasium. The results of these experiments indicate that in this realm neuroevolution can be a significant addition to deep learning, to generate accurate emulation of behavioural agents.

Furthermore, a recent remarkable work is seen in [157]. In this research the authors propose a neuroevolution method that automatically generates lightweight You Only Look Once — a state-of-the-art computer vision model, which is able to realise real-time white shrimp biomass assessment in recirculating aquaculture system, used for their detection. The method starts with an initialisation of a minimal YOLOs population, from a restricted search space of available building blocks. A constructive evolutionary search strategy is then employed to incrementally grow the YOLOs, adding and modifying their modules throughout the run, until reaching a preset performance threshold.

In the work of [158] the NEAE algorithm is developed, which is a neuroevolution system, specifically designed for classifying and predicting abnormalities in internet-based applications. In their approach, the system targets multivariate anomalies using parallel dimension processing, to achieved a synchronised intelligence. This proposed method is assessed using a large internet dataset on network traffic. Simulation results provide evidence on the effectiveness of this approach in detecting abnormalities, based on specific performance metrics.

Finally, a further interesting recent contribution in neuroevolution comes from the work of [159]. In this research the authors remark on the computationally expensiveness of neural architecture search, often related to the huge and intractable search spaces. In their work they propose the use of local differentiable stochastic neural architecture search to generate a hybrid neuroevolution algorithm, used in the fitness evaluation step. The results of this work highlight the successfulness of this approach in effectively selecting the correct recurrent neural network memory cells.

3.8 ILLUMINATING FITNESS LANDSCAPES AND THE SEARCH SPACE

Search trajectory networks (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. STNs were initially inspired by Local Optima Networks (LONs), created in [46] for the study of NK Landscapes. LONs, on the other hand, were inspired by network-based models of energy landscapes for computational chemistry [160]. This discipline made use of barriers trees [161] and disconnectivity graphs [162, 163] to study these phenomena, graph-based modelling tools later applied to also study fitness landscapes of optimisation [164]. LONs as we defined in details above are a graph model of fitness landscapes where nodes are local optima and edges are transitions amongst optima, with a given search operator.

This technique has been recently successfully applied in many areas of evolutionary computing research, which have produced many profound contributions such as the recent outlook on randomness in LONs sampling [165], or the fitness landscapes of channel configurations for CNNs, derived through Neural Architecture Search [166]. Similar studies have been conducted on evolved CNNs to investigate their hyperparameter configurations, and the effectiveness these have [167]. Furthermore, the tool was used to analyse the fitness landscapes of a popular tabular Neural Architecture Search benchmark, to discover that by using the findings, to build a fitness-aware Iterated Local Search can help to outperform popular evolution and reinforcement learning algorithms [168]. LONs have also been applied on real world domains such as the investigation into assisted seismic history matching, where the optimisation problem is to find the best subsurface reservoir model for prediction of field performance in the energy industry [169]. Moreover, the technique has been deployed to study the varying strength of perturbation on fractal dimension for the Quadratic Assignment Problem of the fitness landscapes generated by Iterated Local Search [170]. Further studies have used the local optima modelling technique to illuminate crossover in fitness landscapes and discover that the operator can induce networks of overlapping hypercube lattices, on the travelling salesman problem. Interestingly, these network model can be used to infer the existence of local optima not reached through the sampling process itself [171].

Further inspirations for these visualisation techniques were derived from *interaction networks* [172] to study population-based algorithms. The inception idea was to deploy graphs whose connections indicate the interaction of individuals, where vertices are individuals activated by others, throughout generations. Moreover, such strategies of modelling and visualisation were deployed in the examination of differential evolution [173, 174] and particle swarm optimisation [175, 176], to map the influence of individuals/particles inside the swarm (population). Such studies offered the evidence that these advanced tools can be useful to capture the exploration-exploitation dynamics of evolutionary algorithms. On the contrary, STNs, differ from the aforementioned techniques as they do not aim to depict the interactions between individuals but to trace the search path (trajectory) performed throughout optimisation.

Amongst other successful techniques for evolutionary computing on search-specific processes we find Gavel [177], an effective tool for modelling and visualising the effects of crossover and mutation in the assembly of solutions in genetic algorithms. In this study, the innovative tool has been proven effective on three different problems: a timetabling problem, a job-shop scheduling, and Goldberg and Horn's long-path problem.

Furthermore, in multi-objective optimisation the work of [178], shows, through dimensionality reduction, the progress of optimising and improving objective values. A similar line of enquiry involved trace generation plots [179], where they depict the hypervolume values changes over iterations. These differ from search trajectory approaches as they focus on the illumination of objective spaces rather than the solutions themselves.

Moreover, other similar techniques are found, where to track search behaviours, multidimensional solutions are turned to lower dimensions. The dimensionality reduction techniques found are principal component analysis [180], t-distributed stochastic neighbour embedding [181], and Sammon mapping [182]. In this way the position of individuals and movement is visualised by running a sequence of 2-D frames [183] or stacking of these to produce 3-D models [183].

STNs can be considered similar to the aforementioned approaches, but in addition, this technique offers a clear insight into specific search dynamics, as the paths through the search space are captured in the graph object. In this approach, graph information can be analysed at different granularity levels by changing the definition of a location.

STNs differ from LONs in that the nodes represent states of the search process, not necessarily local optima, which generalises and extends the use of this graph-based model of search dynamics. Once a system is modelled as a graph (network) it can be visualised and analysed with the vast variety of powerful analytical and visualisation tools provided by the science of complex networks [17]. STNs were initially proposed to characterise Differential Evolution and Particle Swarm Optimisation for several classical continuous optimisation benchmark functions [47]. STN analysis was later extended to cover not only population-based algorithms but also stochastic local search methods, and both continuous and combinatorial optimisation problems [184]. More recently, the technique has been successfully applied to the Cyclic Bandwidth Sum problem in [36], also in the field of multi-objective optimisation in [185, 186, 187], to highlight phenotypes and mutational transitions of linear genetic programming system as graph-based models [188]. This further corroborates the usefulness of this modelling technique and its applicability, and potential extensions to different realms.

3.9 CONCLUSIONS

In summary, as discussed in the literature, we observe the significant growth of the field of neuroevolution since its inception. The literature review focused on assessing this realm, particularly the research concerned with the visualisation of meta-heuristics through complex networks, namely Local Optima Networks and Search Trajectory Networks.

The review of the literature presented various strands and facets of neuroevolution, including categories of algorithms and subfields aiming at the bio-inspired evolution of neural networks. It became apparent that there is a vast array of algorithmic developments, often pursuing similar objectives. Additionally, assessments of the operators developed for these algorithms provided contrasting views on their usefulness and efficiency, especially *recombination*. All of this poses a questions; how well can we understand the mechanisms of these operators and how do we compare and contrast these neuroevolution algorithms?

The successes of LONs and STNs in meta-heuristics, as presented in the literature, have inspired us to pursue this line of study further. Aiming to offer a more advanced tool leveraging complex network visualisation to understand and display the intrinsic mechanisms that operators and search strategies in neuroevolution form. The literature has highlighted this gap. The applicability of Search Trajectory Networks to neuroevolution and its various facets has not yet been explored.

4

CHAPTER 4 — RECOMBINATION IN NEAT: A SEARCH TRAJECTORY NETWORKS PERSPECTIVE

This chapter presents the research work published in [7]; reported in the List of Publications table, at the start of this thesis. The reviewed literature highlighted that the STNs network visualisation technique, initially proposed in Ochoa (2021) [35], and inspired by its predecessor LONs [46], has not previously been used to study neuroevolution search spaces, nor the dynamics of the NEAT algorithm. We aimed to fill this gap by applying Search Trajectory Networks (STNs), for the first time, to the study of NEAT and its *recombination* operator.

As discussed in Chapter 2.2.1, NeuroEvolution of Augmenting Topologies (NEAT) is a Genetic Algorithm (GA) used for evolving dense neural network topologies and the associated weights — without means of back-propagation. It has been proven effective and adaptable at solving reinforcement learning tasks. Nevertheless, earlier work [2, 32] highlighted the partial inefficiencies related to the recombination operator in NEAT. We focus our attention on crossover; in the original publication, the operator was deemed an important contribution to the known issues in neuroevolution [50], discussed in Chapter 2.2.1.

The results of this discussion will demonstrate the success of this approach in analysing whether this contested operator behaves as intended, from a search space perspective. Furthermore, the technique used in this research has helped to pave the way of illuminating neuroevolution search spaces; offering an advanced tool, beyond the analysis of fitness performance, to assess neuroevolution.

In this work, we offer a visual and statistical analysis to examine the behaviours of NEAT, with and without the recombination operator (crossover). The experiment follows two benchmark tasks found in the original NEAT article [2]: XOR and Double-Pole Balancing. Results obtained from inspecting the search space dynamics demonstrate that this operator does not produce the intended results as proposed in [2, 115].

4.1 KEY CONTRIBUTIONS AND MOTIVATIONS

This is the *first* contribution chapter of this thesis. Outlined here is the seminal work related to the novel application of STNs — a complex network visualization technique used to observe search space trajectories — in the realm of neuroevolution. This technique is adapted to model the search dynamics of the NEAT algorithm. The key contributions of this chapter are reported as follows.

- Adaptation of 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
- Explore reported inefficiencies and the role of crossover in neuroevolutionary systems

4.2 HISTORICAL MARKINGS IN RECOMBINATION

In this section we delve deeper into the workings of the NEAT recombination operator, how the operator uses an innovation numbering system to crossover solutions, as well as how the approach was originally conceived for solving the *permutation problem* [50].

As previously discussed in Section 2.2.1, historical markings are a way of numbering the genes innovations that occur in a NEAT genotype. These indicators are used in many operators of the algorithm, especially to perform recombination between two high performing parents selected from the population. This is known as a *global innovation counter*, which is incremented and added to a new gene each time it appears. This is a chronology of every gene that is encountered in the system during a run. Hence, it is easy to imagine how this non-computationally demanding technique can help track topological creations. Innovation numbers are unique IDs that do not change during the same run, and they help to establish the inheritance of genes in offspring. Let us explore the example used in [2] to outline the concept of crossover and its role in addressing the competing conventions problem [50].

Figure 4.1 provides an illustration on how recombination occurs by leveraging the innovation numbering system. By stacking the genotypes of two selected parents, we observe both matching and non-matching numbers in one of the parents; similar to a masking exercise. These can either be *disjoint* or *excess* genes. Disjoint genes occur within the genotype and excess are the

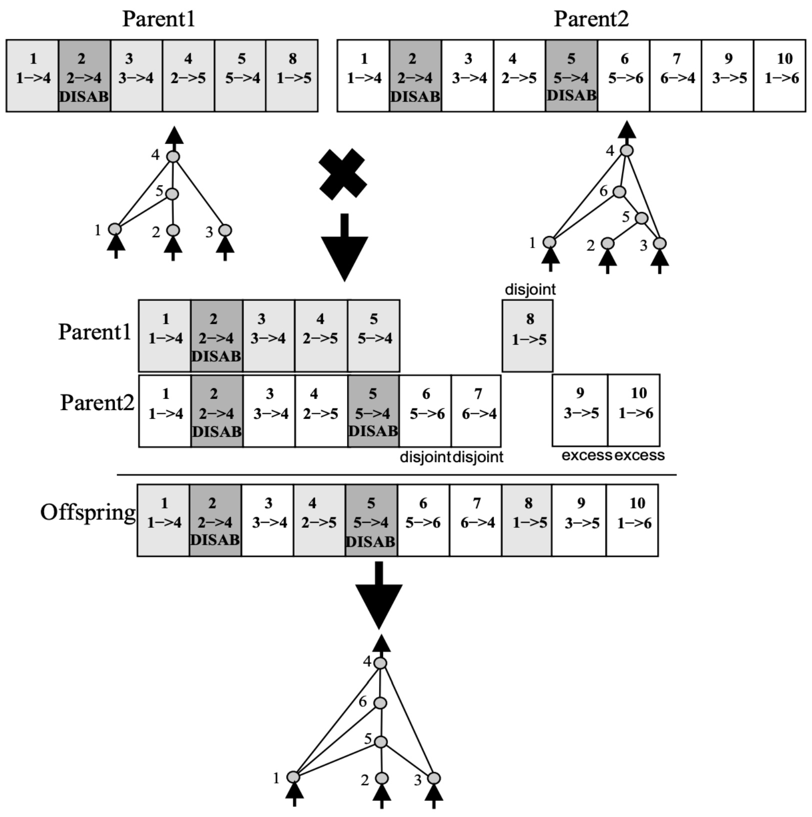


Figure 4.1: The role of recombination in NEAT. Sourced from [2]. The number at the top of each gene cell is the innovation number. The arrows signify the edge component, in the form outbound cell - inbound cell.

ones that exceed the length of a genotype. It is worth noting that this is what occurs for the topological information not the weights of genes. These are in fact transferred from the highest performer of the two parents in excess and disjoint genes; the matching genes, which are those genes that found in both topologies, are picked randomly.

From this explanation we can perceive how this operator should intelligently work to incorporate high quality information from both parents. Although, it can also be deduced from its complexity, how this mechanism might be prone to issues and not work as intended. In fact, as the authors state, this operator would be ineffective without a *speciation* mechanism that would separate topologies into different niches, allowing them to optimise without being prematurely eliminated [2, 115]. This is due to smaller structures optimising faster than larger ones, as initially introducing nodes and connections typically reduces fitness. Topologies that are recently augmented have a lesser chance of surviving without a mechanism of protection. As mentioned earlier, some of these topologies may serve as crucial stepping stones for solving the task in subsequent generations [115].

Now that the recombination operator of NEAT has been explained, let us elucidate how NEAT genotypes — which are network topologies and weight solutions — were converted into flattened signature representation. Forming the nodes of our STN visualisation models.

4.3 MODELLING NEAT ARCHITECTURES TO STNS SIGNATURES

This is an important step, as generating STNs signature from neuroevolution genotypes was a fundamental hurdle that had to be overcome for STNs to be successfully applied to the neuroevolution search space. When we think of a node in an STN network, we can consider this as a signature that easily represents salient aspects of a solution: a neural network. For instance, in the paper that initially proposed STNs [19], combinatorial optimisation using population-based algorithms is achieved for a single objective, static, bound-constrained, multivariate minimization problem. This domain is defined with the following formalism.

$$\min f(\mathbf{x}), f : \mathbb{R}^n \to \mathbb{R}, \mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^n, \tag{4.1}$$

In the definition of 4.1, $\mathbf{x}$ is an n-dimensional vector representing a candidate solution. $\mathcal{S}$ denotes a feasible subregion of $\mathbb{R}^n$ as the domains of the variables within $\mathbf{x}$. $\mathcal{S}$ has boundary constraints, as all components of the *solution vector* (shown in 4.2).

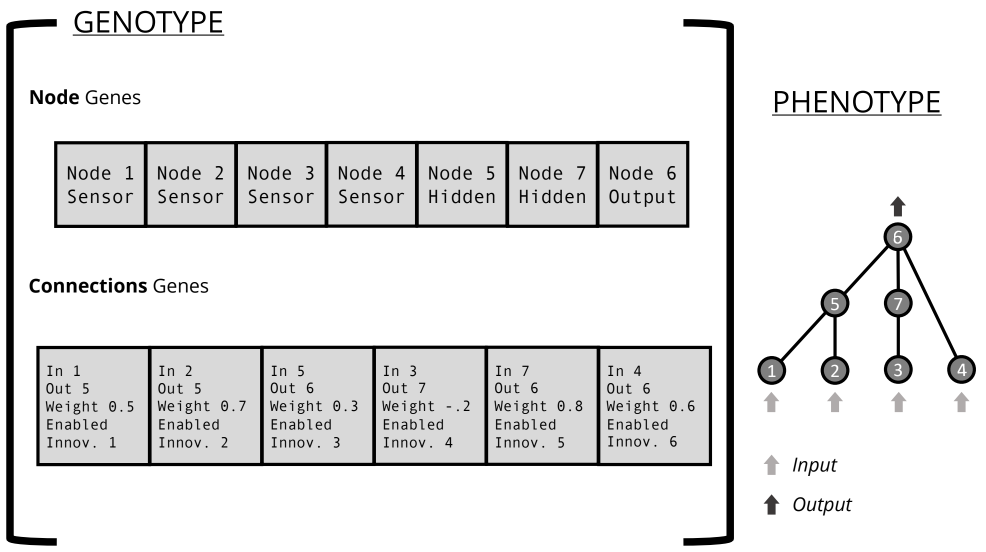
$$x^{\min} \leqslant x_i \leqslant x^{\max} \quad \forall \mathbf{x} \in \mathcal{S}, \quad 1 \leqslant i \leqslant n \tag{4.2}$$

Given the above definition, a solution in traditional STNs is a vector of real numbers, bounded by the domain. Considering a function like *Rana*, used in [19], a candidate solution would be comprised of real values in the range expressed in 4.3.

$$x_i \in [-512, 512] \tag{4.3}$$

When neural networks are involved, things differ greatly, as candidate solutions become increasingly more complex. A larger set of information is encoded in the solution genotype, as the search space is vastly larger in terms of parameters. For an accurate and effective Network Architecture Analysis (NAA) — discussed in Chapter 8 — all these information are essential and must be encoded and modelled in our visualisation approach. The inherent characteristics to be mapped are presented as follows.

If we consider the NEAT genotype, this is composed of two levels as outlined in Figure 4.2. One level takes care of encoding the *node* genes, while the other is responsible for representing the *connections* between these. The node genes part of the genome includes a gene ID and a description of whether the node is an input (sensor), hidden, or output node. On the other hand, the connection genes of the genome include information such as the direction of connections (from node to node), the weight assigned to the connection, whether this gene is enabled (active) or disabled (inactive). In this illustration there are no inactive genes. Finally, a crucial part of the genes encoding, is the innovation number, used by the various operators of NEAT. Every time a new gene is introduced (found by search), this number is incremented; when genes are inherited, they maintain their unique innovation number.



GENOTYPE
Node Genes
Node 1 Sensor
Node 2 Sensor
Node 3 Sensor
Node 4 Sensor
Node 5 Hidden
Node 7 Hidden
Node 6 Output
Connections Genes
In 1 Out 5 Weight 0.5 Enabled Innov. 1
In 2 Out 5 Weight 0.7 Enabled Innov. 2
In 5 Out 6 Weight 0.3 Enabled Innov. 3
In 3 Out 7 Weight -.2 Enabled Innov. 4
In 7 Out 6 Weight 0.8 Enabled Innov. 5
In 4 Out 6 Weight 0.6 Enabled Innov. 6
PHENOTYPE
6
5
7
1
2
3
4
Input
Output

Figure 4.2: An illustration of the genotype-phenotype dichotomy. Adapted from [2].

It is now possible to perceive the difficulty involved in incorporating all this information into a STNs signature that was previously only used to represent a simple vector of real numbers, as it is in the original STNs approach [19].

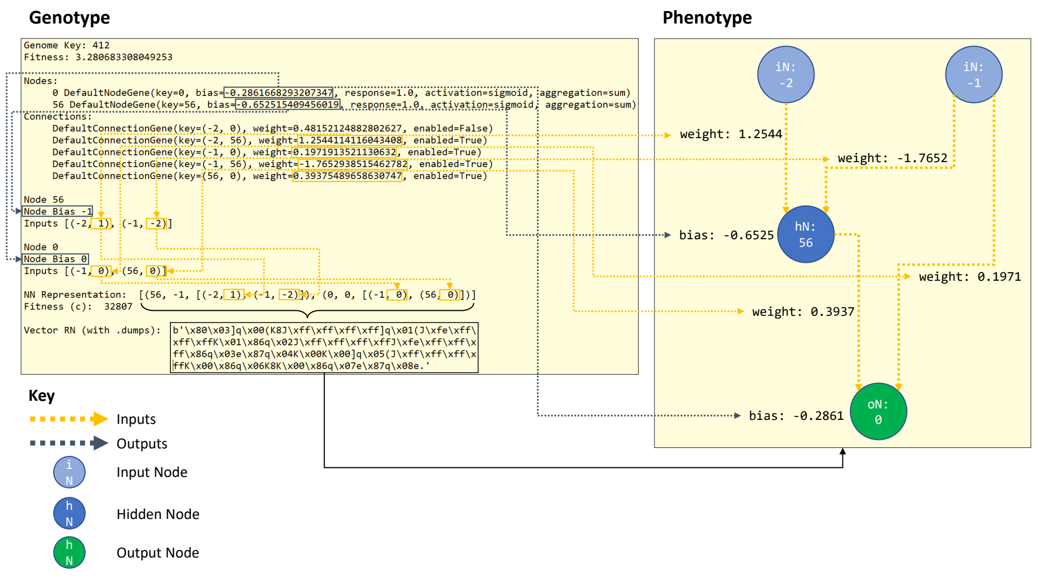
In order to achieve this, an effective method was found, which would linearly map, compress and encapsulate all the information into a unique and reproducible signature encoding, for the nodes of our STN construct. The proposed solution [7] was to leverage a simple function derived from the native `Pickle` python library. This is called `pickle.dumps` and it is used to form a *serialisation* of an object's hierarchy [189]. Since the genotype of a NEAT solution can be expressed as an object, and it represents the architecture of a neural network in an orderly fashion, this function was used to return the pickled representation of that object as a *bytes* object.

Other more manual and bespoke approaches were considered but did not add any improvement compared to the chosen method. An example of this is presented in 4.4.

$$b'|x80|x03]\ q|x00K|x00K|x00]\ q|x01J|xf|xffK|x00|x86q|x02a|x87q|x03a.' \tag{4.4}$$

The use of this function is important as it enabled the conversion of information from potentially large neural network architectures into more concise and consistent string representations, which are unique and reproducible. This was particularly necessary, due to the complexifying nature of NEAT, where topologies may start small and then progress to more intricate and larger architectures.

Given the above example, let us proceed to illustrate further how the mapping process occurs from topology, to structural vector of real numbers, to compressed signature achieved using the `.dumps` serialisation function (shown in 4.4). Figure 4.3 shows the exemplification of this process, by using as an example the extract — derived from an experimental run of this study — of the results produced by the stable python implementation of NEAT [190]. The extract illustrated presents a candidate solution; `genome_id: 412`, which has a fitness of ≈ 3.28.



```
Genome Key: 412
Fitness: 3.280683308049253

Nodes:
     0 DefaultNodeGene(key=0, bias=-0.2861668293207347, response=1.0, activation=sigmoid, aggregation=sum)
     56 DefaultNodeGene(key=56, bias=-0.652515409456019, response=1.0, activation=sigmoid, aggregation=sum)
Connections:
     DefaultConnectionGene(key=(-2, 0), weight=0.48152124882802627, enabled=False)
     DefaultConnectionGene(key=(-2, 56), weight=1.2544114116043408, enabled=True)
     DefaultConnectionGene(key=(-1, 0), weight=0.1971913521130632, enabled=True)
     DefaultConnectionGene(key=(-1, 56), weight=-1.7652938515462782, enabled=True)
     DefaultConnectionGene(key=(56, 0), weight=0.39375489658630747, enabled=True)

Node 56
Node Bias -1
Inputs [(-2, 1), (-1, -2)]

Node 0
Node Bias 0
Inputs [(-1, 0), (56, 0)]

NN Representation:  [(56, -1, [(-2, 1), (-1, -2)]), (0, 0, [(-1, 0), (56, 0)])]
Fitness (c):  32807

Vector RN (with .dumps):  b'\x80\x03]q\x00(K8J\xff\xff\xff\xff]q\x01(J\xfe\xff\
                          xff\xffK\x01\x86q\x02J\xff\xff\xff\xffJ\xfe\xff\xff\x
                          ff\x86q\x03e\x87q\x04K\x00K\x00]q\x05(J\xff\xff\xff\x
                          ffK\x00\x86q\x06K8K\x00\x86q\x07e\x87q\x08e.'
```

Figure 4.3: Exemplification of the mapping process from genotypes to unique signatures for the construction of STNs. Sourced from [7].

The extract, achieved by logging the *best individual* in the population — as a genotype object — is a process which is custom in STNs analysis [19, 35]. The rationale behind this choice, is that the best individual in a population, is often a good representative of the search stage at a given point in the run. This genotype object extract, gives us a precise and adequate inspection of the details that comprise a representative solution. However, this extract is not usable in its current state and needs to undergo a pre-processing stage, which is described below.

Figure 4.3 offers a more comprehensive and detailed overview of the main concepts introduced in Figure 4.2. This illustration portrays the genotype nodes and connections. Nodes have identifying numbers. Note that the input nodes (-2 and -1) are not listed, as they are pre-set in the configuration file by the user. This is because each problem has different input requirements, and these must be set a priori, based on what should be sensed from the domain. Therefore, in this example, output node `node: 0` and `node: 56` are the ones that have been evolved by the algorithm. Evolved nodes and connections are the primary focus of this analysis, as the criteria of evolvability is what needs to be captured and interpreted using this methodology.

Aside from the identification numbers, nodes carry a *bias*. This can be considered a factor informing how positively, or negatively, information flowing through the node, should be considered in the network. Therefore, bias is an important component incorporated in our signature representation. There is a limit to the amount of information we can include in a signature; therefore, the other pieces of data illustrated in the nodes' section are not included. The reason is that, if we enlarge the information carried in signature components, in essence, we explode the search space by shining a light through its vastness — at different dimensional levels. We need to compress our analysis to strictly essential components. After careful consideration, based also on the literature surrounding NEAT [32], weight, bias and node innovation number, were the three fundamental information of interest to our visualisation analysis. This is also because they are the fundamental components that constitute TWEANN algorithmic systems, such as NEAT. The response, activation and type of aggregation were excluded alongside genes that were not enabled by evolution.

In the connection section, there are other crucial components that must be selected and incorporated. These can be, the direction of connections, expressed as binary coordinates (from node to node), the weight of connections, which, similarly to the bias, give us an indication of the significance that connections have, and the validity of these (through a Boolean expression). It is hard to single out how each of the connections weights work, as these are supposed to operate as a conglomerate network apparatus; therefore, all of these are incorporated in the signature. Finally, we have the validity, whether a connection is valid and active or invalid and inactive.

The validity is controlled by evolution. Similarly, as in the nodes section, to avoid search space explosion, only the weights of those connections that are valid (enabled) are incorporated in the signature. Direction coordinates will be implicitly expressed as the topological structure of the signature itself.

Now that the genotype object has been detailed, we can progress to outline the first step in the mapping process using the proposed technique [7]. When the pseudo-phenotypical vector representation structure is created (*NN Representation* in Figure 4.3), the focus is on the flow of information from nodes acted upon by other nodes. In the case of *genome 412*, we observe what happens to `node 56`. Node 56 has a bias and two inputs that are outlined in the connections portion of the genome. Connections come from both input nodes (-1 and -2). Worthy of notice, is the numbers reported in our pseudo-phenotypical NN vector representation do not coincide with the ones shown in the genotype object discussed earlier. This is due to search space partitioning, which is a conversion applied to real values in the genotype. Let us explain and justify this process in the following section.

SHINING A LIGHT ON THE SEARCH SPACE As discussed earlier, including information components to a signature in STNs, is a fundamental choice that can be derived empirically, with an extensive understanding about the algorithm in analysis. This is analogous to having a torch and deciding how bright and vast this torch should shine on the search space. Being too detailed in the components of our information signature, is like narrowing the torch to a laser beam. Doing so, might reach far into the search space and pinpoint specific criteria, but the width of illumination would be rather narrow, and possibly inconclusive. This may lead to a poor understanding of the search space dynamics, related to the algorithm in analysis. Our intent, is to offer an overarching visualisation model, of some critical components of solutions in the search space; essentially tracking the algorithm's traverse. Fundamentally, it helps control the granularity of the visualised models.

Firstly, we must select the cardinal components to be included in the signature, and secondly, partition the search space into discrete portions. In simple terms, this involves reducing the decimal point precision of the numerical values encoded in the signature. This will not only reduce the granularity, and help to construct the overarching visualisation, clustering similar solution to common location of the search space, which is then expressed as a node in the network model.

As per the STNs definitions found in Section 2.1.2 of Chapter 2, a *location* is a non-empty subset of solutions that results from a predefined partitioning of the search space. This partitioning, together with the choice of elements to be included in the signature, is largely established by empirical choice, made with some fundamental knowledge about the algorithm to be analysed. When partitioning, the creation of aforementioned search space portions — which can be described as hypercubes — is trialled by the use of a Partition Factor (PF).

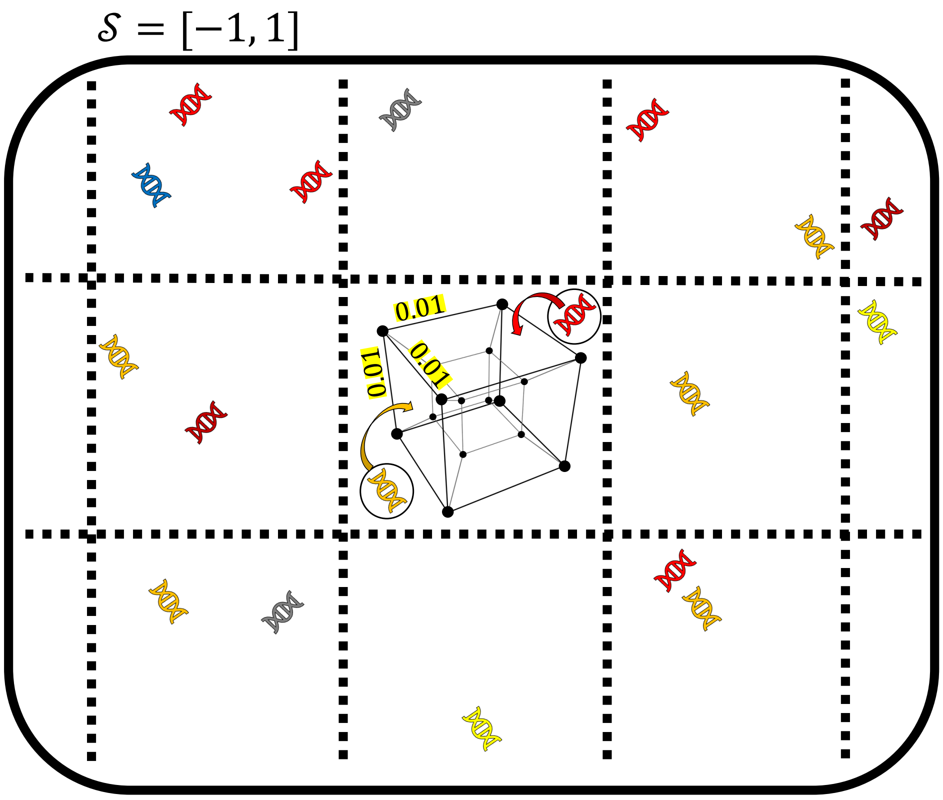


Figure 4.4: A simplified illustration of the 3 dimensional search space partitioning into uniform hypercube dictated by the PF.

Figure 4.4 shows the equivalent 10^{-2} hypercube, using a PF $= -2$. Thus, the solutions are mapped to locations by rounding off the precision values to the nearest 10^{PF}, which defines the identity of the enclosing hypercube. The two (red and orange) circled solutions, shown in the illustration, would both fall into the same hypercube shown. This explanation follows the example given in the seminal paper [35], where the problem is 3 dimensional, with a search space domain equivalent to the solution range $S = [-1, 1]$. Setting the PF to -2 would mean that our adjacent solution points would be discretised in a hypercube of $0.01 \times 0.01 \times 0.01$.

Now that the concept of search space partitioning has been clarified, let us return to the final stages of the signature mapping creation, through byte encoding. Figure 4.3 reports the solution vector with the values' precision reduced by our empirically designated *precision factor* (PF) — a value discussed in the following parameters section. In this vector, denoted by the square brackets ($[*, *, *]$), we track the topology, starting from `node 56`, which has a rounded bias of -1. Within this, we have a list of tuples that coincide with input node `-1` and `-2`, plus their respective weights. The same occurs for the output `node 0`, which systematically represents the further components of the topologies.

Finally, now that this vector has been assembled and partitioned according to the method described above, the whole vector (a list of lists and tuples) is passed to the `.dumps` function. This consistently hashes the vector to a specific and unique byte string, highlighted by the box in Figure 4.3.

4.4 COMPLEX NETWORKS CHARACTERISATION

One of the main objectives of this research, is to evaluate the recombination operator using a visualisation technique, which aims at illuminating the search space, using complex networks. The inspiration was obtained from the *ablations* studies conducted in [2]. These aimed at isolating key properties of NEAT, to assess whether removing algorithmic components/operators had a significant impact in performance. Four ablations were studied in [2], non-growth, random initialisation (instead of minimal initialisation), non-speciation and non-mating, as the authors call it, which is the removal of crossover.

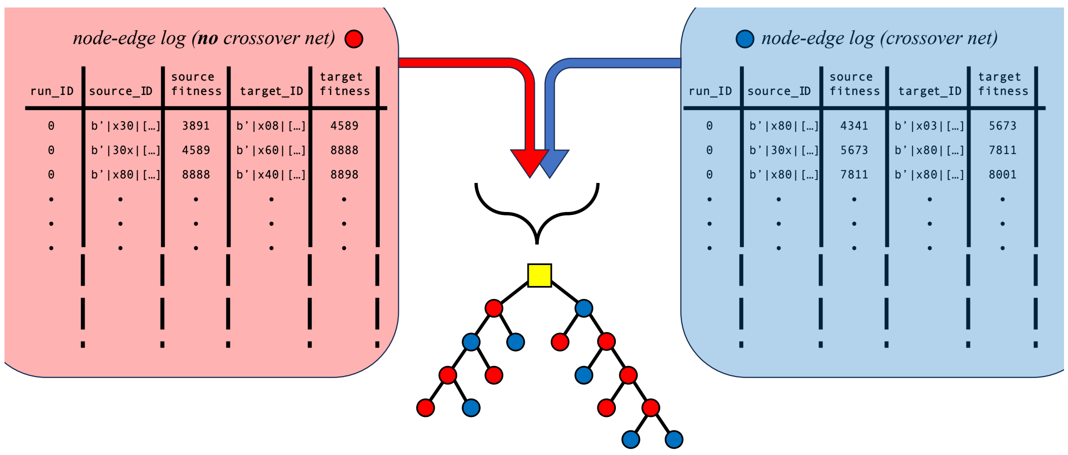
The `neat-python` implementation found in [191] was used for our analysis. With this version, we proceeded to create two variants and compared their generated networks. One variant was left in its original form, as found in the NEAT paper [2]; all parameters were left unchanged, to exactly reproduce the experiment. The other variant is similar, except for the removal of crossover.

Let us proceed to describe how the complex networks visualisation analysis has been ideated and carried out.

4.4.1 *Merged Search Trajectory Networks*

STN models do not require any additional sampling methods. They are built from data gathered while running the evolutionary, or meta-heuristic algorithms under analysis. For each of the chosen problems, an STN is constructed by aggregating all the unique nodes and edges encountered across five independent runs of each NEAT variant. Five runs have been empirically deemed an appropriate sample, capable of illustrating these neuroevolution dynamics.

This is achieved from an STN/NTN log, which is the extract of running the variants on the problem domain and selecting the best individual in the population. This is to be modelled as node-edge transition, which we define as a trajectory. In Figure 4.5 an example of a node-edge log is presented; used to generate the merged networks.


*node-edge log (**no** crossover net)* ●

run_ID	source_ID	source fitness	target_ID	target fitness
0	b'|x30|[...]	3891	b'|x08|[...]	4589
0	b'|30x|[...]	4589	b'|x60|[...]	8888
0	b'|x80|[...]	8888	b'|x40|[...]	8898
⋮	⋮	⋮	⋮	⋮

● *node-edge log (crossover net)*

run_ID	source_ID	source fitness	target_ID	target fitness
0	b'|x80|[...]	4341	b'|x03|[...]	5673
0	b'|30x|[...]	5673	b'|x80|[...]	7811
0	b'|x80|[...]	7811	b'|x80|[...]	8001
⋮	⋮	⋮	⋮	⋮

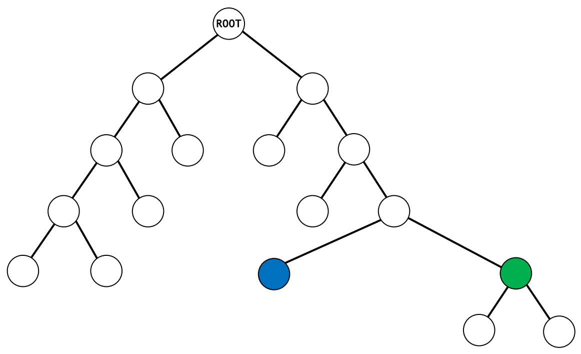
Figure 4.5: An exemplification of the *node-edge logs* necessary to compute the merged STNs structure for our multi-variant experiment.

These logs are used to generate *merged STN* models for the two NEAT variants: *with* and *without* crossover. The merged model is obtained by the graph union of the two individual graphs for that domain.

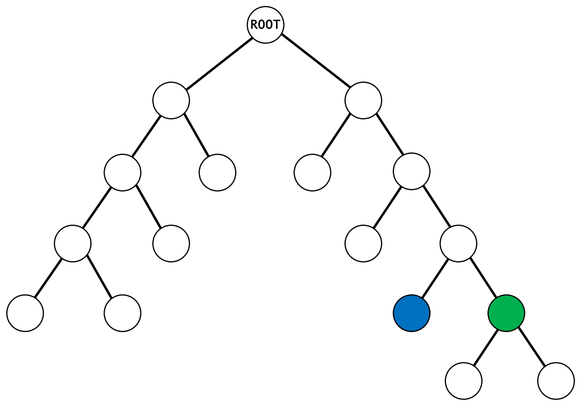
Using the graph theory formalism, let $\text{STN}_X = G(N_X, E_X)$ and $\text{STN}_{nX} = G(N_{nX}, E_{nX})$ be the STNs of algorithm variants X and nX, for crossover and no crossover, respectively. We construct $\text{STN}_{\text{merged}}$ as the union of the two graphs. Specifically, $\text{STN}_{\text{merged}} = G(N_X \cup N_{nX}, E_X \cup E_{nX})$. The merged network contains nodes and edges that are present in at least one of the algorithm graphs. Decorators are crucial; they are utilised for nodes and edges to indicate whether they were visited by both algorithms or only by one of them.

4.4.2 *The Reingold-Tilford tree layout*

In this analysis, the Reingold-Tilford tree layout [8] was empirically selected as the graph layout of choice. This is a directed graph, which aims at producing aesthetically pleasing and tidy trees, using minimum drawing space. This graph has a root node that begins at the top of the plotting plane, and develops to produce trees, following a given vertical progression. The main characteristic of this specific layout is that nodes of the networks and subtrees are closer together, producing a narrower graph. Figure 4.6, provides a simple demonstration of how this layout achieves the intended outcome. In previous layout algorithms — such as the Wetherell and Shannon [192] — the position of the green node in Figure 4.6a, is programmed to stay considerably far apart from the blue node. In Figure 4.6b we can observe how the placement of the nodes in question is changed, to leave a much narrower gap and to produce more compressed layouts. The algorithm strives to organise the vertices into layers, based on their *geodesic* distance (path length) from the root vertex. In our case, the root is set to be the start of trajectories. Another benefit, is that this algorithm strives to minimise the number of edge crossing.



(a) Before RT layout



(b) After RT layout map

Figure 4.6: A demonstration of the tidier tree provided by the RT layout. Illustration adapted from [8].

The fundamental reason for this choice is that, in our analysis, we wanted to produce and showcase *merged* STNs, for the non-crossover and the crossover variant, in one visualisation plane. Upon generating the preliminary plots and conducting careful observations, it was noticed that when generated using the typical *force-directed layout* approach [193] — which is used in [19, 35] — they did not provide a comprehensible depiction of the dynamics of this system's variants. A tidier approach was required, which could deliver maximum information in one single image, hence the justification for this type of tree layout. Furthermore, the modelling of this neuroevolution search space, produced many data points, which translated into many nodes and nodes transitions. Without such layout, it would have been hard to distinguish the progression thorough this space and to compare the variants.

Ultimately, as a researcher in this area of study, our objective is to maximize the information delivery of a network artefact. Doing so, to effortlessly convey and provide plausible explanations of what is occurring in highly dimensional and abstract environments. This is done, while producing mathematical graph objects that are aesthetically pleasing for the observer.

4.5 EXPERIMENTAL SETTINGS

The two variants will be analysed on the two benchmark control problems, used in the original NEAT paper [2]. XOR and DPV. These are outlined as follows, together with the domain specific parameters, used for the experiment. This ensures the possibility of reproducing the experiment in the future.

XOR eXclusive OR (XOR) is possibly the simplest of classification problems. It is a Boolean logic, which compares two input bits and generates one output bit — essentially a simple classification task. A trivial logic: if the bits are the same, the result is 0. If the bits are different, the result is 1. It is usually adopted to validate an algorithm's ability to function, according to its engineered principles. A sort of proof-of-concept testing domain.

This benchmark is intended to test for linear separability. Single Perceptrons do not have the capability to solve XOR, as they lack hidden layers. In essence, they are too shallow to learn the separation of all inputs, as demonstrated by the classification Table 4.1.

The hidden layer, referred to as a *feature vector*, can increase its dimensions and enhance linear separability by applying a non-linear transfer function.

The fitness function, used to calculate the classification, is based on Mean Squared Error (MSE). This is $1 - \sum_i (e_i - a_i)^2$, where the expected output (e_i) is subtracted to the actual output (a_i). Therefore, if the network can produce the exact expected output, it will gain the full fitness score of 1 (100% accuracy). Otherwise, it will receive a value less than 1, depending

Table 4.1: XOR input-outputs classification table.

Input 1	Input 2	Target Output
0	0	0
0	1	1
1	0	1
1	1	0

on the error, which decreases more rapidly, the further expected and actual outputs differ.

DOUBLE POLE BALANCING This is a more complex domain compared to XOR. It will also become evident by the result of this analysis, which will be discussed in Section 4.7.2. This benchmark problem is a control task. The dense neural network, produced by NEAT, are known to have good neurocontrolling capabilities, hence they are particularly suited to problems of this kind.

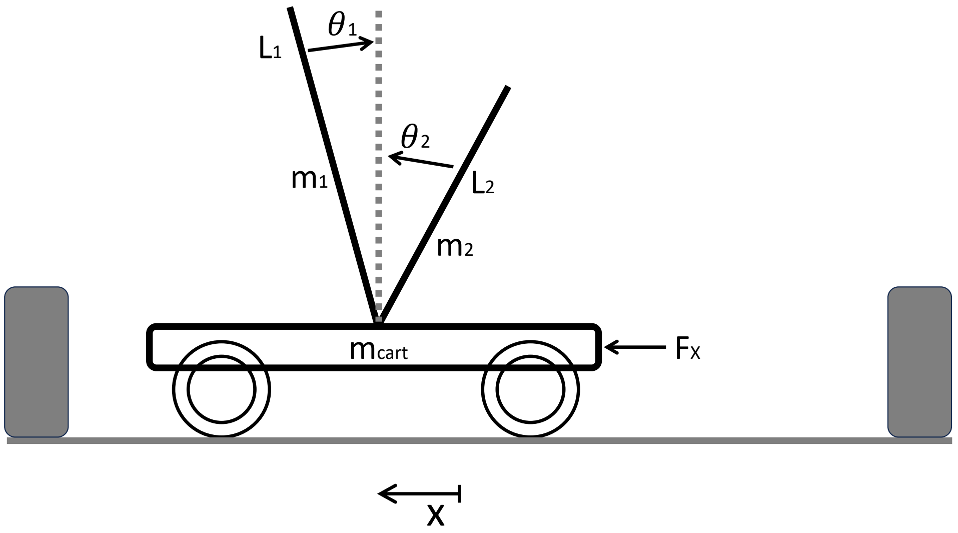


Figure 4.7: Illustration of the Double Pole Balancing domain (DPV). Figure adapted from [9].

The domain involves two poles, which are connected to a cart by a hinge. The task is for the neurocontroller, which is a neuroevolved agent, to maintain these poles balanced for a set time period. The neurocontroller has to master this task, by maintaining the cart within a predefined length of track. It achieves this by receiving some key inputs from this environment. Environment inputs and coefficient are explained in the Table 4.2.

Table 4.2: Parameter values used in for the DPV domain.

Symbol	Description	Value
X	The position of the cart on the track	$\in [-2.4, 2.4]m$
θ_i	Angle of poles from vertical	$\in [-36, 36]$ degrees
F_x	Control force applied to the cart	$\pm\ 10N$
L_i	Center of mass distance of poles to the pivot	$L_1 = 0.5$ m $L_2 = 0.05$ m
M	Mass of the cart	1.0 kg
m_i	Mass of the poles	$M_1 = 0.1$ m $M_2 = 0.01$ m
μ_p	Coefficient of friction of the poles pivot	0.000002
g	Gravity acceleration	$-9.8\ m/s^2$

Since this can be classed as a Reinforcement Learning (RL) problem, a RL signal (r_t) must be defined. It provides the neurocontroller with minimal information about the system state after committing an action.

$$r_t = \begin{cases} 0 & \text{if } -0.63 \text{ radians } < \theta_i^t < 0.63 \text{ radians and } -2.4 \text{ m} < x_t < 2.4 \text{ m} \\ 1 & \text{otherwise} \end{cases} \tag{4.5}$$

As shown in 4.5, this provides the system with a binary signal indicating the control task's status.To receive a 0, the angle of each pole is $\pm 36 degrees$ or ($0.63 radians$) from the vertical dotted line, depicted in Figure 4.7 and the cart stays within the track bounds (see X in Table 4.2), else it will get 1.

The actions, that are available to the controller, are detailed in 4.6 below. Where either a positive or negative force is applied to the cart, based on an action signal $a[t]$, received at time t.

$$F_t = \begin{cases} 10\text{N} & \text{if } a[t] = 1 \\ -10\text{ N} & \text{if } a[t] = 0 \end{cases} \tag{4.6}$$

Lastly, the objective function for this domain is expressed as shown in Eq. 4.7. This implies that to solve this domain, the control agent has to keep the poles balanced, for a maximum number of time-steps (t_{max} in Eq. 4.7). To balance the poles, the parameter set to $100,000$ equates to 30 minutes of simulated time. To calculate the fitness of a suboptimal solution, the loss is subtracted to the maximum fitness score of 1. In this implementation [9], the loss is calculated by subtracting the amount of evaluated time-steps of the candidate solution (t_{eval}), to the maximum time-steps (t_{max}), and normalising this by the given denominator. Additionally, as most trials are found to fail in the first 100 steps, logarithmic scales are used here, to have a better distribution of fitness scores, as we are testing against a large 100,000 steps.

$$\mathcal{F} = 1.0 - \frac{\log t_{max} - \log t_{eval}}{\log t_{max}} \tag{4.7}$$

Table 4.3 holds the main parameters used for the comparisons of these two domains, from a neuroevolution search space perspective. Note the *bias* and *weight* ranges, which determine the scale of our search space. Inputs and output nodes are necessary for the construction of our network, these are unchanged from the configurations and not evolved. Hence why, these are omitted from the nodes signature, as we are only interested in the nodes and connections that are evolved throughout trials and evolutions. The fitness thresholds are established based on the available literature and empirical tests. These are necessary to declare a solution optimal or not.

Table 4.3: Parameter values used in NEAT for each benchmark 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

As discussed, all real values, such as the ranges we have just pointed out, have to be reduced in precision, so that the search space can be visualised in a meaningful way, following the fundamental principles outlined in [35]. To achieve this, we empirically selected the PF value, which dictates the granularity of our search space modelling. Several modelling and visualisation trials have enabled the selection of a suitable factor, for partitioning the search space into hypercubes. This was achieved by rounding off the genotypes real values to $1e-0$ and $1e-4$ for the fitness values. These settings provide a comprehensively meaningful view of both XOR and DPV search spaces, with sufficient details to perform an informative examination of the algorithm variants' dynamics.

4.6 ANALYSIS RATIONALE

The search spaces examination has been achieved by producing 30 runs of each variant, on each of the domains. Independent runs are initialised with the same random seeds, for both compared variants. These were used to form the statistical analysis, in terms of fitness performance. To produce the STNs visualisation models of the search trajectories, a sample of 5 runs was selected for each variant. The selection criteria was to offer some representative of the performance achieved. The 5 best performing runs in terms of fitness achieved at the end of the run, were selected out of the 30. The reduced number of runs was to avoid an overcrowded model and visualisation. This is particularly important in the DPV domain, as these networks have substantially more nodes compared to XOR; mainly due to the larger population and number of generations, which highly increases the diversity and recorded data points.

Moreover, two statistical tests were conducted to determine the significance between the distributions of the variants. Preliminary visual and Shapiro-Wilk tests indicated non-normality. Hence, statistical tests were produced using the non-parametric Mann-Whitney test, setting the *p-value* to 0.05. The tests were conducted based on the following hypotheses:

- 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.

The distributions mentioned above relate to the *effectiveness* and *efficiency* of the algorithm. The first is measured by how quickly the variant is able to reach fitness threshold (see Table 4.3). Effectiveness, on the other hand, pertains to the number of successful runs that reach the threshold compared to those that fail. We performed test at two points of the generation ranges, one at midpoint (test 1) which is equivalent to 50 generations in XOR and 500 for DPV. Additionally, a secondary test (test 2) was conducted, equivalent to 100 generations for XOR and 1000 for DPV. This was an a posteriori decision, after having observed from the convergence plots (see Figure 4.8 and 4.9), that the average fitness performance, mostly began to diverge between variants, halfway through the runs. For successful solution of the domain, a fitness threshold was set, as suggested by the literature, together with the empirical trials conducted. The specified threshold for each domain is outlined in Table 4.3.

Beyond statistical tests, the analysis aims to identify whether differences are easily discernible by visualising the behaviour of these algorithmic variants within the neuroevolution search space. This assessment evaluates the feasibility of applying this analysis type to neuroevolution. In its inception, we witnessed the successes of STNs on general, population based algorithms [19, 35] and later, also in other realms [36, 185, 186, 187, 188]. In our case, the intent is to demonstrate the usefulness of STNs, by establishing a beginning to the NTNs methodology.

From the network plot, we derive and compute well known metrics, previously used for this technique, which are useful to further comprehend and corroborate the findings portrayed by the visualisations. Amongst these we find, the *number of nodes*, which corresponds to the amount of unique locations visited by each variant; the *number of edges*, which corresponds to the amount of unique search transitions between locations; the *average path length* (with standard deviation) from start to solution nodes — the length of a path is the number of edges it contains; the *number of shared nodes* in the STN model, which corresponds to the locations visited by both NEAT variants; and finally, the *number of shared edges*, which corresponds to the total search transitions, traversed by both NEAT variants.

4.7 RESULTS AND DISCUSSION

In this section, statistical and visualisation results achieved from this analysis, are discussed in details. Starting from the statistical results, both in terms of effectiveness and efficiency. Proceeding to the network visualisations, looking for confirmations of the statistical results and tests performed.

4.7.1 *Statistical performance results*

Let us begin to discuss the statistical tests performed which have given us the motivation to dispute this operator in NEAT.

Table 4.4: Significance testing for effectiveness and efficiency between NEAT with and without recombination.

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

As can be seen in Table 4.4, the results of these tests indicate that most of the variants' distributions differ, showing a significant difference ($p < 0.05$); in favour of H_1. On the other hand, for XOR, we can observe no significant difference, in the variants' effectiveness at the end of generations. This highlights that, although a noticeable difference is present halfway through the generations, both algorithm variants achieve similar levels of fitness (on average) at the end of the runs.

This is noticeable, especially by observing the convergence plots of the two variants solving XOR. Figure 4.8 presents the average fitness, over the 30 runs, for the best individual in the population at every iteration.

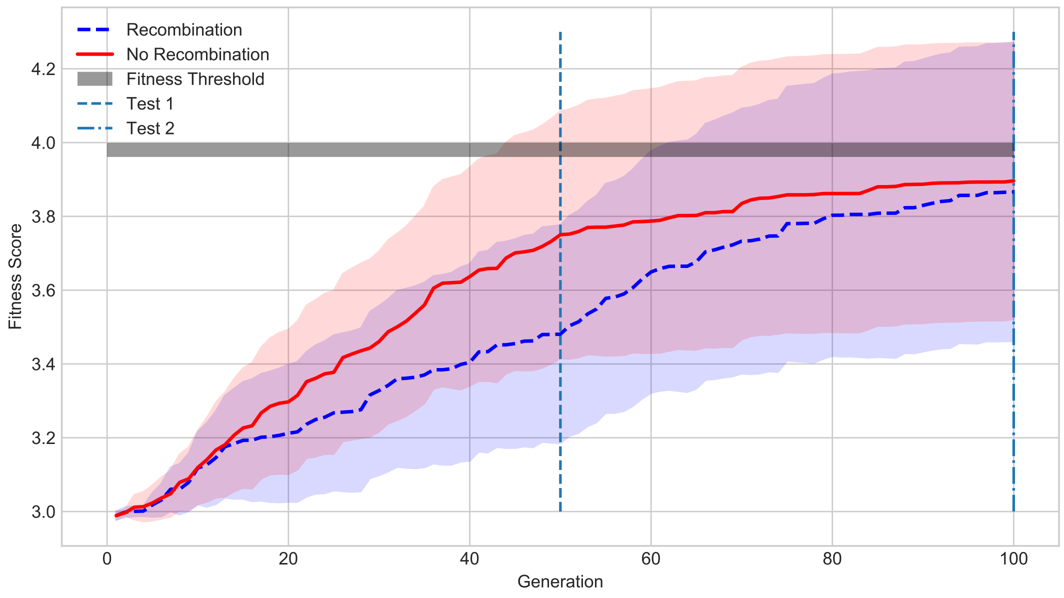


Figure 4.8: Average best fitness with standard deviations across generations for the two NEAT variants on the XOR domain [7].

The two algorithms start from similar levels of fitness, but soon, around the 15th generation, these start to diverge further apart, which is where the significant difference is captured. Then the algorithm variants begin to converge back to similar fitness scores, although the variant without crossover achieves higher fitness on average.

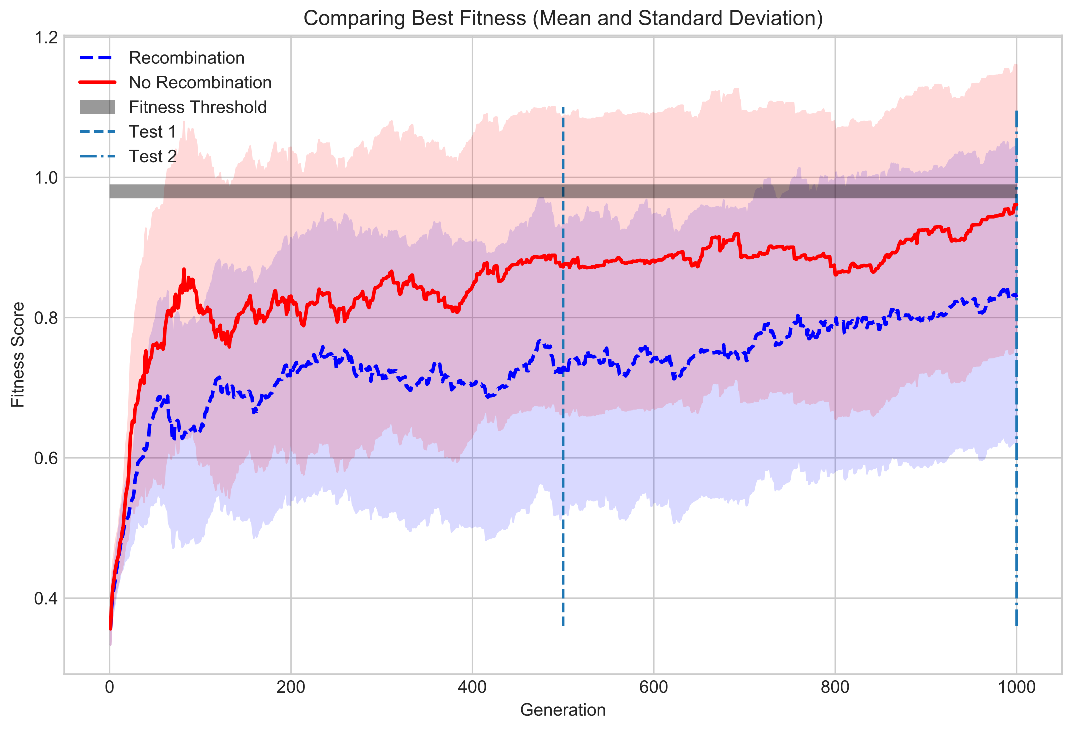
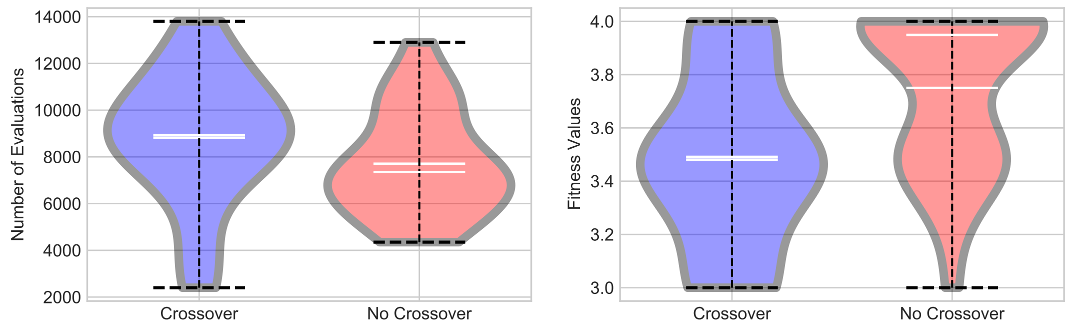


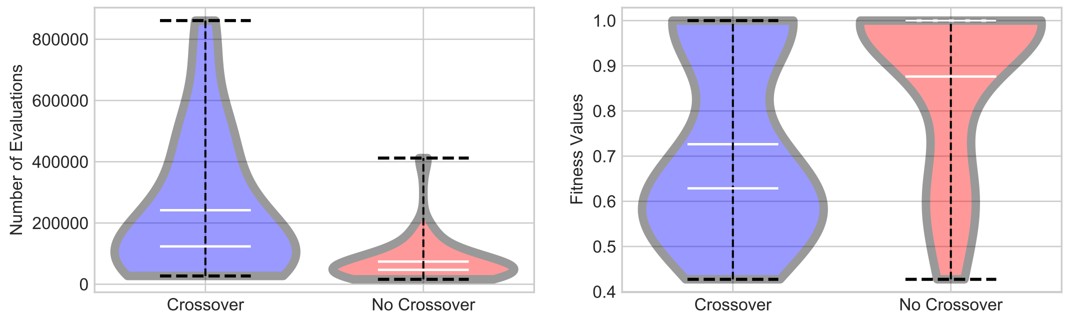
Figure 4.9: Average best fitness with standard deviations across generations for the two NEAT variants on the DPV domain [7].

When looking at the DPV domain (Figure 4.9), the performance characteristic is somewhat analogous. The variants both start from similar levels of fitness, then they rapidly progress to higher levels, maintaining a similar divergence, with the non-crossover variant, clearly superior than the crossover one; progressing all the way to the last iteration (1000 in total). Another observation can be derived from the plots, this is related to the oscillating nature of the best individual in the population. Throughout generations, the search in both variants progresses erratically to higher and lower fitness levels. This is due to elitism not being considered in our experiments. This decision was driven by intuition of recording all genuine iterations occurring in the population, both improvements and deteriorations, to offer a more comprehensive visualisation of the algorithm's evolutionary capabilities. Furthermore, all variants in both domain present very high variance, which is attributed to the inherent workings of this algorithm in general.

Figure 4.10 presents the distribution of average fitness values and number of evaluation required to reach threshold, for both domains and variants. Red is the variant without recombination and blue the one with active recombination. In the left-hand plots, we depict the distribution of the average



(a) XOR Domain



(b) DPV Domain

Figure 4.10: 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 in both problem domains.

number of evaluations required to reach a solution (fitness threshold) for each domain; meaning that a lower concentration of values, signifies a better algorithm. This would mean that it required, on average, less evaluations to find a strong solution that could solve the domain. We observe that no crossover, in both cases, but especially in the DPV domain, is particularly good at taking less time to find good network configurations that can solve the task. Another aspect worth noticing, is that the spread of distributions is narrower for the no crossover variant. The distribution of values in the crossover variant, resembles its counterpart, but presents wider variance, with greater upper and lower bounds.

The right-hand plots, on the other hand, illustrate the fitness values tested at midpoint (halfway through the run). This is where the most significant difference between the variants was observed. Here, a concentration of values, higher on the vertical axis, indicates a better algorithm. Once again, we observe that the no crossover variant, in both domains, can achieve better average fitness values across the 30 runs. Spreads, between variants and domains, are very similar. In XOR, the algorithm without crossover exhibits a bimodal distribution, with a portion of values concentrating between 3.4

and 3.5, similarly to the variant with recombination. Nevertheless, in no crossover, greater density can be seen higher towards the upper whisker (between 3.85 and 4.0). For DPV, in both cases, the distributions are bimodal; this is less noticeable in the system without crossover. A higher density of values that reside lower (between 0.5 and 0.7), can be seen in the crossover variants. On the other hand, the concentration of values for the variant without crossover is in the 0.85 to 1.0 range.

In [7], the values of distributions discussed above, are also presented in a table, in numerical form. The reader can examine these further, though here, in the interest of conciseness, we omit them.

4.7.2 *Illumination of NEAT search space using STNs*

This section presents the results achieved by visualising the solution of NEAT neuroevolution search space, for both domains. These are produced in accordance with the analysis premises outlined in earlier sections of this chapter.

Visualising networks of moderate size can be a useful tool, which allows us to perceive characteristics and features beyond the enquiry of network metrics alone. The node-edge diagrams used in this analysis are one of the most common visual representation of networks. They work by assigning nodes to points on a 2-dimensional plane and connect these adjacent points, forming the transitions. In *directed* graphs, the arrowheads are used to dictate the direction. Alongside these, the type of tree layout chosen in this research also gives a sense of direction, as the graph develops, and should be read vertically, from top to bottom.

To reiterate the rationale of this analysis, both variants' networks were merged into one, and visualised in the same plot. This is to view the interaction of both variants in each domain search space. We assign the colour red for the no crossover variant and blue for the crossover variant, for both nodes and edges. Locations that are visited by both variants (shared) are depicted in grey. The starting nodes are yellow, and the best solutions achieved are in dark grey, both are of slightly larger size, to ease reading. All other nodes sizes are proportionally dictated by the incoming weighted degree, which indicates how often nodes are revisited; thus attract the search process.

The graph visualisations in this work [7] were produced with `igraph` of the `R` programming language [194].

In the visualisation of Figure 4.11 and 4.12 we are able to observe that trajectories commence from the same *five* yellow square starting nodes. This

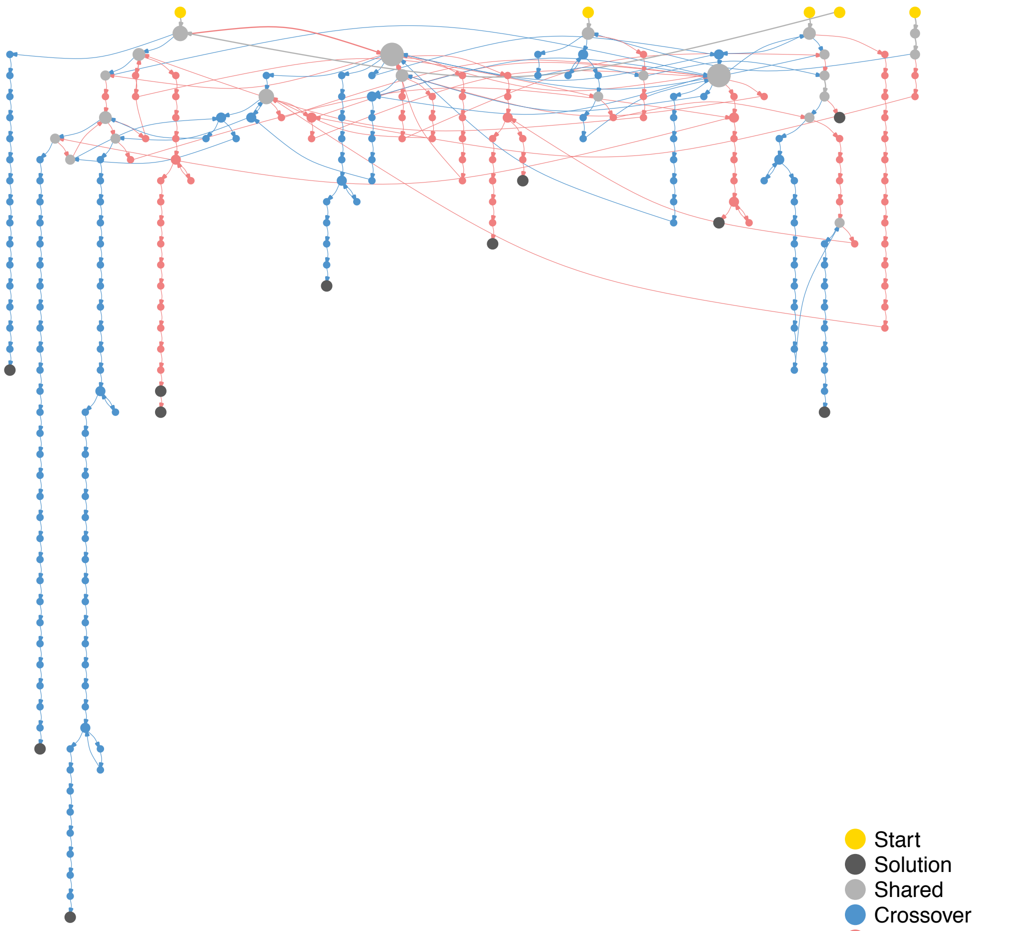


Figure 4.11: Merged STN for XOR. The nodes and edges visited by the crossover variant are decorated in blue and the no crossover variant are in red. Light grey nodes indicate locations visited by both variants. Node sizes are proportional to their incoming degree. The starts of trajectories are represented in yellow and the nodes achieving the fitness threshold (solutions) are shown in dark grey, and both of slightly larger size.

is due to having performed a selection of the same five runs based on performance, for both variants in each domain, using fixed seeds for the pseudo-randomness generator. Specifically, this occurs as in the fist iteration of a run both variants using the same randomness seed will generate the same solutions. From the next iteration onwards the differences of the variants setups will begin to demonstrate through the different trajectories dynamics, which is of key interest to this analysis. From there, the search begins, and the best individual in the population is tracked at each iteration; these are the representative solutions which are modelled as the networks' nodes.

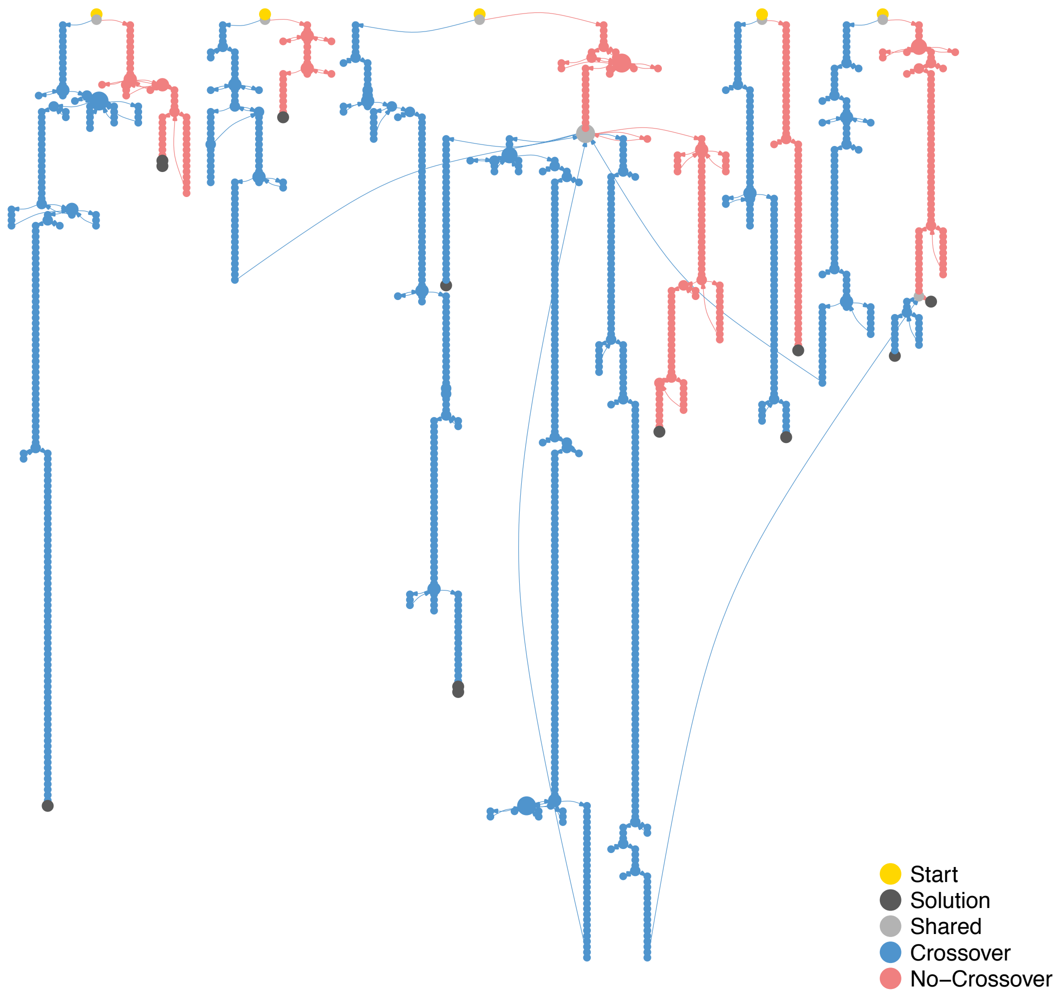


Figure 4.12: Merged STN for DPV. The nodes and edges visited by the crossover variant are decorated in blue and the no crossover variant are in red. Light grey nodes indicate locations visited by both variants. Node sizes are proportional to their incoming degree. The starts of trajectories are represented in yellow and the nodes achieving the fitness threshold (solutions) are shown in dark grey, and both of slightly larger size.

One of the primary aspects that these visualisations convey, which is in line with the statistical analysis results, is that the no crossover variant has shorter trajectories, on average, compared to the original NEAT algorithm (the variant with active crossover). This indicates that the search is more efficient when crossover is disabled, proposing that recombination can generate disruption in this neuroevolution algorithm. This finding is also corroborated by the *avg. path length* metrics of Table 4.5. This is particularly visible in Figure 4.12, where the blue trajectories, pertaining to the crossover variant, are consistently much longer than the red ones. In this domain, the reader may be puzzled by the lack of edges in the representation. This is not the case, and the edges are indeed present. Due to this domain having a far greater population than XOR (see Table 4.3), this produces increasingly more data points. Due to the limited available space of the plotting surface, the algorithm [8] is forced to shrink the distance between nodes, effectively partially hiding these edge components. It was assessed, for the purpose of this analysis, that this does not have a negative impact on the visualisation.

Table 4.5: 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	

Furthermore, it can be clearly appreciated from the graphs that, largely in both domains, a lot of the highest performing solutions (dark grey nodes), achieved by both variants, are different and do not converge to a single (or few) accepted solution. This indicates that in the neuroevolution domain various network topologies and weights compositions provide sufficiently good solutions to this problem.

In Figure 4.11 a high convergence to the same accepted solution can be observed, early in the search stages, a few iterations from the starting locations. These are evident from the variants' trajectories overlapping and creating a multitude of larger light grey nodes. This finding is consistent with the nature of this algorithm, were solutions (neural network topological formations) begin simple and complexify over time. Hence, at early stages of the search process, these solutions have a higher likelihood of being similar, between runs and variants. As search stages progress, this interaction and location convergence, slowly disappears.

From visualising the DPV domain (Figure 4.12), and confirmed by the metrics, we witness a far reduced number of shared nodes. An explanation for this is likely to be that the genotypes to solve this problem are much larger, in this case, exploding the dimensionality to be modelled. This also interestingly shows how the STNs approach, applied to neuroevolution, can be seen as a proxy for the complexity of a domain. Closely tied to the notion that STNs can illuminate the intrinsic diversity of the system under analysis.

A further fascinating observation, can be derived from the merged network of the DPV domain in Figure 4.12. A large, shared node (light grey node in the top middle of the plot) can be observed, this node appears to be visited by the 3rd no crossover (red) trajectory and traversed by other three crossover (blue) trajectories. As the size of nodes is proportionally dictated by the incoming degree, the derived hypothesis is that this shared node is an interesting but sub-optimal neural network configuration that consistently attracts the search process with crossover.

Finally, the plot highlights another compelling aspect of the behaviour examined in our NEAT variants. When the algorithm comes close to finding an acceptable solution, — one that is equal or greater than fitness threshold — at times, two dark grey nodes are being produced. This highlights the fact that, towards the evolution of high performing solutions, slight changes to the network configurations, which result in different nodes, do not impact the performance, and do not disrupt the solutions' fitness. This could be interpreted as good *locality*. That is, when a solution with high fitness is achieved the evolutionary changes occurring in the network are not disruptive to cause a decrease in this solution's fitness; this intuition is further supported by the fact that elitism is deactivated in this analysis.

Network metrics assist us to understand the visualisations further, by complementing them; from these, we deduce the following conclusions. The number of nodes is a good indicator of diversity; intended as the various different network configurations visited in the search space during neuroevolution. Crossover, in both domains, has a larger number of unique nodes (173 and 954), indicating increased diversity. Unfortunately this does not translate to a more efficient neuroevolution algorithm. Instead, this operator leads the search astray from threshold matching/exceeding solutions. In both domains and variants, the number of edges is largely similar to the amount of nodes. This translates to little trajectory overlap across runs. Meaning that only few nodes are revisited by different runs. Moreover, each trajectory follows mostly their own paths — after the early stages of the search process — when most trajectories follow different successful configuration, with fitness values above the domains' threshold.

Regarding the shared nodes (those visited by both variants), visualisation and metrics confirm that these are a relatively smaller proportion, compared to the number of nodes. Specifically, in the harder pole balancing domain, which evolves larger genomes and more diverse topologies-weights configurations.

The above insights allowed us to drill further into "why" and "how" crossover is not an effective operator. This approach allowed us to illuminate search trajectories that are not visible through fitness plots alone. Plots depicting only fitness performance are useful and should be used in conjunction to this technique, nevertheless they only offer a few dimensional perspectives. This search trajectory technique applied to neuroevolution can demonstrate the composition of a neural network structure, the convergence of similar solutions during search, while demonstrating the fitness performance on a universally statistical scale of the search space.

4.8 SUMMARY

In conclusion to this chapter, based on the results, it can be stated that the exploration of NEAT neuroevolution search space, on the classic benchmark domains, using Search Trajectory Network, has been possible and achieved successfully. This has provided sufficient grounds for the inception of a technique, which we will later define as NTNs. This is a novel contribution to the evolutionary computing community that has not been achieved before.

In this research, we have been able to generate a non-mating variant of NEAT, and analysed this against the original algorithm. Our exploration was achieved by statistically testing the variants and examining their search space behaviour via complex network modelling and visualisation. In our assessments, we identified that the no crossover variant was statistically performing better than its counterpart, in the two benchmark tests performed. Statistical significance has also been found ($p < 0.05$).

By examining the search space trajectories, using STNs, we noticed that in these specific testing domain, which are relatively trivial and small, crossover variants produce greater diversity, although, this does not translate into efficiency in finding high performing solutions. This NEAT operator, deployed on specific benchmark domains, such as XOR and DPV, causes disruption to the search process, which leads the trajectories astray. In other vaster domains this diversity could result in improved performance and efficiency, which would need to be tested further.

Another interesting finding, that rose from the examination, is that in these tested neuroevolution search spaces, STNs are able to communicate intrinsic characteristics of a benchmark domain related to the complexity of evolvability requirements. Harder domains, engage evolutionary processes to produce more diverse and increasingly complex solutions, which present lesser location convergence. This is measurable from the shared nodes metrics, especially observable in the pole balancing domain.

In the next chapter, the proposed technique is applied to the study of Novelty Search and the recombination operator in NEAT, for deceptive domains. We will demonstrate how this technique is capable of scaling from simple to harder and more deceiving fitness landscapes.

CONTRIBUTIONS ACKNOWLEDGEMENTS I hereby declare that the substantial portion of the work detailed in this chapter, as documented in the publication [37], is the result of my own efforts. However, it is important to acknowledge the valuable contributions made by my co-author and supervisor, Professor Gabriela Ochoa. Professor Ochoa provided significant insights and guidance, particularly in refining the experimental design and encoding of the signature. Additionally, her expertise in the broader concepts underlying Search Trajectory Networks has greatly enriched the quality of this research.

5 CHAPTER 5 — THE ROLE OF RECOMBINATION IN THE PURSUIT OF BEHAVIOURAL NOVELTY

This chapter outlines the research work published in [10]; this contribution is also detailed in the List of Publications, at the start of this thesis. The formative work [7], discussed in the previous chapter, highlighted that the visualisation technique, used to assess recombination in NEAT, was tested on domains that were perceived to be basic benchmarks. This work suggested a favourable line of enquiry on the that recombination might play in more challenging domains, from a search space standpoint. Among other remarks, there was expressed interest in observing the dynamics of recombination in a different domain setup, thereby making this novel NTNs technique more generalisable and applicable to various realms of neuroevolution. Therefore, the follow-up research intent was to extend previous work and take the examination to a new frontier. The focused shifted to assessing recombination for search strategies aimed at exploring behavioural diversity, while comparing it to objective search. The choice was to select the maze navigation domains, detailed in Section 2.2.2, comparing two maze maps — one simple and linear, the other intricate and deceptive in structure. Results provided insightful perspectives on the interplay between novelty and recombination, highlighting the role of this operator in various search strategies.

5.1 KEY CONTRIBUTIONS AND MOTIVATIONS

This is the *second* contribution chapter of this thesis. It is aimed at extending the work conducted in [7], to more complex and deceiving domains; highlighting the role that recombination plays in neuroevolution systems, aimed at identifying Novelty in the form of behavioural characteristics. The key contributions of this chapter are the following.

- Extend and generalise STNs to model and capture the intrinsic dynamics of NEAT variants, on more complex domains. Strengthening the case for the NTNs visualisation technique.
- Observe the interplay between recombination and Novelty, as mechanisms for exploration and diversity in neuroevolution.

- Enhance our understanding of the role crossover has in neuroevolution systems.
- Offer an STN/NTN network representation of compressed nodes for managing neutrality in neuroevolution.

5.2 DECEIVING DOMAINS, RECOMBINATION AND DIVERSITY

In robotics and neurocontrolled navigation, studies have shown [3, 25, 27, 53] that following a fitness gradient, through the use of objective search, can be counterproductive, and in specific settings, it does not produce the intended result. Those same studies, proposed an innovative search strategy that aimed at producing neural networks (agents) capable of exhibiting diverse and explorative behaviours: Novelty Search [3, 25]. The idea behind it, is that the stepping stones required to reach a specific result are often counter-intuitive, and do not always follow a logical and expected progression of events. In the book "Greatness Cannot Be Planned" [195], the authors manage to elucidate this idea in a clear and intuitive manner. Often, specifically aiming at a particular outcome can lead nowhere near that outcome. This is because indirect, intermediate progressions are necessary to reach a particular solution or creation.

Consider the progression from a simple abacus to a modern digital computer. A scientist inventing the first abacus might find this innovation interesting and useful due to its improved computational abilities. The scientist wants to improve it further, increasing its capabilities by many orders of magnitude. They do not know that, possibly, what they are seeking to achieve, is a computer. Following its objective, the scientist begins to include more beads in the abacus, increasing it to a larger setting, but ultimately it will get stuck. This is because the necessary milestones are not often ingrained in the objective itself. To achieve his underlying intent, the scientist, in fact, would have to work on basic science for years, understanding electricity and electromagnetism, to then consider the invention of vacuum tubes; an important component, which will eventually lead to the intended achievement. This would not be possible, just by selecting an objective and following an intuitive path to it, as deception is ingrained in every solution and each solution's neighbourhood has its limitations.

This philosophy is intriguing, leading us to consider recombination and its role in neuroevolution; specifically in the search for *novelty*. One can think of recombination as the operator that disrupts the search progression. A more powerful perturbation that can diverge the search, away from stagnation;

those confinement regions of the search space. This has been demonstrated in the research outlined in previous Chapter 4 [37]. Another accepted view is that this operator helps high performing parent solutions to come together, to form offspring that would surpass their parents' performance [196]. It is still unclear to which extent this is true and which of these two descriptions best applies to neuroevolution. Therefore, the intent of our work is to propose this network visualisation technique, to assess the interplay between novelty and recombination. Determining, or at least, beginning to suggest, which is the description that best fits crossover for this search strategy, in this realm.

The Novelty Search strategy, as described in Section 2.2.2, is aimed at overcoming deception and achieve a more comprehensive exploration in the space of possibilities. The corpus of study for this is a large one in EC. An initial definition for deception was offered by Goldberg in [49, 197], which relates to the "building blocks" hypothesis. This states that crossover is used to insert small genetic building blocks, to form larger blocks, suggesting that a problem can be defined deceptive, if this combination of lower-order building blocks do not lead to an optimum solution. Determining the hardness of a problem, without running an algorithm on a set of data for that domain is unfeasible [198]. The work of [199] proposes that assessing the extent to which the fitness heuristic aligns with the actual distance to the goal provides a practical measure of a problem's difficulty. This suggests that problem's difficulty often stems from an uninformative objective function. As shown by [3], following the perceivable objective, does not translate in a reasonable objective function. In this work we have considered the deceptiveness (hardness) of a domain according to [3] where: "A deceptive problem is one in which a reasonable EA will not reach the desired objective in a reasonable amount of time.". The authors further reinforce this by stating that a population trajectory, equipped with an objective search strategy would not be able to identify the objective in the search space. As the authors express, other important factor are at play, and the intention is not to simplify and diminish these, but to reinforce the notion that a deceptive objective function is, in fact, a defective one that leads the search astray. A fundamental philosophy, rising form these conjectures, is whether diversity preservation, via crossover, can mitigate the effects of deception.

Delving deeper into the concept explained in Section 2.2.2, the fundamentals underlying this novelty search strategy are related to the notion of *sparsity*. This type of search has to begin by characterising which is the type of descriptors (i.e. behavioural attitudes) we intend to see diverge from the neuroevolved agents controlling the robotic progressions through the maze. These authors touch upon the importance of defining a metric that can conceptualise, in a rigorously mathematical manner, how novel a behaviour might be. The approach uses behaviours collected in a *novelty archive*. These reside in the behaviour space — one of unique behaviours. NEAT traverses the search space to produce neurocontrollers, which in turn, generate said behaviours. In order to calculate novelty of the archive and the population of solutions, occurring during evolution, the authors propose a calculation of sparseness. This is achieved using a clustering algorithm such as *k-nearest neighbour*, the authors proposed to detect those behaviours that are clustered together, to be less novel, and those that are distant from the clusters, high in Novelty scores. Therefore, a current generation, together with the novelty archive, should provide a comprehensive sample of where the search has been, where it is , and how it should progress away from clustered regions at each iteration. This is done through the novelty descriptor space perspective, which is a consequence of the NEAT search in the space of candidate neurocontrollers. This, effectively swaps the search gradient towards that which is *new*, rather than that which is objective-driven.

5.3 COMPLEX NETWORKS CHARACTERISATION

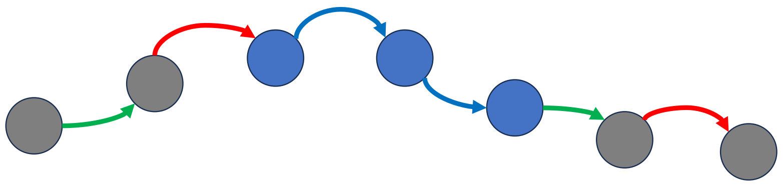
In this specific analysis, the approach adopted for visualising the network models was different than what was outlined in the previous chapter. In this case, the complex networks are constructed using Merged Compressed STNs (CSTNs). Another distinction lies in the generation of these networks, utilising a specific version of the *force-directed* layout algorithm [193]. A form of directed graph structure, which is not in a vertically directional tree-layout format like in previous work [7]. This allows us to increase the number of sampled runs from 5 to 9, generating a better representative sample of the dynamics at play.

In this analysis, the intent remains, to depict the two variants — with and without recombination — for the same search strategy in one visualisation plot. Hence, the need for a network merging technique, which also compresses the networks representations. Our aim here is to generate visualisations categorised by problem's fitness landscape deceptiveness. One set of plots for the *medium* maze set up, and one for the *hard* maze set up (shown in Figure 2.5 of Chapter 2).

5.3.1 *Merged Compressed STNs*

This compressed model variation was first proposed in [10], and was inspired by the works on Local Optima Networks [34]. The attempt is to address the modelling of search spaces, which present large amounts of *neutrality*. Meaning, adjacent portions of the search space that have the same fitness scores. It is important to consider this factor in this analysis, as studies have shown that in NEAT there are several ways of setting neural network weights, which instantiate identical behaviours, primarily due to permuting units or redundant mappings [72].

All definitions related to STNs, as described in Chapter 2, are applicable for this approach. This section, introduces specifications related to CSTNs and Merged CSTNs. A Compressed STN is a directed graph with the following composition $\mathrm{CSTN} = (\mathrm{CN}, \mathrm{CE})$, where CN denotes the compressed node, and CE the compressed edges of the network construct.


(a) Pre-neutrality-compression

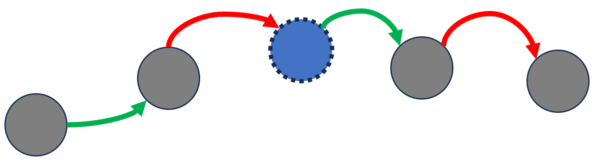

(b) Post-neutrality-compression

Figure 5.1: Illustration of the CSTNs technique for handling fitness landscape neutrality [10].

Figure 5.1 illustrates this concept with a simple example. Before compression, in Figure 5.1a, we observe that the trajectory presents some neutrality. That is, adjacent nodes that have no fitness progressions, in essence, different locations that produce no improvements. To deal with this, and effectively compress the models, we reduce the 3 neutral nodes, which in Figure 5.1a are depicted in blue, to a single node as shown in Figure 5.1b, outlined by a dotted line.

This technique follows similar principles to merged STNs, as described in Chapter 4. Once CSTNs models are constructed for variant/algorithm-problem pairs, we merge these into a single network. Then, progress to distinguish the visualisation planes based on the search strategy and hardness of the problem. Merged graphs consist of nodes and edges from at least one of the algorithm graphs. As in the work of the previous chapter, network attributes are maintained for nodes and edges, indicating whether they were visited by both algorithmic variants, decorated as shared components, or conversely, by only one of the variants. This will become evident once the visualisation are discussed in the following sections.

5.3.2 *Fruchterman-Reingold Force Directed layout*

The chosen layout is the Fruchterman-Reingold [193], aiming to generate data-driven graphs with aesthetically pleasing properties. This layout is part of the `igraph` [194] library of `R` programming language. The idea is to mimic the natural phenomenon of electrically charged particles, described by Coulomb's law. This type of layout works with three main forces: repulsion, springs and network energy. These are illustrated in 5.2, and work as follows. In this model, nodes are analogous to charged particles producing a repulsive force, pushing them away from each other. This force is inversely proportional to the square of the distance, meaning that closer nodes repel each other more strongly than distant ones (red arrow). The spring force will pull nodes closer together. A tightness layout parameter controls the natural length of the spring. A stretched spring means that the nodes are pulled closer to the link end (black spring in illustration 5.2). On the other hand, when the spring is loose, nodes are pushed further away from the link end. Lastly, a general *energy* for the network (blue box) is a parameter added to the system, where each node is given a random direction of movement. This layout simulation iterates until the system finds a state of equilibrium.

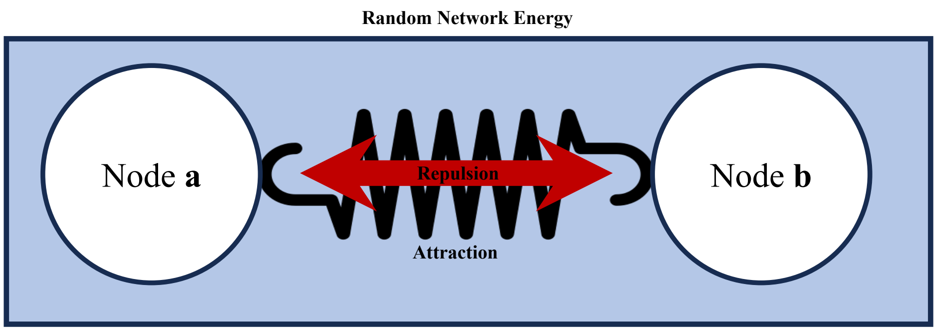


Figure 5.2: Illustration of the forces involved in the Force Directed layout.

5.4 EXPERIMENTAL SETTINGS

The experiment is based on the maze navigation domain used in [3]. This problem was previously explained in Section 2.2.2 and visually depicted in

Figure 2.5. For completeness, we briefly reiterate how this domain works and detail how the neurocontroller agents form.

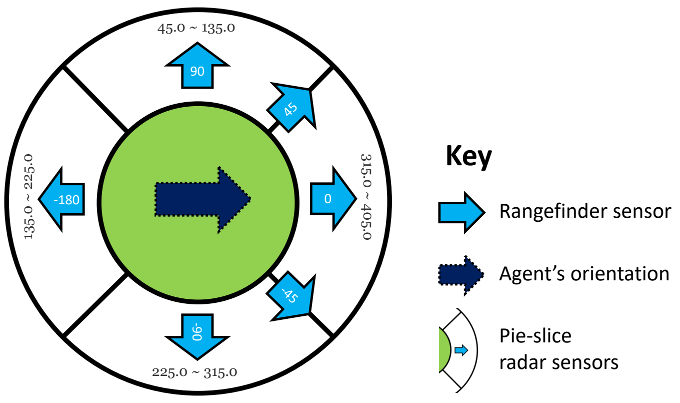


Figure 5.3: Physiognomy of the maze navigating agent. Illustration adapted from [9], derived from [10].

The task in this domain is to place the neuroevolved agents at the starting point of the maze (dark-gray dot in Figure 2.5), letting these navigate the maze, progressing until the exit point is found (yellow dot). As already stated, the domain has two difficulty levels. The more linear maze presents a low to medium level of difficulty in navigating to the exit. The harder and more deceptive one, includes "culs-de-sacs" that create traps, complicating the identification of the goal.

Figure 5.3 shows the agent physiognomy, featuring six rangefinder sensors for obstacles detection and four pie-slice radar sensors, acting as a compass, to detect the goal orientation. Pie-slice labels indicate the degree range of the compass. Arrows show rangefinder sensors positions, both in reference to the agent's orientation.

This experiment explores two algorithmic variants: recombination and no-recombination. Each of the variant is equipped with a fitness-based search and a Novelty Search strategy. In this analysis we name them: *Novelty_X, Novelty_NoX, Fitness_X and Fitness_NoX*. The function for these strategies are explained as follows. The original NEAT algorithm is guided by a standard objective function, which is used to assess genotypes. Equations 5.1 and 5.2 detail the loss and fitness functions, respectively. This experiment associates the fitness to genotypes based on the behaviour these neurocontrollers produce.

$$\mathcal{L} = \sqrt{\sum_{i=1}^{2} (a_i - b_i)^2} \tag{5.1}$$

This Equation is used to calculate the Euclidean distance between the agent's simulated location with respect to the exit point of the maze (objective). $\mathcal{L}$ represents the root-mean-squared error function used for proximity evaluation, where a is the position of the agent at the end of simulation and b the fixed maze exit location (expressed as 2-dimensional coordinates). With the loss function described above, we can now detail the equation for fitness.

$$\mathcal{F} = \begin{cases} 1.0 & \mathcal{L} <= R_{\text{exit}} \\ \mathcal{F}_n & \text{otherwise} \end{cases} \tag{5.2}$$

In Equation 5.2, R_{exit} represents the 0.05 radius of the exit circumference. This is the solution threshold which is set to the fitness score of 0.95. This means that any results falling within that threshold will be regarded as a whole score of 1 (100%).
$\mathcal{F}_n$ is a necessary normalisation, which is calculated as follows: $\mathcal{F}_n = \frac{\mathcal{L} - D_{\text{init}}}{D_{\text{init}}}$, where D_{init} denotes the agent's initial distance to the maze goal.

For the novelty score calculation, this search strategy uses a variant of the k-nearest neighbour as a metric of sparsity for the behaviour solutions. For this, we redirect the reader to Eq. 2.4 presented in Chapter 2. We remind that in our implementation, which is inspired by [9], to simplify the calculation, behaviours vectors are comprised of just the agent's trial end coordinates $(j = n)$. Instead of the entire movement trajectory.

Table 5.1 displays all parameters utilised for this experiment. All values remain consistent with the standard parameters used in NEAT. They also do not differ between maze difficulty, with the exception of *solver time steps*. For the hard maze, the test done in our studies, using this specific implementation [9], had shown that 400 time steps were not a sufficient allowance to reach the map's goal. This was increased to 600. The k parameter is exclusively utilised for the sparseness calculation, necessary for the Novelty Search strategy setup.

5.5 ANALYSIS RATIONALE

This analysis aligns with the rationale detailed in Chapter 4. The experiment we carry out reproduced what was done in [25]. To achieve this, we leverage the implementation of Novelty Search for the domain used [9]. The analysis begins with the statistical test on the performance of variants, and then moves to the complex networks modelling and visualisation using STNs.

Table 5.1: NEAT parameter values used. The k parameter (from k-nearest neighbours) is relevant only for the novelty search variants.

Parameter	Value
Population size	250
Maximum generations	1,000
Solver time steps (medium maze)	400
Solver time steps (hard maze)	600
Solution fitness value	1.00
Fitness threshold	0.95
Bias range	[-30, 30]
Weight range	[-30, 30]
c1	1
c2	1
c3	3
Probability add link	0.1
Probability add node	0.005
k (k-nearest neighbours)	16

5.5.1 *Statistical analysis*

For the statistical tests, 30 runs were generated for each variant-search strategy pair, on each of the problem difficulty levels. A neurocontroller that achieves a fitness of 1.0 (100%) is deemed to have solved the maze problem, as it successfully navigated the maze to reach its goal, or at least got close enough to enter the threshold range. Algorithmic variant performance was studied based on three criteria: (i) the success rate, which is the ratio of runs reaching a solution, (ii) the best fitness achieved at the end of a run, averaged over 30 runs, and (iii) the number of generations necessary to reach a solution (for successful runs), averaged over the number of successful runs for each variant. Furthermore, the analysis encompassed the study of the best fitness values and the number of generations across all runs.

5.5.2 *Merged CSTNs analysis*

For the Search Trajectory Network analysis, a sample of 9 (seed enumerated) runs were selected from the ranked fitness performance of total runs executed. Three were taken form the best, three from the worst and three from the intermediate ones. This selection ensured fairness and a comprehensive representation of the data, while maintaining the visualisation manageable and informative.

In complex network visualizations, there exists an empirical trade-off between providing sufficient information and avoiding information overload. Cluttered visualizations, which incorporate numerous runs may offer statistical robustness but often sacrifice the nuanced analytical insights achievable with fewer runs, which can better communicate the dynamics of neuroevolution variants.

Visualisations were created using the previously described Merged CSTNs technique, displaying crossover and no-crossover variants in the same plot. Plots were split based on search strategy and hardness of the problem (maze).

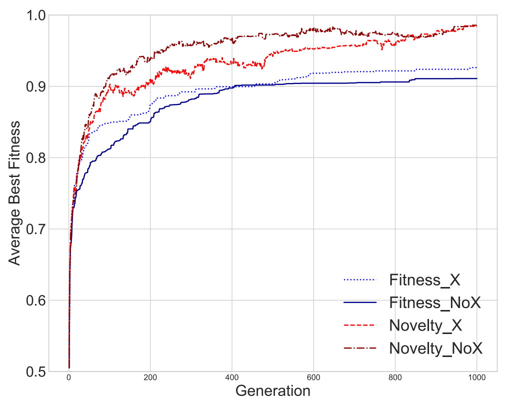
After generating the networks, a further examination of their structural characteristics was conducted by computing accepted metrics [35]. The most common ones are the number of nodes (indication of diversity) and edges. Other fundamental ones in this approach include the length of paths, community structure, degree distribution, and centrality of nodes [17]. We limit our examination to a simpler selection of six metrics, assessing the structure of trajectories, bringing insight into the behaviour of studied search variants. The total number of *nodes*, indicates the amount of search space exploration. The *solutions* metric relates to those nodes that reach the target, this indicates how many different locations solve the domain. A ratio of compressed nodes to total number of nodes is computed. This reflects the amount of neutrality in the explored search space, that is, the proportion of adjacent solutions with the same fitness. A higher value for this indicates, high neutrality. The variance of the neutrality metric is not calculated, which could be an additional inclusion to explore in future work where neutrality is calculated. This metric is computed as follows: $\text{c-ratio} = 1.0 - \frac{|CN|}{|N|}$. *w-edges* refers to the number of worsening edges, which are those that link a higher fitness node to a lower one. This indicates the amount of non-greedy exploration during the search process. Lastly, the number (*n-path*) and length (*p-length*) metrics signify the shortest paths from start nodes to solutions in the CSTNs. These explain the reachability of solutions. The last two metrics are not defined, if no solution is achieved.

Decorators in STNs are crucial for enhancing the information delivery within a complex network representation. Nodes and edges are crucial components, decorated to emphasise relevant features. Compressed node sizes are proportional to number of individual nodes (locations) they contain. The start of trajectories are represented as dark-grey squares, the end nodes reaching a solution as yellow circles, and suboptimal end nodes as dark-grey triangles. Nodes visited by both *X* and *No_X* variants in the merged CSTN are represented in light grey. Finally, bright green lines are used to highlight worsening edges. These are relevant to appreciate the explorative (non-greedy) dynamics of Novelty Search.

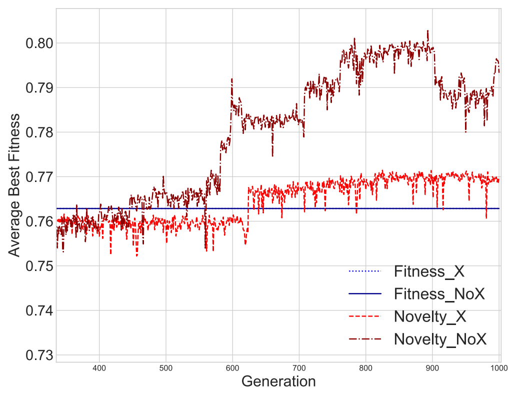
5.6 RESULTS AND DISCUSSION

The discussion begins with a statistical performance analysis, followed by CSTNs analysis involving visualisation and metrics.

The plots presented in Figure 5.4 provide a general overview of the convergence behaviour of all the variants studied. Based on the plots and performance metrics in Table 5.2, Novelty Search emerges as the strategy with the highest success rate and average best fitness. This confirms the findings presented in [3]. Our results differ from the original ones, as the average best fitness for all variants remains substantially lower; despite all parameters and algorithms being faithfully reproduced.



(a) Medium maze - all variants



(b) Hard maze - all variants (magnified) map

Figure 5.4: Convergence plots showing the averaged fitness performance over 30 runs for all variants tested in both maze domains. The convergence curves are shown from generation 300 to highlight salient differences in the Novelty Search strategy for the hard maze.

The introduction of a no-crossover variant, in our experiments, allows us to perceive the usefulness of crossover in different search strategies, for deceiving domains. Similar success rates can be seen for the medium map between *Novelty_X* and *Novelty_NoX* (left side of Table 5.2) nonetheless, similarly to what was witnessed in the work of Chapter 4, the variant without crossover reaches the solution with fewer evaluation on average. The non-parametric Mann-Whitney statistical test revealed statistically significant differences ($p < 0.01$) between the strategies. Indicating, once again, that for the novelty search strategy, the use of crossover negatively impacts performance. On the other hand, when looking at fitness-based search on the medium map, we observe a higher success rate and best average fitness, and lower number of evaluation to reach the goal, when the recombination operator is active. Despite the fact that these distributions are not statistically significant, this finding is interesting and unexpected, as it seems that crossover, in this scenario helps to generate neurocontrollers with diverse behaviours, capable of solving the maze better.

In the hard maze, both variants (X and NoX) of fitness-based search are not capable of solving this domain. The success rate is nil, the best fitness reached in all the trials is consistently 0.7629, indicating the search process was trapped in a potential local optimum, and was not able to escape such region. This is clearly visible in Figure 5.5, where the best evolved neurocontrollers' paths, for each of the NEAT variants, are depicted in the hard maze domain.

In this case, the distributions difference for the number of generations, each variant took to find this local optimum, is not statistically significant. This suggests that the positive effects derived from the inclusion of recombination is minimal and possibly inconsequential, for the fitness-based variant, in this maze setup. The result does not conclusively determine the usefulness of this operator for fitness-based search.

One of the main takeaways, which can be derived from the convergence plots shown in Figure 5.4 is that Novelty_NoX, the variant without recombination, performs better than all other variants. Figure 5.4 offers a closer look at the plot detailed in [10]. This is to highlight the differences between the *X* and *No_X* Novelty Search variants, which approximately occur at generation 400 of 1000. Novelty_NoX progresses steadily, increasing fitness up to the 900th iteration, reaching 0.80 of 1.00, returning then to lower levels, concluding approx. at 0.789 of 1.00. On the other hand, Novelty_X increases slightly from the point of divergence to around 0.77, on which it remains steadily until the end of run.

Table 5.2: Performance metrics of the best fitness and generations average values with standard deviations in parenthesis. Average generations are computed for the successful runs only.
S.R. = Success Rate, B.F. = Best Fitness and G. = Generations.

	Medium map		**Hard map**	
	Crossover			
	Novelty_X	Fitness_X	Novelty_X	Fitness_X
S.R.	86.67% (26 runs)	20.0% (6 runs)	3.33% (1 runs)	0.0% (0 runs)
B.F.	0.9842 (0.0422)	0.9264 (0.0387)	0.77 (0.0437)	0.7629 (0.0)
G.	422.54 (309.89)	417.67 (361.71)	623.0 (0.0)	-
	No Crossover			
	Novelty_NoX	Fitness_NoX	Novelty_NoX	Fitness_NoX
S.R.	86.67% (26 runs)	6.67% (2 runs)	13.33% (4 runs)	0.0% (0 runs)
B.F.	0.9857 (0.0376)	0.9109 (0.0324)	0.7933 (0.0855)	0.7629 (0.0)
G.	265.38 (243.63)	583.0 (265.0)	719.75 (165.6)	-

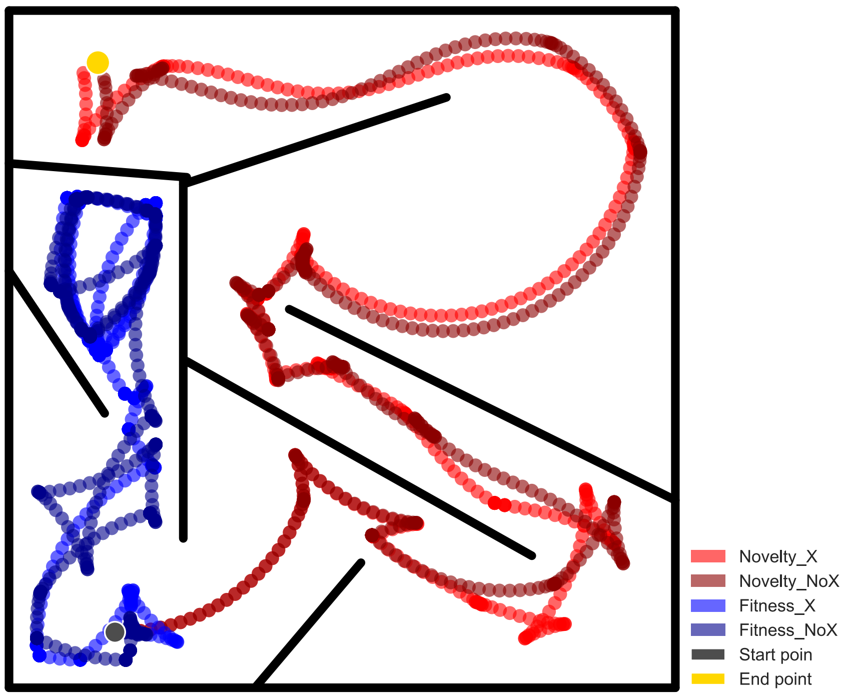


Figure 5.5: Navigation paths of the best genomes evolved by each NEAT variants.

Figure 5.5 visualises exploratory paths in the hard maze, showcasing the best-performing (fitness) neurocontrollers, produced for each variants out of the 30 runs. This view of the Cartesian behavioural space was particularly helpful for assessing the efficiency of variants, and to better comprehend the navigation in local optima regions of the maze. Aside from the stagnating behaviour of both fitness-based search variants, which was already pointed out, we can visually contrast the behaviours differences of Novelty_NoX (dark red) compared to Novelty_X (bright red). It appears that Novelty_NoX is faster and more effective in identifying the left turn required to reach the diagonal channel of the maze, which leads to discover the goal (maze exit location).

In this experiment, the analysis was extended further, generating violin plots to assess the performance of variants in each of the mazes. Visualisations in the following figures depict the best fitness at the end of the run, and the number of generations needed to reach the maze solution. The plots utilise split violins, visually distinguishing each search strategy and its crossover and no-crossover variant. These visualisations are described starting from the medium maze.

5.6.0.1 *Medium maze*

In Figure 5.6, Novelty Search is depicted on the left with two shades of red, representing the variants with and without crossover. Fitness-based search is shown on the right with two shades of blue. Additionally, black dots overlay the 30 individual data points for each variant. Distributions confirm that the final best fitness values for Novelty Search consistently surpass those found in fitness-based search. The Novelty Search variants (X and NoX) show similar distributions, with Novelty_ NoX slightly skewed towards higher values, having also a tighter distribution. For fitness-based search, a larger concentration resides nearer higher values for the crossover variant (bright blue).

Figure 5.7 depicts the distribution of generations needed to reach a solution for all variants. The individual 30 data points for each variant are represented as black dots. The domain has a maximum of 1000 generations, if the solution is not reach in this time-frame, the black dots are represented above the dotted line, signifying unsuccessfulness. Here, lower values indicate better performance, as this means a more efficient algorithm. The plots demonstrate a notably faster convergence of the novelty variants compared to the fitness-based ones. For the novelty search strategy, the no crossover variant (dark red, Figure 5.7) shows a tighter distribution towards lower generation values than the variant with crossover (bright red). This insight

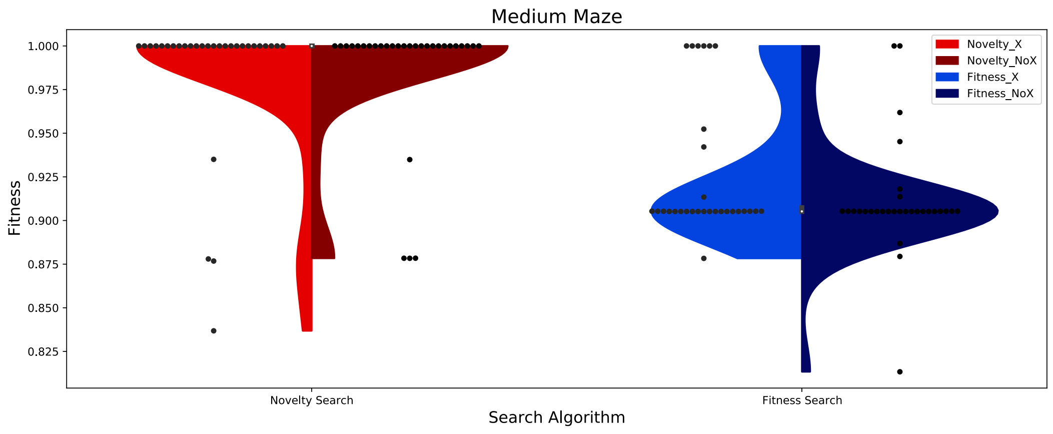


Figure 5.6: Distributions for best fitness values on the *medium map* for all variants. Swarm plots overlaid to demonstrate the individual data points for each variant.

supports the hypothesis that recombination slows down the progress for novelty search. The situation is reversed for the fitness-based variant. Here, crossover (light blue), shows a tendency towards lower values. However, most runs are unsuccessful for both fitness search variants, as indicated by the majority of (black) data points above the dotted line. It appears that crossover can be of some use for fitness-based search on this maze, although this strategy is not competitive against Novelty Search.

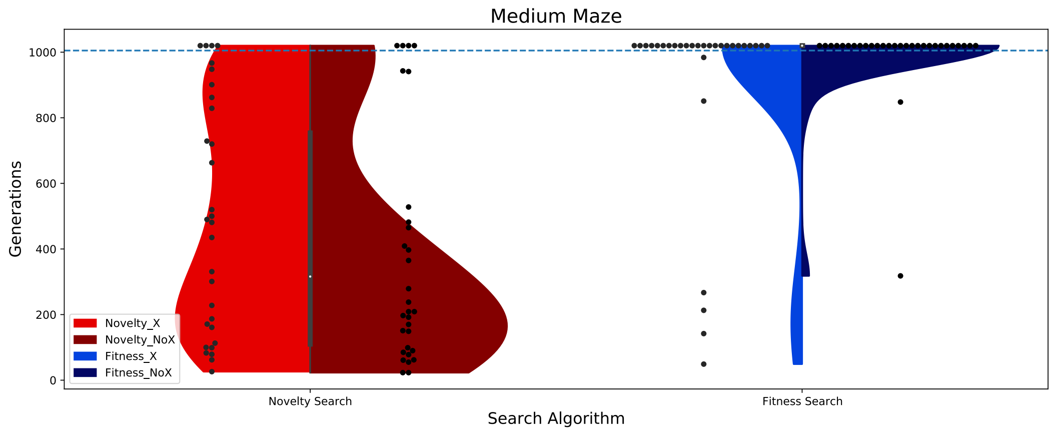


Figure 5.7: Distributions of the generations needed for each variant to reach a solution on the *hard map*. Swarm plots overlaid to show the individual data points for each variant — those that lie above the dotted line represent runs that failed to reach a solution.

5.6.1 *Hard maze*

Figure 5.8 shows violin plots distributions of best fitness values for the four NEAT variants. Once again, these distributions confirm that for this map, the final best fitness values for Novelty Search are higher than those found by objective led variants. However, for both search strategies, the distribution concentration hovers around the local optimum, with a fitness value of approximately 0.76 out of 1.00. Novelty Search has a few points above (and below) this value, whereas for fitness-based search all the data points reach a fixed fitness corresponding with the local optimum, which has been found trapping the search process. Novelty_NoX exhibits a larger number of successful runs for this map compared to the crossover variant.

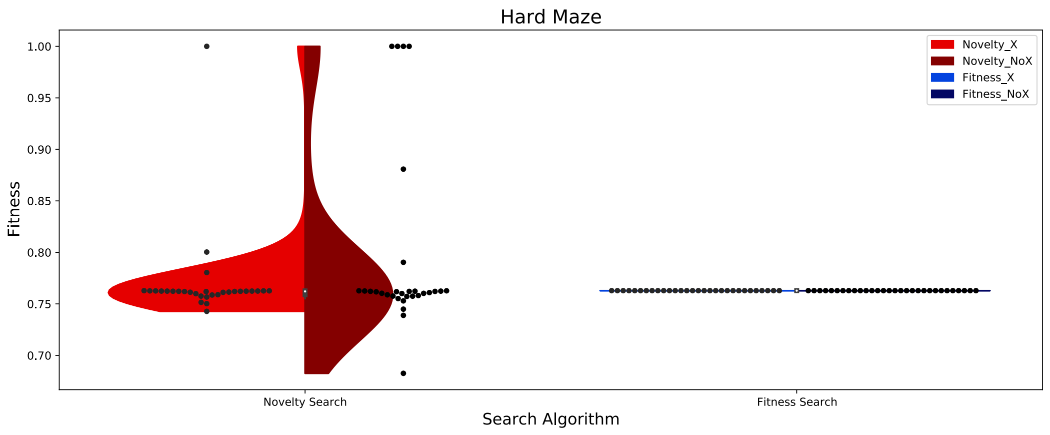


Figure 5.8: Distributions for best fitness values on the *medium map* for all variants. Swarm plots overlaid to demonstrate the individual data points for each variant.

In terms of generations necessary to reach a solution, Figure 5.9 illustrates these distributions for all NEAT variants executed on the hard maze. 30 independent runs are overlaid as black dots. We can observe that all runs were unsuccessful for fitness-based search (right-hand plot), as all black dots are above the segmented line (1000 generations). Only one run out of the 30 for Novelty_X was successful, while Novelty_NoX saw 4 runs out of 30 reaching a solution, with a number of generations that varied from over 550 to almost 1000. These results suggest that crossover can be detrimental for the novelty search strategy.

5.6.2 *STNs analysis of search strategies dynamics*

The search trajectory analysis was carried out using the same principles for signature and models generation outlined in the previous Chapter 4.

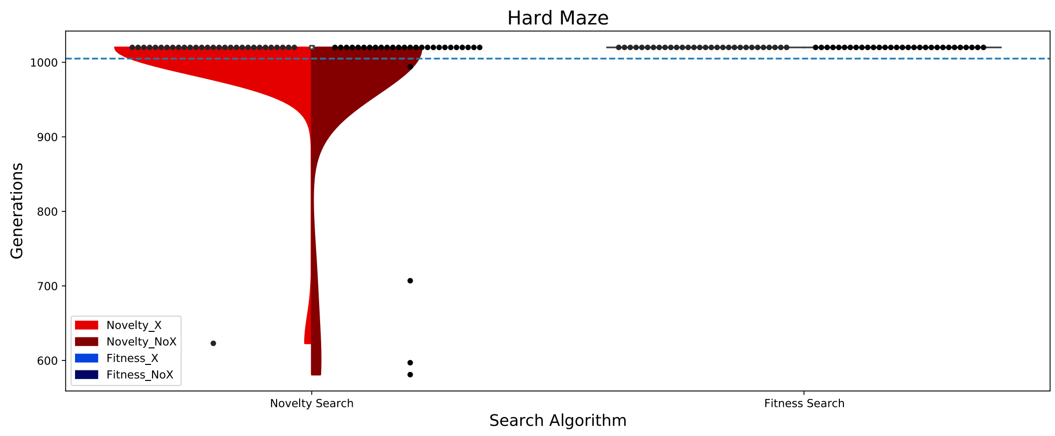


Figure 5.9: Distributions of the generations needed for each variant to reach a solution on the *hard map*. Swarm plots overlaid to show the individual data points for each variant — those that lie above the dotted line represent runs that failed to reach a solution.

A sample of 9 out of the 30 runs was selected for visualisation. These are representatives of the performance distributions of successful runs. Three run ranked from the best, three from the worst and three intermediate runs.

Table 5.3 presents the computed metrics deriving from the networks. These choices, alongside the explanation of what they are, have been detailed in section 5.5.2. Metrics are fundamental for network analysis, as they assist the reading of visualisations and they help to further corroborate visual assumptions. Based on these metrics, the following findings were derived.

The number of nodes is indicative of the exploration/diversity that an algorithmic is able to produce. For Novelty Search this metric is always consistently larger than for fitness-based search; meaning that this strategy works and can produce a higher diversity compared to the canonical approach. Another confirming aspect of the superiority of this strategy is the fact that it is able to reach more unique solutions than fitness search. Furthermore, the number of worsening edges, which offers insight into non-greedy explorative properties of an algorithm, is also larger for this strategy. Regarding the variants within each search type, Novelty_NoX is the sole variant achieving multiple solutions in both the hard and medium maze setups. The number of paths to these solutions are more for this variant and they are shorter on average, highlighting efficiency; challenging the notion that crossover is useful for Novelty Search. In regard to fitness instead, in the medium maze configuration, the variant with crossover produced more solutions and paths than its No_X counterpart. This suggests that recombination plays a role for exploration in fitness-based search. Further enquiries should be conducted to examine these results, as the same could not be confirmed on the hard maze; fitness search failed to solve the most deceptive (hard) domain. In the hard maze, the compressed ratio (*c-ratio*) is very large for fitness search. This signifies that the algorithm traverses several individual sub-optimal solutions with the same fitness (large neutrality). Contrastingly, the c-ratio for Novelty Search is relatively low, indicating a wider exploration of candidate solutions with greater diversity.

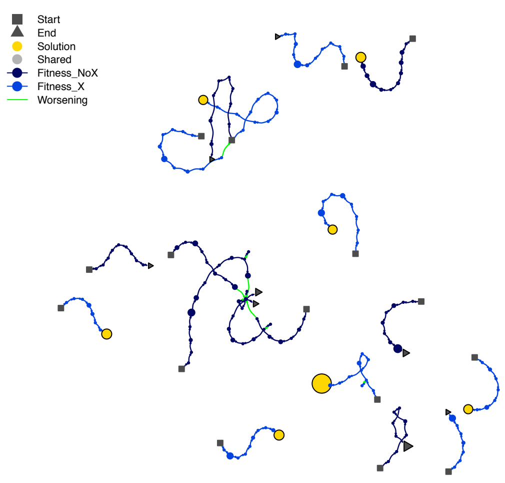
Table 5.3: STNs metrics for all variants in both maze configurations.

	Medium Maze				Hard Maze			
	Novelty		Fitness		Novelty		Fitness	
	X	NoX	X	NoX	X	NoX	X	NoX
nodes	322	798	185	190	1523	1065	138	146
solutions	48	520	39	2	1	12	0	0
c-ratio	0.20	0.66	0.48	0.50	0.09	0.09	0.81	0.77
w-edges	155	163	2	7	826	555	0	0
n-path	8	9	6	1	1	2	0	0
p-length	21.25	19.67	11.33	11.00	43	91	NA	NA

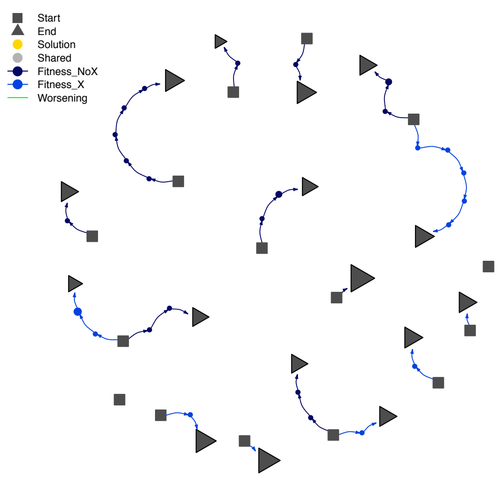
In Figure 5.10, we observe the network plots related to the fitness search strategy of both maze setups. Similarly, Figure 5.11 shows the variants for Novelty Search. Note that these represent *merged CSTNs*, showcasing both recombination variants in the same network plot with compressed fitness neutrality.

In Figure 5.11a and 5.11b we can observe longer trajectories than fitness search (in terms of nodes and edges). Novelty Search trajectories typically feature higher counts of worsening (bright green) edges, denoting a broader, non-greedy exploration approach. Additionally, these variants' trajectories (Figure 5.11a and 5.11b), are able to reach a larger number of yellow solution nodes, which are of large size due to the incoming degree strength achieved from these trajectories. In the CSTNs space, trajectories may overlap, forming similar paths to solutions. This can create the visual impression of observing a single trajectory entering larger yellow solution nodes.

Comparing variants within a strategy, let us observe that Novelty_NoX, which are the dark-red trajectories visible in Figure 5.11a and 5.11b, reach a greater number of solutions (yellow nodes), which are of larger size, compared to the Novelty_X variant in bright red, which has recombination. The opposite occurs in fitness search for the medium maze, seen in Figure 5.10a;



(a) Medium maze



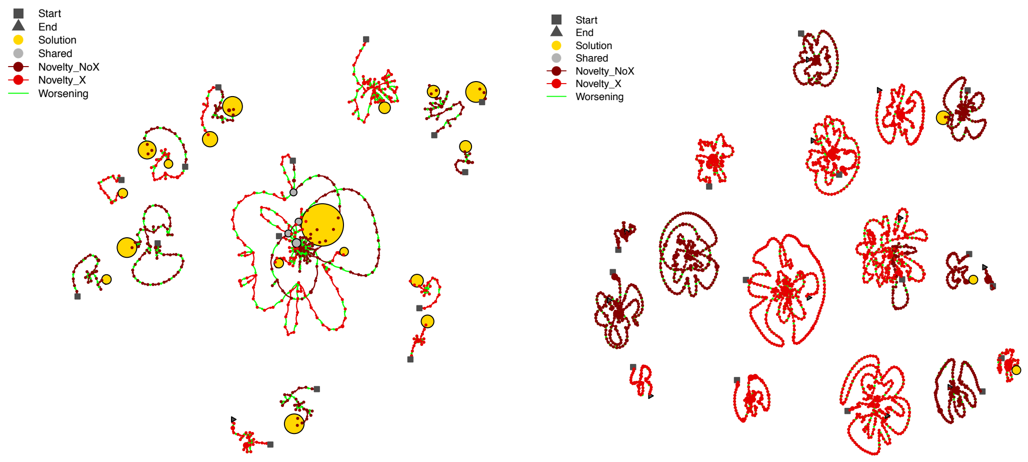
(b) Hard maze

Figure 5.10: CSTNs for crossover and no-crossover variants of *fitness search* for both medium and hard maze configurations.

the crossover variant, depicted in bright blue, is able to reach a higher number of solutions compared to its no-crossover counterpart.

The fitness search trajectories of the hard maze shown in Figure 5.10b are much shorter than the other networks, featuring no edges to a few edges at most. All trajectories tend to terminate in sub-optimal locations of the search space, illustrated by the large size dark-grey triangles. This suggests that, as seen in the metrics and statistical results, trajectories quickly reach local optima, with several different neural networks configurations producing the same sub-optimal fitness.

Another noteworthy observation can be derived from Figure 5.11b, in this plot of the hard maze configuration the trajectories of Novelty_X and Novelty_NoX appear to traverse separate paths in the solutions search space. This is interesting, as this dynamic differs from the medium maze (Figure 5.11a) where trajectories intersect and share similar paths of the search space. Similarly as in the DPV domain results of Chapter 5 this behaviour could be attributed to the fact that STNs are capable of demonstrating the intrinsic characteristics of evolvability requirements related to a search space landscape. Little interaction between solutions and independent trajectories may demonstrate the inherent hardness (deceptiveness) of a problem domain.



(a) Medium maze

(b) Hard maze

Figure 5.11: CSTNs for crossover and no-crossover variants of *Novelty Search* for both medium and hard maze configurations.

5.6.3 *Results discussion*

We continue from previous work to assess the role of recombination in neuroevolution algorithms, with specific attention to evolutionary dynamics that leverage the generation of behavioural diversity, for search gradients driven by novelty versus fitness. In the work outlined in the previous Chapter [7], the problems examined were simple reinforcement learning task, where the application of this visualisation technique helped to assess the search space dynamics. In this case our interest progressed, to produce an assessment on the applicability of STNs to different search strategies, inquiring the usefulness of recombination in harder, and more deceptive domains. It is due to this deceiving maze structure that the fitness gradient for this type of search is provably insufficient, which requires the use of novelty-driven fitness.

The added knowledge which is derived from the STNs analysis of neuroevolution can be used to inform any future variations of this algorithm; by testing components in isolations, and deciphering whether operators can be applicable in the search strategies for specific domains. Additional, the structure of the search spaces and the trajectories quantity, length and behaviour, can also inform if a certain search strategy is necessary or not for a domain. Parameters tuning of novelty criteria could also be analyses in-depth through this proposed technique.

The statistical analysis on fitness performance and the STN assessment work presented in the previous chapter [7], complemented each other, and together offered comprehensive but also independent findings. These findings suggest the technique's applicability to NEAT and potentially broader neuroevolution search spaces.

One of the outcomes identified from this work is that each of the analyses is useful, and can be performed in isolation. Statistical analysis is key, as it helps to form a comprehensive picture of the performance behaviours for evaluated algorithmic systems, such as the four variants proposed in this work. Performing only this type of examination could have erroneously suggested that fundamentally different solutions were similar, simply because they achieved the same *fitness* results. Contrastingly, STNs assist us in exploring further, by drilling into genomes' characteristics, offering a virtually modelled picture, faithful to the inherent optimisation search processes; which are often highly dimensional and not visible to the naked eye — such aspects that would be otherwise missed from a statistical viewpoint.

Finally, performance analysis is indeed essential to highlight those algorithm or variant that are capable of performing best in particular settings, configurations or ablation of operators. The technique of STNs has proven successful in neuroevolution, and NEAT in particular, deciphering whether performance is in essence comparably equal, or not. STNs assist us further in our examination, by identifying "why" a specific solution is different (or similar), from the perspective of evolutionary search space dynamics. For example, are transitions in the search space, present in shorter trajectory, converging and interacting with increasingly more nodes, reaching the solution faster, as they explore niches of the optimal neural network compositions (Figure 5.11a)? Are the simpler, isolated, successful trajectories comparable to longer and more converging trajectories (Figure 5.11b)? What is the recurrence of stalling trajectories failing to identifying the solution, can the path length and trajectory shape inform us on the problematic nature of the search strategy Figure 5.10b?

5.7 SUMMARY

In conclusion to this chapter, it is possible to confirm that applying Search Trajectory Networks modelling and visualisations, to further understand different search strategies, in hard and deceiving domain, was achieved successfully.

This work focused primarely on the role of recombination. A statistical comparative analysis was conducted to assess variants with and without recombination operator. The assessment was made on two search strategies, the canonical objective-driven method, solely dictated by a fitness score, versus a strategy that searches for behavioural novelty. A specific adaptation to the standard STNs approach was generated, so that the technique could be used to model complex neural genomes found in search spaces, that present large neutrality regions; it is known that several ways exist to configure a neural network which instantiate the same behaviour.

This analysis confirmed the superiority of Novelty Search, over an objective-driven strategy, in evolving neurocontrollers capable of operating in deceptive maze structures, constituting hard problems. Amongst interesting finding that surfaced from this work [10], it was found that crossover appears to play a positive role in fitness-based search. On the other hand, it is less clear to perceive the same when searching for novelty. On these specific maze navigation domains, the results derived, suggest that Novelty_NoX, the variant without recombination, is very effective at reaching many good solutions. By deactivating crossover, a greater number of trajectories reach successful neural network designs, able to generate thriving behaviours, doing so with shorter trajectories overall. Nonetheless, recombination paths, seem to conduct a more wide search space exploration, suggested by the longer trajectories, confirmed also by the metrics. This insight, may lead to consider the recombination operator as a catalyser of additional population diversity, which can potentially be useful in yet more complex and deceiving domains, where an objective function can struggle to represent the domain state. The question that rises here, which could form the focal point of future research, is whether extremely strong permutation can demonstrate, through STNs, the same level of diversity and trajectory dynamics.

In evolutionary computation, there is a wide consensus that recombination can be useful for merging high performing solutions together. If done correctly, this can be the case, although this operator could also be seen as a disruptor of good solutions, which may require smaller mutations to optimise. In essence, an algorithmic component that splits high performing solutions in a way that diverges the search gradients. This could in turn produce diversity, which for deceiving problems, can be necessary. Novelty Search is equipped with principles that assist in the discovery of novel behaviours and high-performing solutions. Through this research results have shown that the removal of crossover from the NEAT algorithm, did not impact the performance of this search strategy, but improved it instead. For fitness-driven search, there appears to be benefits of having crossover active in the system, for neurocontrollers deployed to solve these maze navigation tasks.

Furthermore, the STNs modelling and visualisation tool applied to these neuroevolution search spaces allowed us to perceive the similar and contrasting behaviours of these strategies traversing domains of different levels of deceptions. Particularly, the tool allowed us to observe further dimensions, not observable in the fitness convergence plots. The simplicity of the trajectories, and length for the fitness strategy in the medium maze, compared to the longer, more converging and interconnected trajectories, visible in the hard maze. These, together with the neutrality assessment of the c-ratio metric, offered a more detailed picture of the traversed search space and neural network compositions. The trapping regions and lengths of the search were universally visible in the STNs of fitness-based search for the hard maze, which were in strong contrast with the more complex trajectories taken by novelty search. Within this domain we have observed that the novelty strategy, deprived of crossover, was identifying more solutions than its counterpart while being more explorative, signalled by the increased path lengths, which did not show increased levels of neutrality (c-ratio).

These observations lay the groundwork for further exploration of the interplay between Novelty and recombination in evolutionary algorithms. It highlighted some key intrinsic characteristics, which if further examined, could tell us more about the search for diversity and high performing solutions in the realm of neuroevolution.

CONTRIBUTIONS ACKNOWLEDGEMENTS I hereby declare that the substantial portion of the work detailed in this chapter, as documented in the publication [10], is the result of my own efforts. However, it is important to acknowledge the valuable contributions made by my co-author and supervisor, Professor Gabriela Ochoa. Professor Ochoa provided significant insights and guidance, suggesting experimental settings. Additionally, her expertise in the broader concepts underlying Search Trajectory Networks has greatly enriched the quality of this research. They have additionally contributed in refining and perfecting aspects of the STNs visualisations. My co-author and second supervisor Dr. Jason Adair has also contributed to this research, and has been a valuable source of assistance in the statistical visualisations, related work section, and the overall proofreading.

6

CHAPTER 6 — THE BEHAVIOUR SEARCH SPACE UNDER NEUROEVOLUTION TRAJECTORY NETWORKS OBSERVATION

This chapter outlines the research published in [11]. This significant contribution is also listed in the List of Publications section at the beginning of this thesis. Here we dwell on the work that led to the inception of Neuroevolution Trajectory Networks (NTNs), as an associated technique. This work marks the first time we have recognised and established the notion and definitions of this methodology. The main idea is that this technique is specifically leveraged to assess neuroevolution search spaces. This approach establishes the study of neuroevolution dynamics and characteristics from the perspective of search space, diverging from stand-alone, performance-driven analysis. However, the latter (performance-driven analysis) plays a vital role in presenting a comprehensive view of algorithm performance. When combined with our proposed technique, it offers a clearer picture of the mechanisms within neuroevolution algorithms. Suggesting why certain phenomena occur rather than simply highlighting these in therms of fitness achievement. This work aims at providing explainability to the world of neuroevolution.

6.1 KEY CONTRIBUTIONS AND MOTIVATIONS

This is the *third* contribution chapter of this thesis. It is a fundamental one, as it details the work that lead to the key notion of NTNs, the technique that spawn from Search Trajectory Networks, being the principal theme of this thesis. Here, we propose a break from the general STNs parent technique, moving on to explore traits that are uniquely relevant to the neuroevolution realm. As outlined in Section 2.1.3, the specialisation of NTNs and its definition are related to the modelling and visualisation of characteristics that are prevalent and solely found in Neuroevolution and related search spaces. Specifically focusing on the notion of Behavioural Characterisation (BC), diversity generation and exploration, in relation to topological complexity of evolved neurocontrollers. The specific motivations and contributions of this work are detailed as follows.

- Offer, for the first time, NTNs as a network visualisation technique that aims at modelling Neuroevolution search spaces and algorithm mechanics.
- Use NTNs, as an advanced tool to examine divergent/explorative search strategies and behavioural characterisation, in the evolution of neurocontrollers.
- Assess the interplay between topological structure, behavioural diversity, and divergent search.

6.2 ILLUMINATING BEHAVIOURAL CHARACTERISATIONS

This research has enabled an advanced exploration of Behavioural Characterisations — as explained in Chapter 2. BC specifically catalysed the shift to an NTN technique, as BC is a specific trait of Quality Diversity algorithms and in this experiment it is specifically fuelled by neuroevolution. The search for novelty in BC is integral to optimising divergence by seeking diverse features. As it was seen in the previous chapter, there is a necessity to abandon objectives and convergent search strategies, because in some domains these are ineffective and possibly even *defective*. Often, this shift was inspired by the phenomena observed in evolutionary biology [72], indicating that a high diversity of unique niches leads to superior fitness and survival characteristics. The BC concept was initially introduced by [54], focusing on a specific set of evolutionary algorithms aimed at producing both high diversity and fitness in solutions, thoroughly exploring the underlying search space. These are known as *quality/diversity algorithms* [26, 29, 53, 200]. A seen in the previous chapter, the inception of this successful line of enquiry was given by the work of Lehman and Stanley [3, 25], suggesting the exploitation of another interesting phenomena known as *open-endedness*. A system aiming for a variety of diversified and performing solutions that can be achieved through no inherent objective, nor a defined termination criteria. The specific algorithm central to the study of BC in [54], is a successor and essentially first of the QD algorithms: Novelty Search with Local Competition (NSLC) [26]. This is a powerful algorithm, which incorporates multi-objective NSGA-II to exploit trade-offs between high quality and behavioural diversity.

In the study mentioned above, the definitions of BC are given, as a method of categorisation/classification. This is dictated by the notion of quality, based on the problem domain, the BC can either be *aligned* or *unaligned* to this, and no other intermediate categorisations are provided (see Section 2.2.2 in Chapter 2). The authors remark that closely aligned BCs can result in favourable diversity and demonstrate positive exploratory qualities. The intent of this research is to offer our tool to assess the counterargument in the literature [25, 26, 27], which predicates that unaligned BCs are not directly related to the notion of quality that should be capable of offering undirected explorations, leading to divergent ways of discovering a greater variety of optimal capabilities [55]. Therefore, our NTN tool is deployed to illuminate BCs, evaluating these with a focus on *topological complexities*. This is due to the importance that a network's topology and characteristic has on the behaviour produced by a neurocontroller — which is what our research is restricted to. The hope of this work is to see NTNs thrive, as a tool to highlight the mechanics and underlying relationships that exist between topological complexities for BC and explorative diversity. Particular characteristics that can be discerned effectively only through a combination of the NTNs modelling technique and canonical performance assessments.

In order to achieve this, we move away from illuminating the genotypic search space, to replace it with the *space of behaviours* produced by the neurocontrollers.

6.3 THE INCEPTION OF NEUROEVOLUTION TRAJECTORY NETWORKS

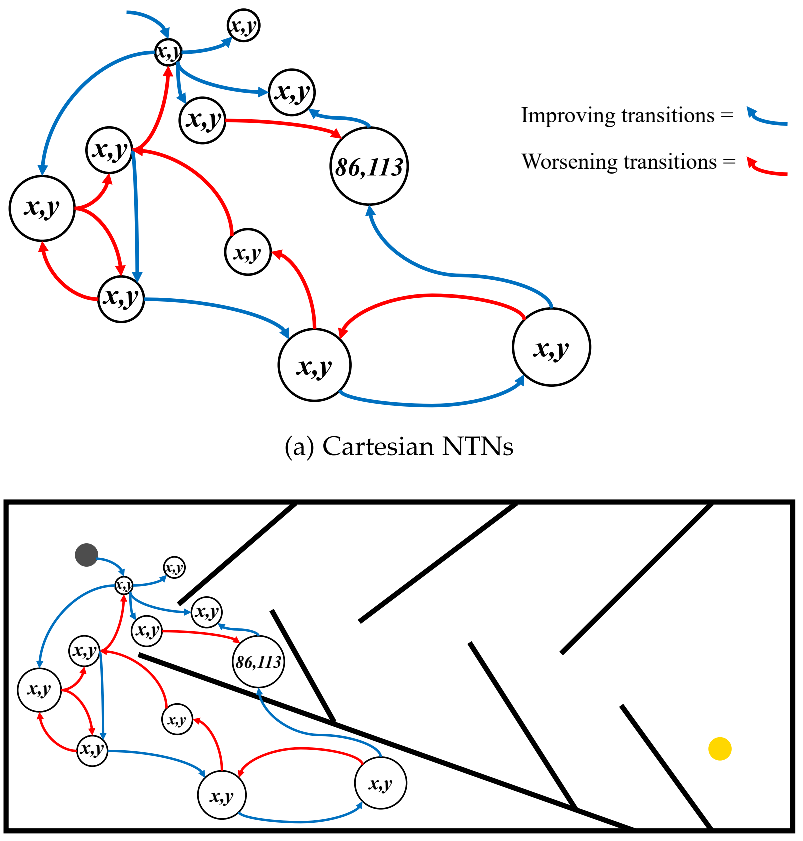
The defining distinction of the NTN tool from its parent technique is the realisation that exploring the dynamics of neuroevolution and its search spaces constitutes a separate postulate In this work we see the clear example of modelling behavioural diversity with a topological complexity focus.

This task was achieved by concentrating on the behaviour space derived from the experiments outlined in Chapter 5. The behaviours are a result of the evolved neurocontrollers from the medium and deceptive (hard) maze. In addition to relying on traits and characteristics that are found primarily in neuroevolution algorithms, part of the intuition here is that behaviours produced by agents, integral components of our divergent search algorithms, directly derive from the capabilities provided by the underlying evolved neurocontrollers. The idea is to use these algorithmic derivations, which reflect neuroevolution dynamics, to move away from the complicated modelling of highly dimensional spaces; with the aim of deriving a more meaningful appreciation of the algorithmic dynamics at play (i.e. convergence of solutions).

This approach offers multiple benefits to our analysis. First, the compressed signature only needs to incorporate the final x and y Cartesian coordinates where the robotic agent lands in the maze upon trial termination, significantly simplifying the signature construction and mapping.

Secondly, having a set of coordinates reduces the information captured in the signature, thereby reducing its dimensionality. Modeling neurocontroller derivations also aids in observing clustering areas, the convergence regions that would otherwise be missed through a genotypical space representation. Convergence in network visualisation is a desired feature, serving as a helpful indicator to comprehend the underlying workings of variants with shared traits.

Finally, since the network nodes are formed by geometric coordinates relative to the maze domains, rather than using the force-directed layouts applied in previous chapters, we designate the layout as the Cartesian plane itself. This allows for the network to be placed directly into the maze, enabling to see the optimisation search dynamics of the best neuroevolved controllers at each iteration. This approach, introduced as a novel component to this research, was inspired by the work on STNs by Ochoa G., Malan K., Blum C. [19]. This will be later demonstrated in Section 6.5. Figure 6.1 illustrates this concept briefly.



(a) Cartesian NTNs

(b) Cartesian NTNs in domain

Figure 6.1: Illustration of the Cartesian layout used in this research.

Figure 6.1a displays an illustrative NTN structure composed of Cartesian coordinates. Each node is generated by a behaviour of the neurocontroller. A larger node can be seen with an example signature, 86, 113 for the location visited. All other nodes contain similar Cartesian values. Since in this experiment, nodes' signatures denote search transitions in the network and correspond to physical locations in the maze, Figure 6.1b illustrates how the same NTN can be represented in the problem domain. This approach is limited to problems which entail forms of geographical coordinates, inherent to the structure of the problem domain — for instance routing tasks.

6.4 ANALYSIS RATIONALE

This experiment utilises settings similar to those in Chapter 5. Additionally, a *random search* (RAND) is included as a form of control. Additionally, based on findings from previous experiments highlighting the inefficiencies of recombination in NEAT [7, 10], the decision was made to exclude this operator and focus solely on variants without crossover.

Random search is the most basic amongst the strategies. It assigns continuous stochastic values derived from a pseudo-random number generator as the evaluation score of the genomes. As a control, this strategy is used to explore the effect of divergent search, and it is not expected to yield good performance. All other strategies employ the same criteria and parameters outlined in Chapter 5. This analysis will use Neuroevolution Trajectory Networks for the behaviour space, with specific focus on divergent search and topological complexities of the neurocontrollers generating behaviours. The performance analysis covers statistical tests, in terms of evaluations required to reach a solution (efficiency) — which tracks the success rate of each strategy — and quality of solutions, once the evaluation cycle ends (efficacy). Furthermore, this work offers a behavioural diversity analysis, placing particular attention on their relation to topological structures and complexity.

To accomplish this, 30 independent runs are executed. Out of these, 10 are selected by ranking them based on performance. The top 3, worst 3, and 4 intermediate runs. The intention was to choose a representative sample of the 30 runs, as visualising all runs would make the plots overly complex, hindering the extraction of visually meaningful features. As discussed in previous chapters, this empirically dictated runs filtering, is strategic to strike the balance between meaningful, informative networks which do not result in complicated visualisations. Excessive data points in a complex network visualisation as such do not always result in extra useful information about the underlying dynamic. This is a supporting motivation for the CSTNs technique detailed in the previous chapter.

In this work we use the force-directed layout algorithm proposed by Kamada and Kawai [201], using *single* (non-merged) Compressed NTNs. These leverage the same definitions outlined in the previous chapter. This layout is a variation, inspired by the same physics principles of [193]. Specifically, in their approach, a virtual dynamic system is formed, in which two vertices are connected by a "spring" of such desirable length. The optimal layout of vertices is achieved when the system's total spring energy is *minimal*. In NTNs, layouts such as this play an important role in the communication of the internal mechanics of an algorithm. The choice for a layout is often empirically driven and lead by intuition and prior knowledge of the neuroevolution algorithm in analysis. In this experiment the specific variant of the layout chosen helped us to convey important characteristics of topological complexity and diversity, witnessed in the supplementary analyses and distribution plots.

Additionally, two new metrics specific to this analysis are computed on top of the canonical ones already used. A *complexity* metric, which is the average topological complexity of the NTNs, in relation to divergent and convergent searches respectively. Moreover, *in-strength*, the average strength of incoming connections, indicating to what extent some nodes in the network attract the search process (calculated as a normalised average). These two metrics corroborate our behavioural diversity and topological complexity analysis, and together with the statistical approach, offer a comprehensive view of the neuroevolving system.

6.5 RESULTS AND DISCUSSION

In this section we will present the results achieved in this study, starting from the behavioural diversity analysis, progressing to the NTNs visualisations and the associated metrics.

6.5.1 *Behavioural diversity related to topological complexity*

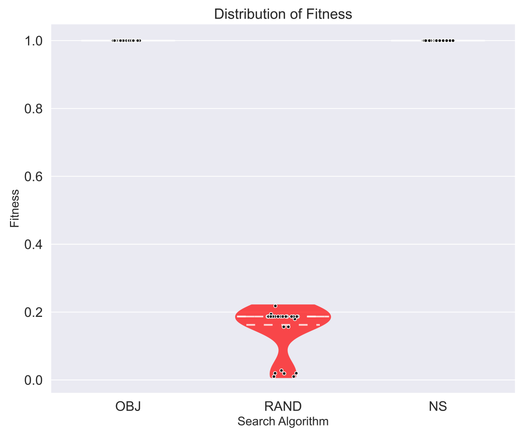
In the analysis of this section the objective is to begin by testing the known differences between divergent search strategies such as *novelty* and *random*, compared to the convergent strategy of *objective* search, in relation to the topological complexities of the neurocontrollers evolved to solve the maze. These preliminary assessment served as tests, and are the guiding force and confirmation signals for the advanced NTNs analysis. These assessments are particularly important for the BC study mentioned in earlier sections, but also because for these studies we are deploying variants of NEAT, which leverages a complexification mechanism in its evolutive process. Complexification, as described in Chapter 2.2.1 is a property of NEAT, a mechanism which is intended to generate simple neural network composition and incrementally increasing their complexity without exceeding or unreassuringly bloating the solutions.

In the results, random search, despite being considered a divergent strategy, it does not produce any successful results. The only variant succeeding in the hard (deceptive) maze, as witnessed in the previous chapter is Novelty Search. Nevertheless, this examination does not focus on what succeeds or works, but why something does not succeed and what are the underlying effects generated. Highlighting what makes a strategy different, both statistically and using NTNs.

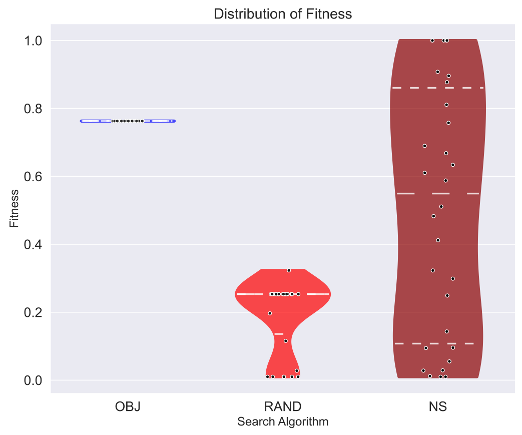
In [54], the authors remark on the importance of identifying the correct Behavioural Characterisation for a deceptive domain. It is fundamental for a successful algorithm that explores solutions with the aim of diversifying. As the name suggests, this approach is primarily intended for actions produced by neurocontrollers. Novelty Search is aimed at diversification and exploration of behaviours, but it can also be leveraged to identify topologies with novel performing complexities. These results bring to light some interesting insights into this potential search setup.

These results are studied from two particular standpoints, which we can easily derive from the systems during evolution. We capture the *topological complexities* information in terms of neurons and connection numbers. As our study involves evolutionary dynamics, the focus is only on those neurons and connections that are evolved and not preconfigured. Our metric is calculated by summing these two components and then subtracting the number of neurons that are preset by the configuration file. A fair comparative analysis involves a normalisation of this complexity metric to be in the range $(0 - 1]$. The choice for this specific approach was purely empirical, as the literature did not suggest any alternative to this neurons and connections aggregation metric. The behavioural diversity component is calculated based on the ending x and y coordinates of the neurocontroller at trial termination, for simplicity, as specified by our implementation [9]. For this statistical outlook and the NTNs analysis, these values are reduced in precision by rounding off decimal points and concatenated into the nodes signatures. This way we discretise the space of possible behaviours, a custom practice in STNs/NTNs techniques. To produce this analysis these are treated as identifiers and only distinct behaviours are picked.

From the statistical results related to the distributions of fitness values and evaluations required to reach the threshold, in Figure 6.2 we observe that in the medium maze (Figure 6.2a) both objective (OBJ) and Novelty Search (NS) perform comparatively well, with all data points (runs) reaching the top value of 1.0. Conversely, as expected the divergent random search method (RAND) had most run reaching values around the 0.2 mark, proving that this strategy, although divergent, is insufficient even in the simpler medium maze.



(a) Fitness medium maze



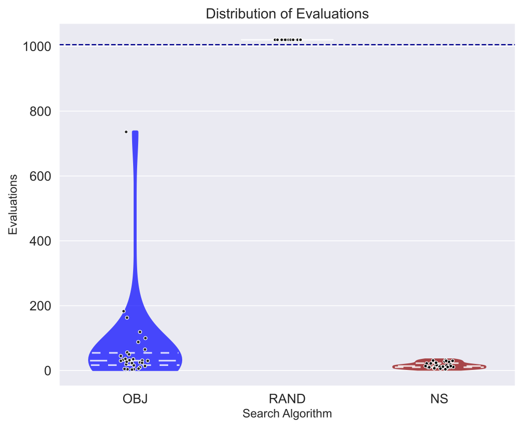
(b) Fitness hard maze

Figure 6.2: Fitness distribution of all strategies for each domain difficulty

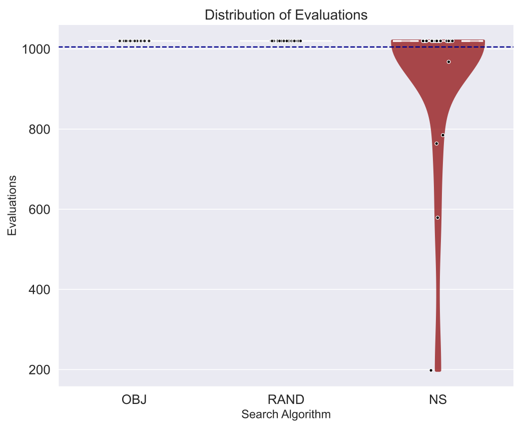
In Figure 6.2b, we observe that neither OBJ nor RAND are able to achieve successful fitness values. On the other hand, the NS divergent search strategy has 5 runs reaching/exceeding the threshold and succeeding in the hard maze set up. It is noticeable how, similarly to RAND, this strategy does not produce stagnated fitness values, as seen in the flat distribution line of OBJ search.

Looking at the distribution of evaluations in Figure 6.3, it is observable that, although both OBJ and NS are successful, the divergent NS strategy achieves the optimal fitness threshold, in less iterations — around 100 on average — producing a narrower spread. Contrary to OBJ, which has values of evaluations that exceed 100 iterations, confirming its negative performance in this domain compared to the divergent strategies. Failing runs are depicted above the dotted line, signifying that the runs exceed the termination criteria of total iterations.

Figure 6.4a and 6.4c illustrate the distributions of normalised topological complexities for neurocontrollers in the 30 runs executed. These values are computed as an average of all the best topologies found from all the iterations, in each of the runs. This proposes a planarisation, flattening the



(a) Evaluations medium maze



(b) Evaluations hard maze domain

Figure 6.3: Evaluations distribution of alls strategies for each domain difficulty.

time dimension, to capture all topological complexities seen during the runs. This practice, is effective as it mimics the same approach used in the NTNs methodology. Figure 6.4b and 6.4d show the number of distinct behaviours recorded for all search strategies in each of the 30 run for both maze setups. These distributions help us to perceive the diversity of solutions found by each strategy. We believe these give a sense of how divergent a variant can be. The findings, identified through this analysis, are reported as follows and will aid our understanding further in the NTNs analysis section.

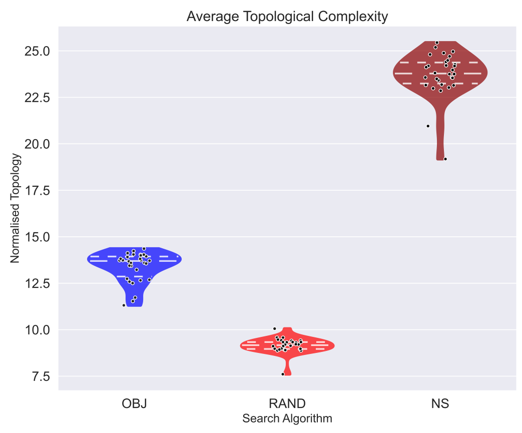
The NS strategy has on average more complex topological structures of the neurocontrollers than RAND and OBJ. The spread in RAND is rather narrow, and topologies appear notably less complex, in both maze configurations.

Looking at RAND, high and consistent diversity values (narrow spread) are visible. This points to the hypothesis that simpler topologies may be capable of creating more diverse behaviours.

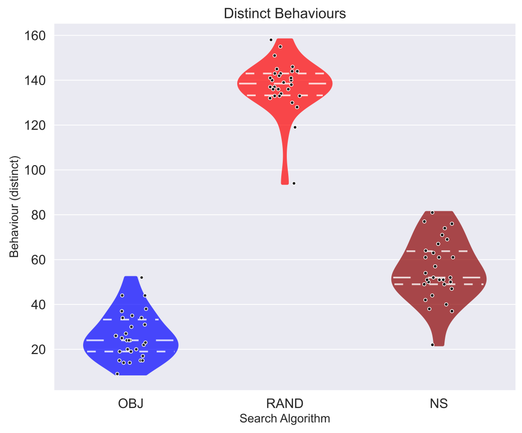
Due to its observed poor performance compared to other strategies, RAND is deemed ineffective. Although, in terms of diversity, we argue that this strategy produces more consistent results. These consistent results of diversity are achieved by topological structure that are on average less complex.

OBJ appears to show similar dynamics, with notably complex topologies on average that do not produce highly distinctive behaviours compared to the other variants. Corroborating the point expressed above, about complexity not allowing an extensive behaviour space exploration.

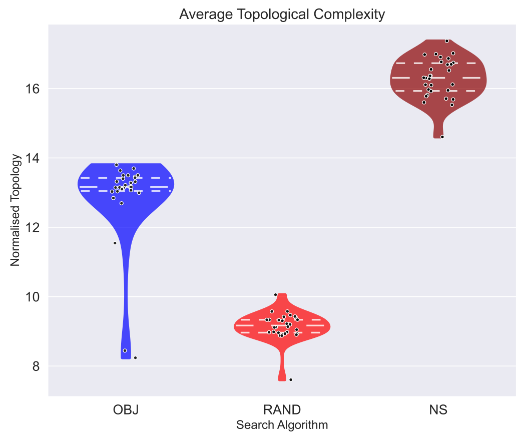
The variance in distribution of distinct behaviours is high for NS, signalling an instability of this strategy, between low and high diversity generation. Indicating a less consistent exploration. A result which was not suggested by [25].



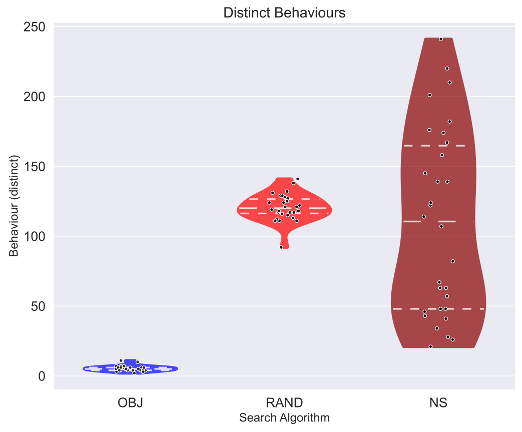
(a) Average topological complexities medium maze



(b) Distinct behaviours medium maze



(c) Average topological complexities hard maze maze



(d) Distinct behaviours hard maze domain

Figure 6.4: Topology complexity and behavioural diversity analysis of distributions for all variants in both maze configurations.

6.5.2 *Visualising the behaviour search space with NTNs*

Here we presents the results achieved through the NTNs analysis. As mentioned in Section 6.4 this examination leverages *single* CNTNs [10]. Ten runs out of the 30 are selected and portrayed based on their success — based on fitness performance; these individual plots are categorised by search strategy and domain difficulty. Furthermore, as conducted in previous studies, specific network metrics are computed to corroborate the findings. Finally, a further aim of this study is to witness the applicability of NTNs when transferred into the problem domain, with a layout dictated by their x, y coordinates.

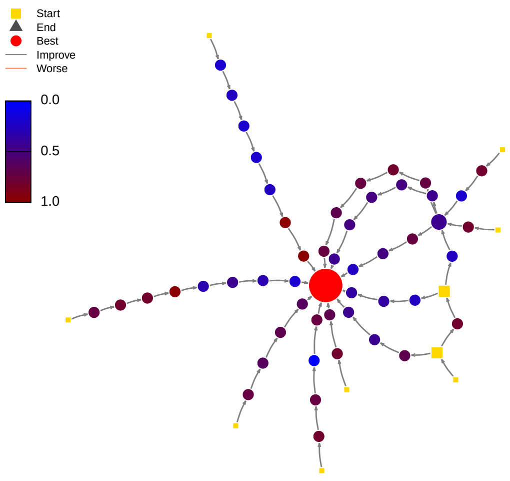
Figure 6.5 illustrates the NTNs generated, for each search strategies, on each of the domains set up. These are comprised of 10 selected runs, the starting points of these runs are marked by yellow squares, a grey triangle signifies the end of a behaviours trajectory and the best solution in terms of fitness found are depicted as red node. The colour gradient is used to signify the normalised topological complexity of the neurocontrollers producing behaviours. Dark red meaning highly complex and light blue being extremely simple.

Table 6.1: NTNs metrics.

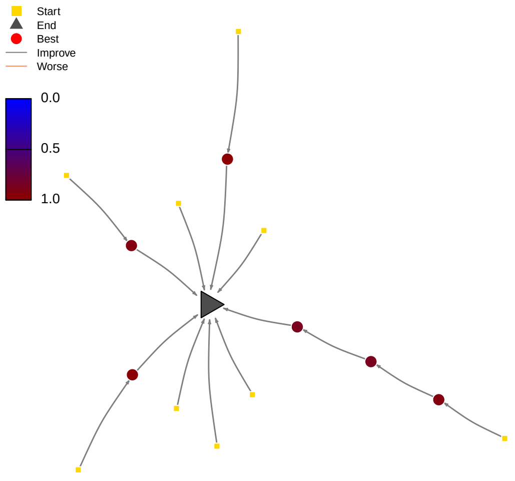
	Medium Maze			Hard Maze		
	OBJ	RAND	NS	OBJ	RAND	NS
nodes	66	653	67	25	536	701
solutions	1	0	1	0	0	1
w-edges	0	0.460	0.293	0	0.421	0.456
n-path	10	0	10	0	0	9
p-length	5.7	N/A	5.9	N/A	N/A	8.555
complexity	0.360	0.155	0.177	0.339	0.135	0.226
in-strength	0.015	0.015	0.016	0.070	0.022	0.004

The OBJ networks clearly highlight the greedy, convergent behaviour of this search strategy — also represented by the worsening edges (*w-edges*) metrics of Table 6.1, which reports a nil value. The global solution of the medium maze (red node) is easily achieved by this strategy, with some convergence to common behaviours observable by the larger nodes with higher *in-strength*. In the hard maze configuration, the behaviour search process of OBJ quickly gets attracted and trapped in a shared sub-optimal behaviour. This illustrates that in very few search steps the strategy gets trapped in a location, finding only one underperforming behaviour. In the medium maze, a variety of simple to complex topological structures reach the optimum. For the harder domain, all neurocontroller which are trapped in the attraction regions (possible local optimum) present high topological complexities (darker red).

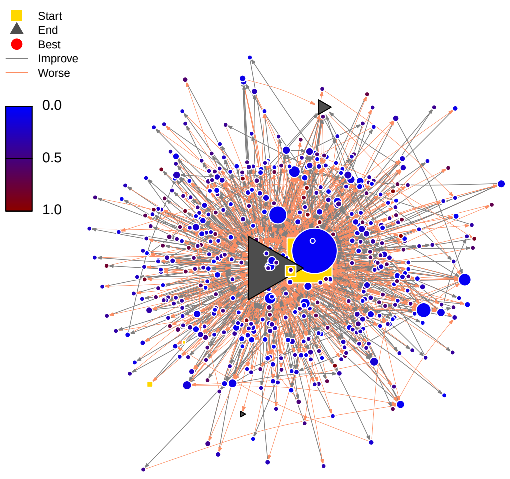
In RAND, high levels of diversity are maintained for both domain configurations, with some commonly attracting behaviours. In both mazes, this strategy appears to have large attracting behaviours, primarily occurring at the end of trajectories. The large metric values of *w-edges* and *in-strength* observable in Table 6.1 are visible in the NTNs of this strategy, manifesting



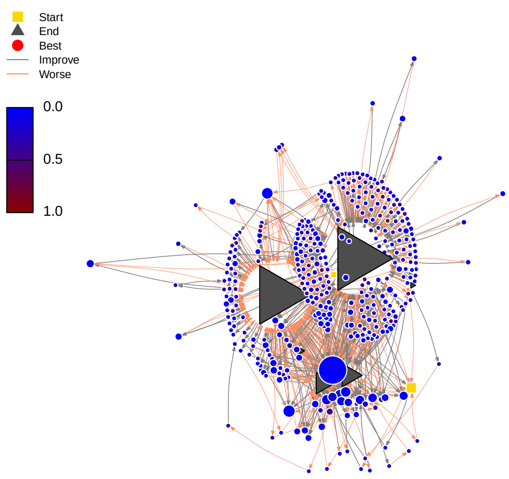
(a) OBJ, medium maze



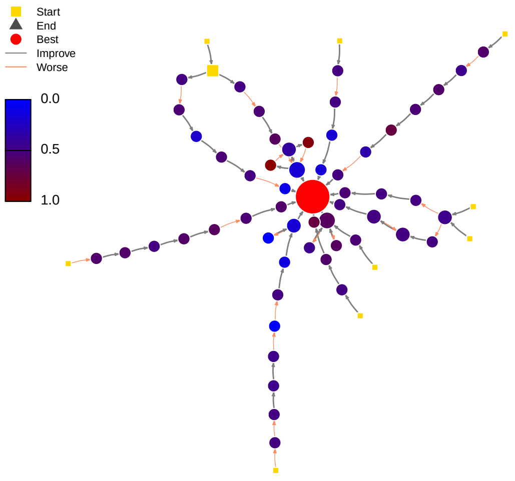
(b) OBJ, hard maze



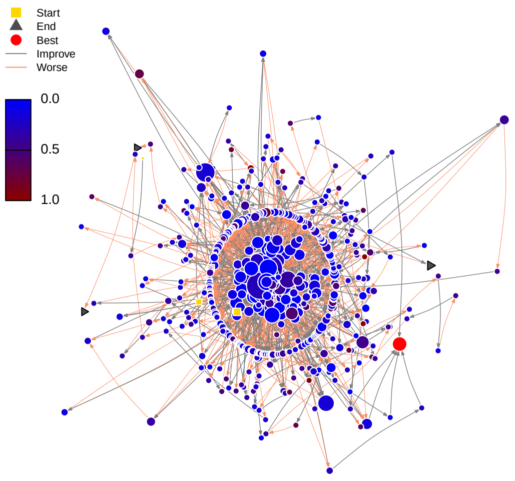
(c) RAND, medium maze



(d) RAND, hard maze



(e) NS, medium maze



(f) NS, hard maze

Figure 6.5: single CNTNs for 10 selected runs, in terms of performance for all strategies in both domain configurations [11]. The colour of the nodes are based on a gradient signifying normalised topological complexities, and the size is related to the incoming strength of the connections.

in an erratic and oscillating dynamic, between improving and worsening behaviours (see edges). This encapsulates the random essence of this variant, which is a diversification characteristic of this divergent search type. In these NTNs, most nodes hue attributes (neurocontrollers' complexities) appear mid to low (blue/light-blue).

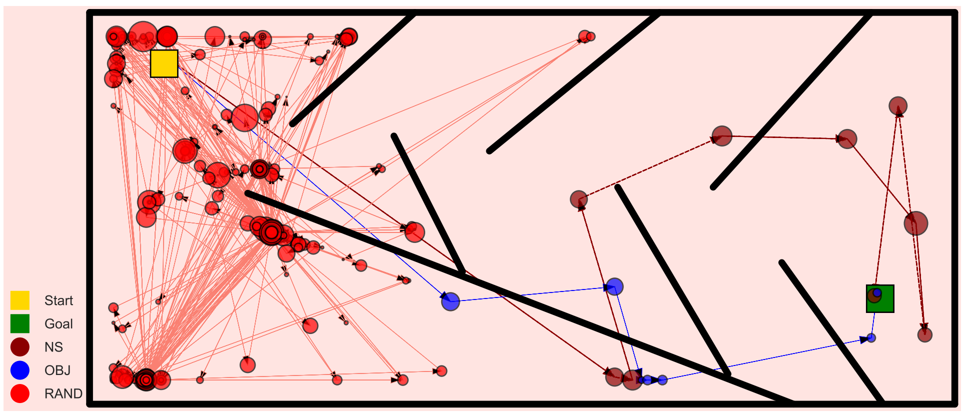
The NS network for the medium domain obtain similar node convergence levels, meaning that the same behaviours are re-visited several times, in close proximity. This dynamic produces a higher *w-edges* value. The average length (*p-length*), related to the solutions paths, are similar to the objective-driven strategy, but NS, being a divergent search, can produced increased exploration. This points to the observation that NS can have similar levels of topological complexities as random search.

From the NTNs depicted in Figure 6.5f it is possible to observe an interesting feature that provides insight and support to our topological complexities hypotheses. In the centre of the network visualisation, it is possible to discern a distinctive circle comprised of nodes that have similar size (convergence), we assume from this composition that these nodes inside this circle (formed artefactually by the layout algorithm) present truths about the underlying system related to lesser behavioural diversity (exploration). This phenomenon is explained by the high convergence values, seen and reported in the metric, while having topologies which appear more topologically complex — nodes with darker shades. The nodes outside this circle, are visibly simpler, more diverse, capable of greater exploration, reaching the solving behaviour (red node). These are strong intuitions and It is in the intent of future research to test these further.

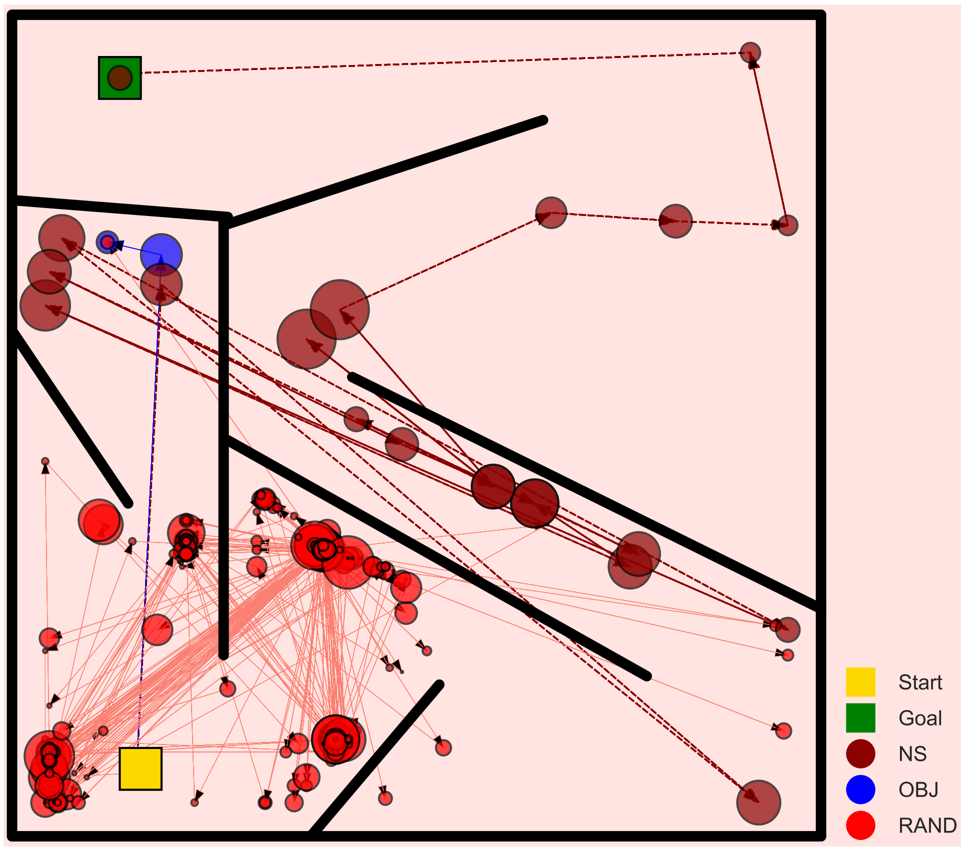
In our examination used NTNs to offer an explanation to this network formation. It may signify a clear breakthrough from higher complexity topologies, showing lower diversity nodes trapped by local minima regions, to simpler neurocontrollers that lead to higher diversity and more successful explorations.

Finally, our analysis concludes by considering a representative example of the best runs based on fitness, identified out of the 30 executed for each of the search strategies, and placing their respective NTN models inside the domains. This is achieved using a simple layout dictated solely by binary coordinates on the Cartesian plane.

Figure 6.6 illustrates this approach. The trajectories colours indicate the strategy type and all other decorators are similar to previous plots, for the exception of node size, which in this case is associated to the normalised topological complexity metric. Meaning that larger node signify more complex networks and vice versa.



(a) Medium Maze



(b) Hard Maze

Figure 6.6: Cartesian NTNs for the best runs in fitness performance terms, for all strategies in both mazes configuration [11].

In these plots, by looking at the behaviours of random search and their topologies, we can visibly corroborate our previous intuition that simpler neurocontrollers offer greater explorative capabilities, due to their further divergence from the starting point of the maze.

The breakthrough of Novelty Search described earlier, observable in the NTN model of Figure 6.5f, resembles the changes in topological complexities occurring in this plot, seen in the maze's diagonal upward channel. When topologies diverge from this deceptive region, they appear to become topologically less complex. As if neuroevolution offers a simplification of the neurocontrollers in order to achieve greater exploration.

We further support this intuition by suggesting that simpler networks possess a stronger flow of signals between input and output nodes compared to larger networks.

6.6 SUMMARY

This contribution chapter is where we first describe the inception of Neuroevolution Trajectory Networks. A visualisation method that solely focuses on illuminating neuroevolution search spaces and the intrinsic dynamics of algorithms and variants operating in this space. We propose differentiating from the original STNs technique, to discriminate the study of neuroevolution from a standpoint that is not anchored on the use of fitness performance alone.

The true novelty in this differentiated approach also lies in the intent to move from signatures which encapsulate solution vectors, to specialised neuroevolution ones that define network constructs and architecture, in addition to behavioural characterisations and descriptors of novelty. To further strengthen the NTN case, it is dutiful to declare that the use of the Cartesian layout without the canonical complex network compositions (primarily deployed through force directed and tree layouts) would not support the overall analysis; losing the novelty and uniqueness of this proposed approach.

In this chapter we outline how NTNs were leveraged to study Behavioural Characterisation (BC) a thematic proposed in [54], concerning divergent search strategies and exploration capabilities within deceptive domains. This led our NTNs examination of search strategies dynamics to be corroborated by a statistical analysis of topological complexities related to behavioural diversity and exploration of these domains.

In the experiment outlined here, our proposed visual assessment of behavioural characterisation is achieved on three search strategies deployed in NEAT. Two divergent ones, novelty (NS) and random (RAND), and a convergent one, objective-driven search(OBJ). These were tested in medium to highly deceptive maze domains. Our findings highlighted some important characteristics and relationship on the interplay between diversity, exploration and topological complexity of the evolved neurocontrollers.

Amongst these observations, we discovered from both violin plots and NTNs visualisation that topologically less complex neurocontrollers have the tendency to create increasingly distinct and diverse behaviours, with potentially high exploration capabilities. Capabilities which in the majority of cases, eventually lead to successful solutions (the domain goal). The phenomena were also visible through the visualisation of the best NTNs for NS through a Cartesian-driven layout, placed in the domain.

On the notion of BC, detailed in [54], we recall from Chapter 2 that this is the way by which behaviours are classified. The classification are found to be either *aligned* or *unaligned* to the notion of quality (fitness) that is determined by the assessment domain. Our work suggests that alternatives between a closely aligned and non-aligned BC potentially exist. These may be described as *Transitive BC*, indicating that divergent search strategies could focus on BCs indirectly related to divergence and exploration, to achieve appropriate quality-diversity (QD), which follows the open-endedness notion of completely abandoning the objective [25, 195, 202, 203]. In the case of this experiment, it could be framed on restricting the evolutions to low topological complexities. As seen in these specific experimental setup, these offer neurocontrollers that generate diverse, explorative behaviours. It is in our future research intent to test these indirect search mechanism, the identification of these could enhance the performance of divergent strategies. A further suggested direction would be to compare and analyse these against automatically defined novelty descriptors [56].

CONTRIBUTIONS ACKNOWLEDGEMENTS I hereby declare that the substantial portion of the work detailed in this chapter, as documented in the publication [11], is the result of my own efforts. However, it is important to acknowledge the valuable contributions made by my co-author and supervisor, Professor Gabriela Ochoa. Professor Ochoa provided insights and guidance on the overall research approach and helped the structure of the research narrative. They have additionally contributed in refining and perfecting the metrics of the NTNs visualisations. My co-author and second supervisor Dr. Jason Adair has also contributed to this research through general proofreading and sense checking of the overall research narrative.

7

CHAPTER 7 — REVEALING THE PAST OF INCREMENTALLY NEUROEVOLVED CONVOLUTIONAL NEURAL NETWORKS

This chapter outlines the research work published in [22]; this contribution is reported in the publication list of Section 1, at the start of this thesis.

This research advances the study from *shallow* networks to *deep* ones, structured based on layers. The fundamental aim of this work is to enhance NTNs by applying them to deep neuroevolution. Furthermore, we analyse a class of algorithms which incorporates *incremental development*, which leads to effective transfer learning properties. The class of algorithm that will be of focus in this chapter are powered by a type of Grammatical Evolution (GE), known as Dynamic Structured Grammatical Evolution (DSGE) [62]. The specific algorithm used for this experiment is Fast-DENSER with incremental development; both of these have been detailed extensively in Chapter 2, Section 2.2.3 and 2.2.3.

7.1 KEY CONTRIBUTIONS AND MOTIVATIONS

In this fourth and final contribution chapter of this thesis, we detail a fundamental research that solidified Neuroevolution Trajectory Networks as a complex networks visualisation methodology. This work helped to highlight the applicability of NTNs/STNs to various domains. In particular, we bootstrapped this technique from its inception domain of shallow/dense ANNs, to apply it in the world of DNNs. Furthermore, aside from corroborating to supporting our argument that this technique has strong basis for explainability in neuroevolution, the intention was to strengthen this further by showing that it can indeed be used to highlight fundamental dynamics of transfer learning related to image classification.
The principal contribution that this work has provided are as follows.

- Provide an assessment of incremental development/transfer learning for Fast-DENSER variants that include recombination and local search.
- Provide an in-depth NTNs analysis of characteristics intrinsic to these variants, when used for evolving CNNs tested on benchmark image classification problems.

- An augmentation of current NTNs technique, to signal knowledge transferred during the incremental development of CNNs; in order to globally assess the fundamental components selected by neuroevolution.

7.2 FROM SHALLOW TO LAYER-BASED NEUROEVOLUTION

The progression of NTNs from the study of dense neural network which are *shallow*, to the study of deep learning and DNNs has strengthened our claim for the successfulness of NTNs technique in this realm. Aside from further corroborating the applicability and usefulness of this approach, the shift to study Deep Neural Network — which to this date is more actual and relevant — has highlighted some interesting benefits. As shown in Figure 7.1, we illustrate the dichotomy of modelling neuroevolution search spaces, between shallow and deep neural networks.

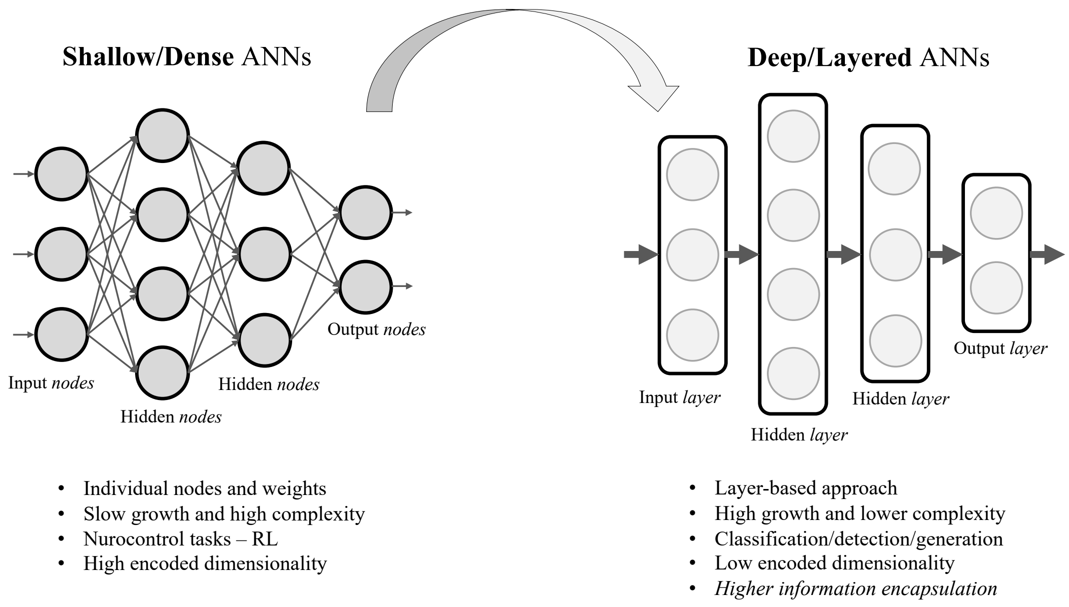
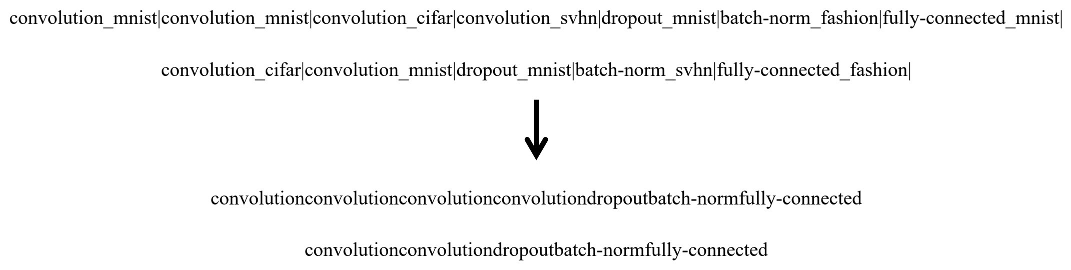

Shallow/Dense ANNs
Input *nodes*
Hidden *nodes*
Hidden *nodes*
Output *nodes*
• Individual nodes and weights
• Slow growth and high complexity
• Nurocontrol tasks – RL
• High encoded dimensionality
Deep/Layered ANNs
Input *layer*
Hidden *layer*
Hidden *layer*
Output *layer*
• Layer-based approach
• High growth and lower complexity
• Classification/detection/generation
• Low encoded dimensionality
• *Higher information encapsulation*

Figure 7.1: Illustration of the progression in NTNs analysis from *shallow* to *deep* neural networks.

Dense ANNs evolution, used in our experiments until now (Chapters 4, 5, 6) are characterised by a genotypic encoding of individual nodes and weights, this carries with it a very slow evolution/ optimisation, which includes diverse forms of encoding characterised by high complexities. On the other hand, evolving DNNs, in this experiment, but also in the general context could be defined more easily as more modular and dictated by a layer-based approach. The evolutionary growth in this case is quicker, and less complex from an encoding perspective, due to these building blocks being segmented in larger definable chunks. It is known that evolving dense ANNs, as in the case of our experiments thus far, has often been associated to the reinforcement learning RL category of problems. As opposed to the more recent DNNs which, aside from being applicable to RL, it is used extensively for image classification, object detection and as generators [63, 204, 205]. Furthermore, one aspect specifically worth of consideration, between the two architecture structure is that from an NTNs prospective, the encoding and signature mapping can become far less complex in the deep networks, due to this modular nature. It makes the dimensionality to be encoded in our NTNs/STNs signature, lower than in the shallow approach. This sounds counter-intuitive, but a layered based neuroevolution can reduce dimensionality and narrow the search space scope. This is due to the fact that we are not evolving individual nodes and connections, but a defined set of allowed evolvable units and hyper-parameters. Finally, this neural network category, improves the information encapsulation, meaning that the genetic material is grouped in these layer-chunks, which can be tagged and labelled, maximising the information inclusion within a NTN signature.

7.3 TRACKING THE TRANSFER OF EVOLUTIONARY UNITS


convolution_mnist|convolution_mnist|convolution_cifar|convolution_svhn|dropout_mnist|batch-norm_fashion|fully-connected_mnist|

convolution_cifar|convolution_mnist|dropout_mnist|batch-norm_svhn|fully-connected_fashion|

$\downarrow$

convolutionconvolutionconvolutionconvolutiondropoutbatch-normfully-connected

convolutionconvolutiondropoutbatch-normfully-connected

Figure 7.2: Illustration of two example NTN signatures with and without origins tags.

In this research, the intention was to place a particular focus on the transfer of knowledge occurring due to the incremental development nature of Fast-DENSER [21]. This investigation was inspired by preliminary work conducted by the authors, which engineered the incremental development neuroevolution algorithm [21]. In their publication [5], the authors propose a visualisation of Fast-DENSER with incremental development. Their intent was to visualise the transfer of genetic material occurring from generation to generation, on a single run selected (shown in Figure 7.3). This was considered an initial attempt at showing that the transfer of knowledge does work in this neuroevolution algorithm and genetic material encapsulated in layers can be ported based on the performance of best individuals. Moreover, incremental development is seen to improve convergence on optimal solutions faster than without the transfer of learning from previous domains; a process which is detailed in the Background Knowledge at the end of Chapter 2.

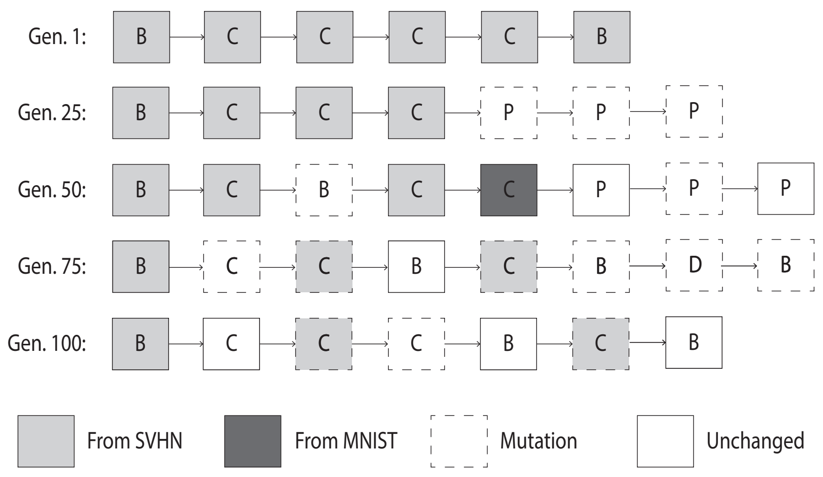


Figure 7.3: Illustration of the first attempt of revealing incremental development on one run of the original F-DENSER. Diagram sourced from [5].

Our process of signature mapping is based on *origins* tagging. An example of two signatures for the NTN models created in this analysis is shown in Figure 7.2. The process is simple; as CNNs evolve, they acquire evolutionary units. In this analysis we use this genetic material encapsulation, used for the deep networks encoding, to form the signatures. In this case, as shown in the illustration, as this is a layer-based neuroevolution, the focus is placed particularly on the layer which compose the networks. This research is paramount for the establishment of the overarching NAA methodology, which will be discussed in the conclusion Chapter 8. Furthermore, our examination places attention on the learning optimisers that are ported from previous learnings; this can be *Adam*, *Gradient Descent* or *Rmsprop*, as per the predefined CFG.

As the evolutionary system progresses in the search for new solutions through incremental development, these units are inherited. Our aim is to shed light on this inheritance and the previous stages of this process. Therefore, while this process occurs the *origins* tags — that is, from which domain these were inherited via neuroevolution — are allocated to the signature.

For instance, if we look at the second signature from the top of Figure 7.2, we observe a sequence of two distinct units starting the DNN. Two *convolu-*

tional layers, a fundamental component of CNNs. The first originated from the CIFAR-10 and second form MNIST classification datasets. Meaning that these encapsulated evolutionary units were inherited and introduced in the solution genotype, through evolution and learning of those specific classification datasets. Following the arrow, we observe the unique signatures of the nodes, stripped of these origin tags. This is to ensure that the tags are used for the decoration of nodes and that the architectures of said CNNs are the only one generating the NTNs nodes and edges interactions. These origin decorators are then used to fill nodes with a hue, in the visualisation models.

7.4 ANALYSIS RATIONALE: NTNS CHARACTERISATION

Let us delve into the details of how the models were generated and how these visualisation should be interpreted. Due to the high dimensionality of this analysis, the exemplification that is provided in Figure 7.4 taken from [6], will serve as a didactic medium to convey the concept.

These network models, constructed from a NTNs data log of node, edges and attributes/decorators, is derived by the study of our proposed variants, running these on the last benchmark classification dataset — Fashion-MNIST. The NTNs are generated based on the 10 executed runs for each variant. As outlined at the end of Chapter 2, the process works such that solutions are developed on all classification datasets, starting from MNIST, progressing to SVHN, then CIFAR-10, and finally, Fashion-MNIST. The analysis specifically focuses on this last dataset, retrospectively looking into what occurred through past evolutions. Therefore, our generated NTNs are based on 10 runs from the last classification dataset. This will allow us to detect what evolution has favoured to carry over as fundamental components of the CNNs architecture, to optimise for Fashion-MNIST classification.

The unique signatures generated represent evolved CNN structures, reflecting the permissible architecture of the networks discovered — from feature extraction to classification. The usable structures for CNNs is well known, and in this DSGE-powered approach, the allowed search space is dictated by the CFG.

Each representative solution is associated to its fundamental attributes. Aside from the layers *origins* we also track the learning units, in the same way. Depth, thought of as the size of CNNs, is also recorded, in terms of *trainable parameters*. With this log file generated, solutions are mapped to locations, based on their origin, through a post-processing stage, which models the network object using the canonical NTN/STN procedure.

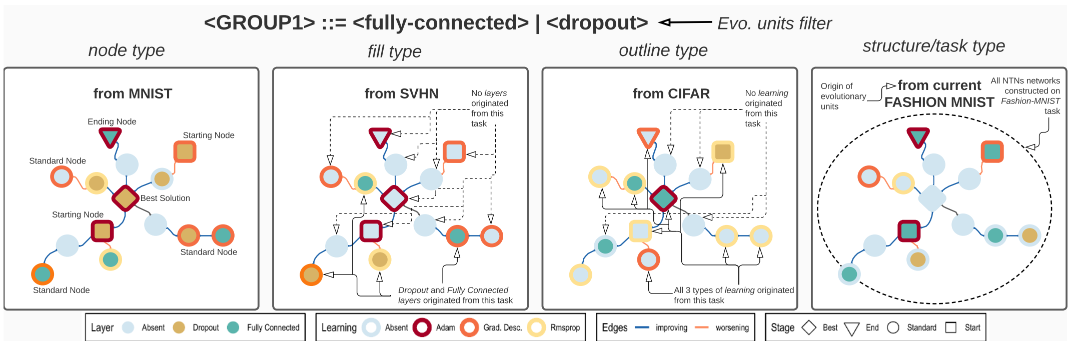

<GROUP1> ::= <fully-connected> | <dropout>
Evo. units filter
node type
fill type
outline type
structure/task type
from MNIST
Ending Node
Starting Node
Standard Node
Best Solution
Starting Node
Standard Node
Standard Node
from SVHN
No *layers* originated from this task
Dropout and *Fully Connected layers* originated from this task
from CIFAR
No *learning* originated from this task
All 3 types of *learning* originated from this task
Origin of evolutionary units
from current FASHION MNIST
All NTNs networks constructed on *Fashion-MNIST* task
Layer
Absent
Dropout
Fully Connected
Learning
Absent
Adam
Grad. Desc.
Rmsprop
Edges
improving
worsening
Stage
Best
End
Standard
Start

Figure 7.4: Exemplification of the NTNs visualisation for this complex multidimensional space [6].

As stated, owing to the numerous components in this analysis, the decision was made to compartmentalise the network visualisation into layer-groups related to the CNNs specified in the grammar: features and classification. Six layers types are defined in the grammar; they are the one that constitute the standard CNN architecture type. Each category (see *Evo. units filter* in Figure 7.4) highlights two layers at a time, producing three plots in total for each of the variants. This approach helps to removes other architectural components, leaving only the DNNs layers of interest for that visualisation. The three filter group are the following.

- group 1 for *fully-connected* and *dropout* layers (used in our illustration)
- group 2 for *pooling average* and *pooling max* layers
- group 3 for *batch normalisation* and *convolution* layers

These groups have their associated colours as decorators, representing them. Our three aforementioned learning optimisers (Adam, Rmsprop, Gradient Descent) are also represented and carry the same colour throughout the groups; these colours are used for the nodes outline.

The structure and layout of NTNs remain consistent across compartments and dataset groups as the visualisations relate to the network constructed using the representative solutions (architectures comprised by layer types) for the final dataset, Fashion-MNIST. This was done to further enhance the interpretability, reproducing the same visualisation, allowing to easily compare the knowledge transferred between datasets. This was achieved by fixing each seed of the force-directed layout [193].

The NTNs structures are then subdivided by classification problems. If the category layers were not ported from one of these datasets, a pale grey colour decorates the nodes (CNN solutions) to indicate their absence, shown in the illustration above. Therefore, in Figure 7.4, each of the four dataset visualisation describes the contributions of that dataset classification problem to the final network.

The shape nodes, as in previous work, help to represent the different stages that are possible in a trajectory. The start of a trajectory which coincides with the start of a run is depicted as a square. Standard nodes are circular. The end of a trajectory (end of a run) is depicted as a triangle. While the best solution found (best CNN architecture in terms of accuracy) is represented by a diamond.

Similarly to the NTNs analysis proposed in Chapter 6, the size of these shapes is proportional to the normalised metric of trainable parameters, recorded for each CNN in the logs. This attribute is seen as a proxy of the DNN depth, the more trainable parameters, the more layers and deeper is the architecture. In Figure 7.4, the label "from SVHN", helps us illustrate how the node hue fill works. As discussed, this is dependent on whether the specific layers of the filter group, originated from that dataset in which the NTN is. The decorators for nodes' outlines are illustrated in Figure 7.4 "from CIFAR", consistent throughout groups, as the available optimisers are the same in all categorisations.

7.5 EXPERIMENTAL SETTINGS

This neuroevolution approach uses a ES algorithm as the evolutionary engine for Fast-DENSER, coupled with DSGE core [5]. Two variants of these algorithm were compared, aside from the original version. One with mutation and a single-point crossover implemented for the ES. Another variant is a simple *local search* where the starting solution is the best of five initial solutions derived from the population inception. Our rationale behind experimenting with these specific settings is that the first variant tests our hypothesis. Similarly to what was observed in previous neuroevolution analyses [7, 10], using crossover as a means of optimisation does not produce any significant improvements to the algorithm as it originally is. The Local Search variant, on the other hand, is inspired by the work that led to the development of Fast-DENSER [21]. The objective of the authors was to improve DENSER, to search for optimal solutions with limited resources, in a much faster time frame, due to the highly reduced population (see end of Chapter 2.2.3 for details). This led us to the creation of local search for Fast-DENSER, further reducing testing and training times, while preventing degradation of performance; an intuition offered by the notion that local search being is a simple and powerful algorithm [49, 206].

The variants are analysed primarily through our NTNs visualisation technique, although, as done in previous studies, we also compare the performance of incrementally developed solutions, statistically from the Fashion-MNIST perspective. This is because the last dataset should incorporate knowledge gained from the previous three datasets. This preliminary analysis is performed on 10 individual runs, the same ones used for the NTN analysis.

In the preliminary analysis, the goal is to compare the three algorithmic variants purely from a performance perspective on the basis of the defined objectives derived through the characteristics of each variant. This assessment is based on three aspects in particular. Primarily, we look into the training and testing accuracies of classifications. Secondly, the size of evolved DNNs, using the trainable parameters metric. This is to assess the depth of CNNs each variant is able to evolve. Mainly because more parameters to tune is likely to correspond in longer training; to achieve this, each variant's average time to train is assessed. The primary objective is to explore whether reducing Fast-DENSER, from a five individuals population, to a single-point local search, could speed up neuroevolution even further, without impairing performance.

As this is an experiment reproduction, the parameters used in Fast-DENSER are those used in [5, 21], except those variant-specific ones introduced by this investigation. In the variant where we included recombination, the ES algorithm has a 50/50 chance of choosing between mutation and single-point crossover. Local Search uses a single individual and the algorithm has a $(1, \lambda)$, where λ — the amount of offspring to be generated from the parent — is equal to 1.

Finally, the NTNs visualisations are simplified, reducing data points by decreasing the total number of generations for Fashion-MNIST from 100 to 50. This design choice was assessed empirically and deemed to improve the visualisation with little to no loss of information. The studied variants include elitism and retraining, which in [21] is a function that allows the elite to have the same training time as the parent selected, for mutation or crossover.

7.6 RESULTS

Figures 7.5, 7.6, 7.7, showcase the average evolutionary performance of the variants on the last dataset, Fashion-MNIST. These are, classification accuracy, amount of trainable parameters for the evolved DNNs — as a proxy to how deep the networks are — and the training time required for learning.

The assessment metrics were chosen as considered relevant for testing our hypotheses. Averages in the plots are depicted by lines with markers as the observation points of each generation. Then the runs individual values, at each generation, are overlaid as scatter points — represented with the same markers but hollow and smaller. Being this the last dataset problem, all the incremental development occurred in previous classification tasks will be incorporated in the information at this stage. The expectation is to see the network produced having inherited learnings from past evolutionary stages. The plots are complemented by the averages detailed in Table 7.1, with standard deviation values in brackets.

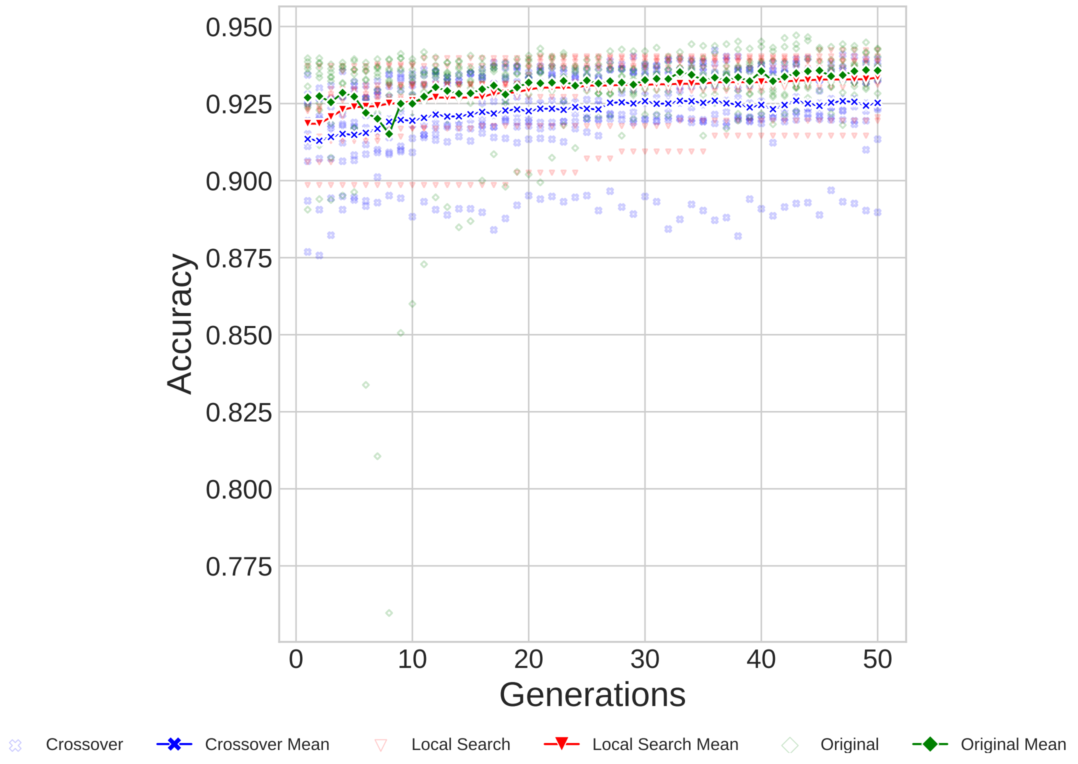


Figure 7.5: Average classification accuracy for all variants on Fashion-MNIST. Individual run values overlaid

The 7.5 plot and the results in Table 7.1 demonstrate that the performance between Original Fast-DENSER ($\lambda = 4$) and our Local Search ($\lambda = 1$) variant are comparable in classification accuracy performance. This shows that reducing the population even further does not impact the algorithms ability to classify images, which we assume is also a consequence of an effective incremental development and the transfer of evolutionary units and learning.

In terms of trainable parameters, in Figure 7.6 we observe that Local Search has a higher number of these, signifying *deeper* networks. The values seen in the average lines indicate that this system is also more stable, generating consistently deep networks, regardless of the higher standard

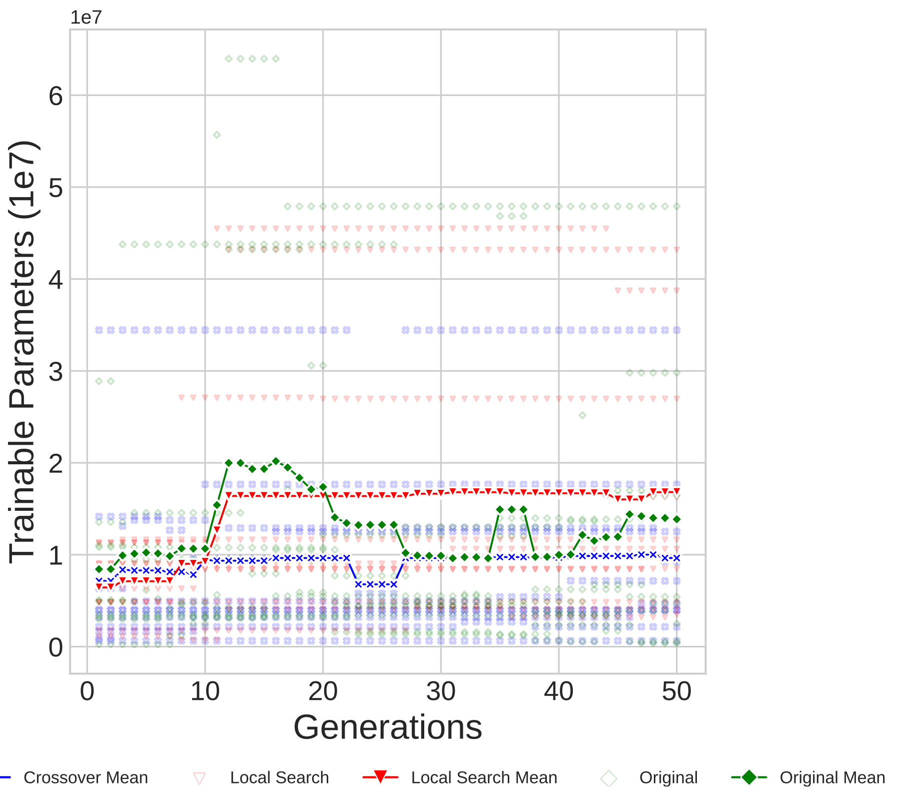


Figure 7.6: Average trainable parameters for all variants on Fashion-MNIST. Individual run values overlaid.

deviation. This stability, seen in the plotted averages — is also present in the classification accuracy, in contrast to the Original variant, which drastically loses performance around the 8th generation.

Furthermore, Figure 7.7 shows that, despite Local Search having a higher number of trainable parameters, this variant is able to train faster without any significant loss in performance. Hence, to sum up, from the evolutionary perspective Local Search appears to be superior in training time and comparable in performance to the original algorithm. Through the test results, it is observed that this variant produces and average accuracy score of 92.73% compared to a 93.27% achieved by the Original algorithm. Fast-DENSER with crossover performed the worst, both in evolutionary and testing setups, producing a 91.87% accuracy. Preliminary assessments through QQ-plots and the Kolmogorov-Smirnov test suggested Gaussian distributions. Therefore we proceeded to conduct a T-test, investigating whether there is any significant difference in classification accuracy between the best performing variants, using a significance level of $p = 0.05$. These tests highlighted that both evolutionary $(p = 0.478)$ and testing $(p = 0.0736)$ accuracy are not significantly different, corroborating to our initial hypothesis that a reduction in population size would not negatively impact the classification performance.

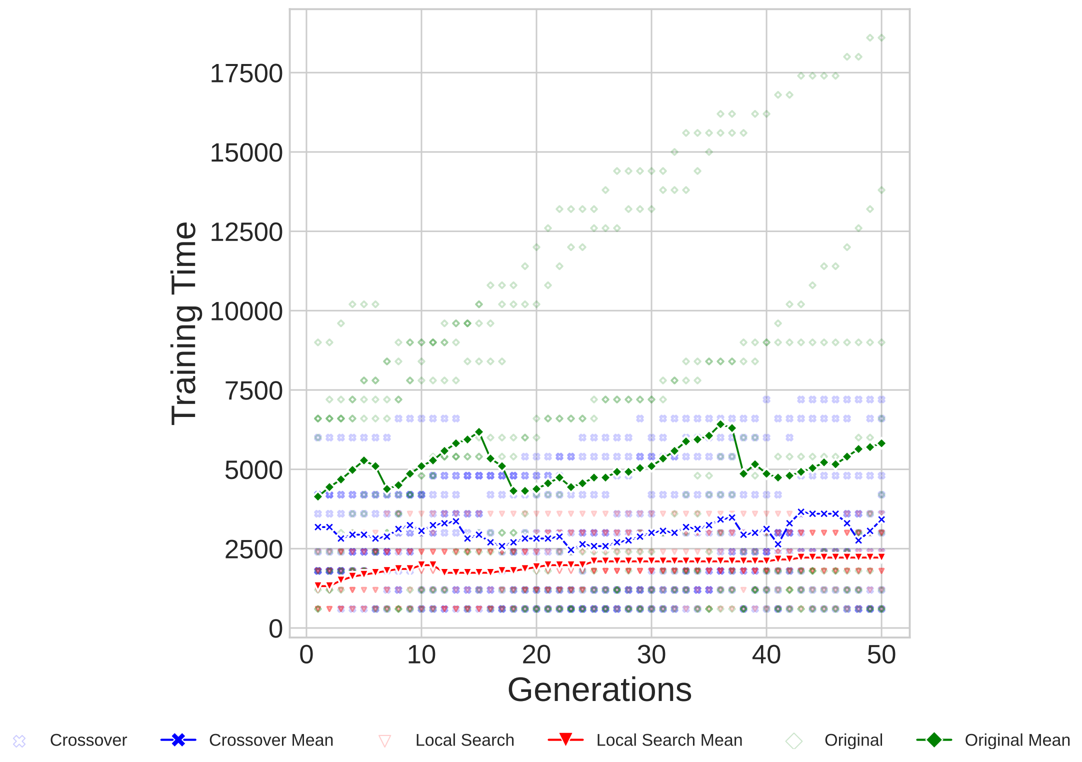


Figure 7.7: Average training time for all variants on Fashion-MNIST. Individual run values overlaid.

Table 7.1: Performance metrics for Fashion-MNIST classification problem.

	Train	Test	Time(sec.)	Trainable Parameters
Local Search$^{(std)}$	$0.9316^{(3.5e-3)}$	$0.9273^{(7.2e-3)}$	$2236^{(479)}$	$1.996318e7^{(8.048529e6)}$
Original$^{(std)}$	$0.9329^{(2.6e-3)}$	$0.9327^{(5.2e-3)}$	$5852^{(1166)}$	$1.406319e7^{(1.703391e6)}$
Crossover$^{(std)}$	$0.9236^{(3.6e-3)}$	$0.9187^{(1.46e-2)}$	$3234^{(412)}$	$1.191039e7^{(1.779603e6)}$

7.6.1 *Revealing transfer learning through NTNs*

The illustrations presented in Figures 7.8, 7.9, 7.10 depict the resulting NTNs generated on the last benchmark classification problem: Fashion-MNIST. They represent each of the three variants for each of the three evolutionary filter groups; making nine visualisation in total. These NTNs are generated from data gathered over 10 runs of 50 generations each. A variant produces a different network, which includes various representative solutions. The structure of NTNs in each group, for the same variant, remains unchanged between datasets categories, to enable a clear comparison of the learning that was transferred, based on the hue decorators used.

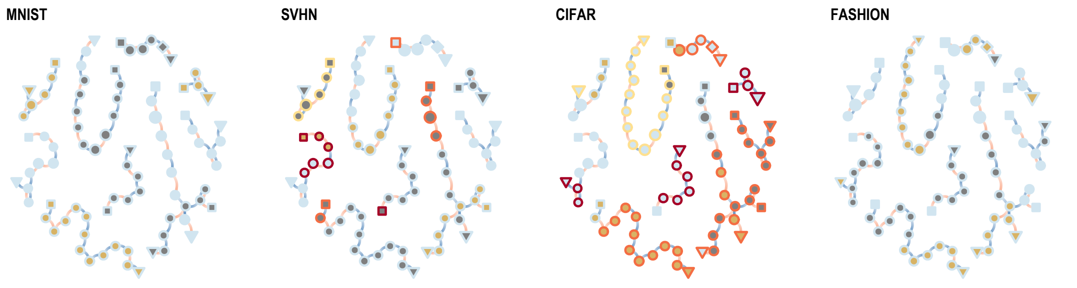
Let us begin by remarking on the networks in general. The predominant findings are that, although the sequence of layers specified by the CFG is concise, all variants traverse this architectural search space in different ways, finding many unique solution and without generating many convergence points to shared solutions. Some nodes are revisited several times in the same trajectory, this reiteration implies successful acceptance of CNN architectures between generations.

As we have seen in the STNs case from previous chapters, NTNs can be seen as a proxy indicator of diversity that an algorithm is able to produce. Bearing this in mind, it is noticeable that although Original Fast-DENSER only leverages the mutation operator at the ES level, it is able to generate high diversity of solutions, both in layers and learning; demonstrated by the computed NTNs metrics, with 84 nodes, 45 improving edges and 34 worsening ones. Confirming that in neuroevolution the absence of crossover can be beneficial to diversity generation. However, the Fast-DENSER variant with recombination does not exhibit a similar characteristic diversity. This is evident from the reduced amount of nodes in the NTNs; 39 nodes, 22 improving edges and 12 worsening ones. Local Search presents the least amount of diversity out of all, 22 nodes, 17 improving edges and 0 worsening ones. In this variant, one of the trajectories shows a strong attraction point of convergence, leading thereafter to multiple ending solutions. Worthy of notice, this is a more resilient algorithm, as no worsening edges are produced between representative solutions.

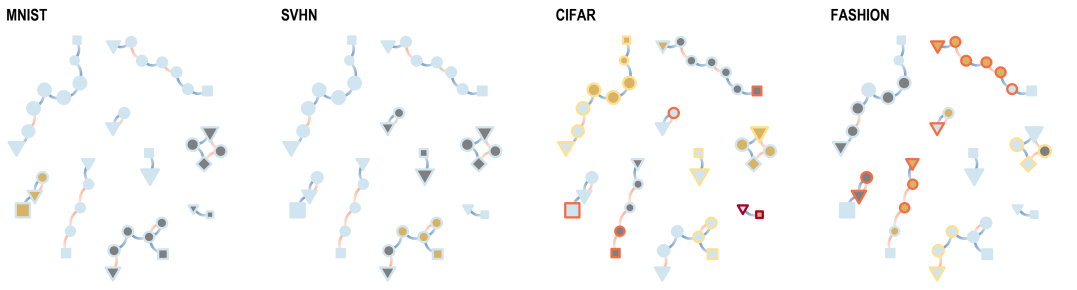
Accordingly with Figure 7.6, the NTNs show that the size of neuroevolved CNNs — represented by the node size — are much greater throughout Local Search than in the other variants. Specifically, via our visualisation approach, it is possible to single those layers acquired through which of the datasets, responsible for size inflation; primarily *fully-connected* and *convolution layers*.

Regarding the learning units that were ported, the vast majority have been derived from the CIFAR-10 classification dataset, closely followed by the current Fashion-MNIST. Gradient-descent appears to be the most transferred learning optimiser, ported from almost every datasets, and present in nearly every variant. This is known to be a powerful optimiser.

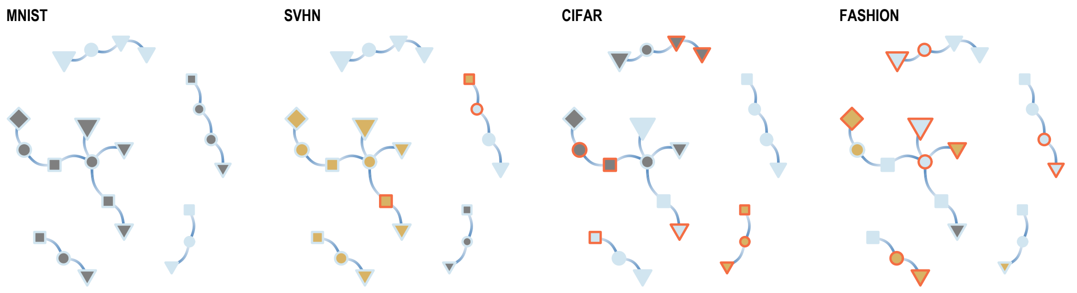
Finally, NTNs trajectories greatly vary in length, between variants. Original, being the most diverse, has the longest trajectories, indicating greater architectural exploration, which may explain the longer training times. Crossover trajectories are less explorative than the original Fast-DENSER, but longer than Local Search, which has the shortest paths. This is explainable by the intrinsic characteristics of this single individual variant.



(a) Group1: Original Fast-DENSER



(b) Group1: Crossover Fast-DENSER



(c) Group1: Local Search Fast-DENSER

Layer ● Absent ● Dropout ● Fully Connected	Learning ○ Absent ○ Adam ○ Grad. Desc. ○ Rmsprop	Edges — improving — worsening	Stage ◇ Best ▽ End ○ Standard □ Start

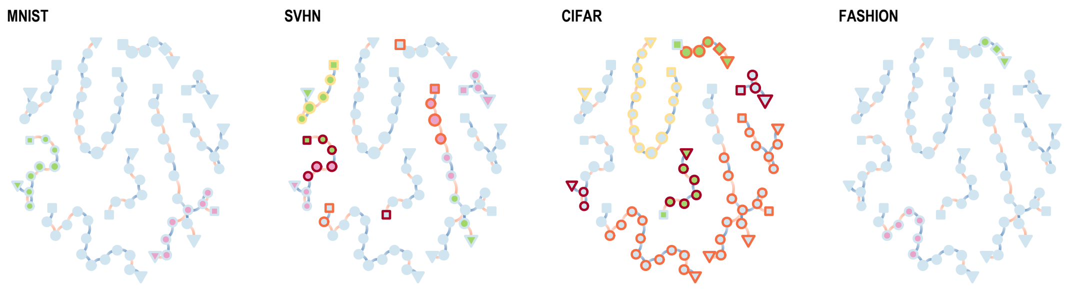
Figure 7.8: Neuroevolution Trajectory Networks showing *group 1* evo. filter for all variants of Fast-DENSER

GROUP 1 From this evo. filter, presented in Figure 7.8 the following results can be derived.

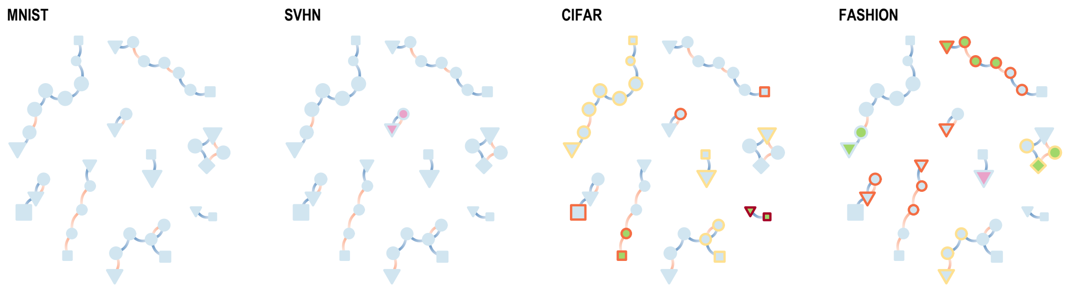
Both from Local Search and the Original variant we can observe that the drop-out layer was correctly deemed by neuroevolution to be a fundamental architectural component for CNNs. The vast majority of nodes include this evolutionary unit. Conversely, the algorithm with recombination does not show a high adoption of these units. This might explain the poor performance seen in the statistical analysis section.

The Original variant successfully acquires fully-connected layers, evident in almost all trajectories and evenly transferred from all past datasets. Crossover does not perform as well, having only a few of these units, ported primarily from the last datasets, CIFAR-10 and Fashion-MNIST.

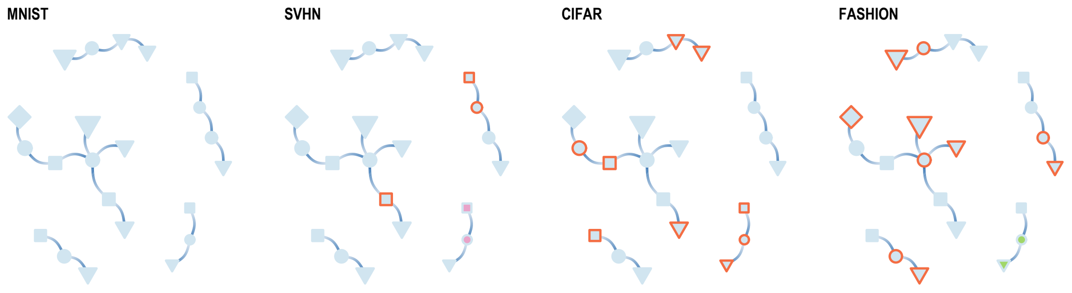
An interesting observation arises from Figure 7.8a: in the left corner of the plot, a trajectory of this variant appears to have inherited and retained *dropout* layers from all past classification learnings, except for the SVHN dataset where all nodes present *fully-connected* units. Inheritance is signalled by the colour of nodes retained throughout each dataset visualisation compartment.



(a) Group2: Original Fast-DENSER



(b) Group2: Crossover Fast-DENSER



(c) Group2: Local Search Fast-DENSER

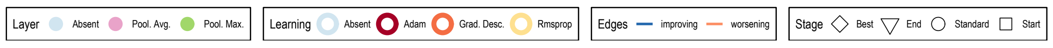


Figure 7.9: Neuroevolution Trajectory Networks showing *group 2* evo. filter for all variants of Fast-DENSER

GROUP 2 From this evo. filter, presented in Figure 7.9 the following results can be derived.

These evolutionary units appear to be lacking in the NTNs, for all variants. This is noteworthy, although not an unforeseen phenomena, as this layer type provides a summarisation — either through average or maximisation — of the feature map of the convolutional layers. This is done in CNNs to reduce the number of parameters to learn from. Although useful, neuroevolution does not view these units as indispensable for successful classifications.

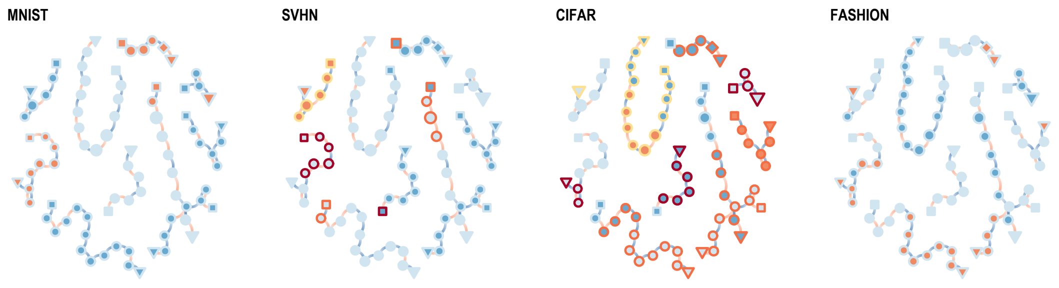
Particularly in Figure 7.9c these layers are almost entirely absent. In the lower right area of the NTNs plots, only one trajectory uses these units in the architectures. Generally these units are primarily found originating from CIFAR-10 and the current dataset, Fashion-MNIST. The Original algorithm in Figure 7.9a is an exception, the variant evolutionary process transferred these units since the very initial MNIST dataset.

GROUP 3 From this evo. filter, presented in Figure 7.10 the following results can be derived.

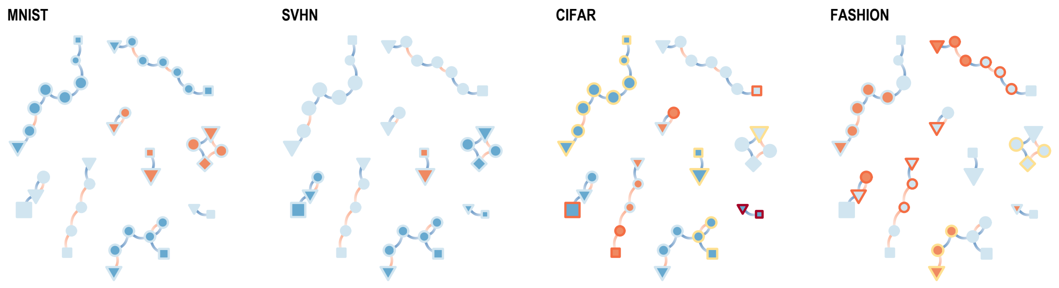
In this analysis, NTNs assist us to confirm that these evolutionary units are fundamental components of the CNNs architectures, as it was retrospectively observable that evolution favoured these units over several iterations from dataset to dataset inheritances. This is noticeable in Figure 7.10a as such units originated from all the past datasets, highlighting their importance in the DNN. Specifically, Original Fast-DENSER shows to have a multitude of these throughout datasets.

The predominance of these evolutionary units demonstrates the importance of these, as essential building blocks for our networks of interest. In the Original Fast-DENSER, the vast majority of these units are transferred from the last two datasets. Local Search, on the other hand, has these units inherited further into past evolutions, since the first two datasets, shown in Figure 7.10c.

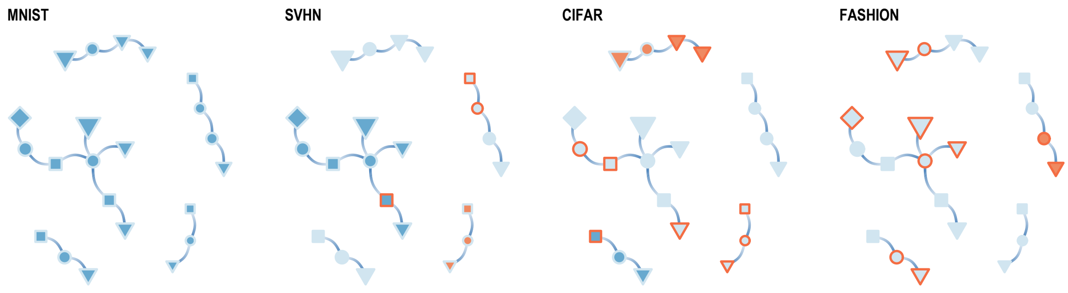
The four trajectories converging to the attracting representative solution in Local Search display *batch normalisation* layers originating from the first dataset MNIST, and *convolution* ones from SVHN. Meanwhile, only two solutions in a single trajectory have the *batch-normalisation* units deriving from the current Fashion-MNIST problem; this strongly indicates that past evolution learning are sufficient for the construction of successful CNNs for the current dataset classification task. This finding would mean that incremental development, which is learning from past training, can significantly speed up convergence to optimal solutions.



(a) Group3: Original Fast-DENSER



(b) Group3: Crossover Fast-DENSER



(c) Group3: Local Search Fast-DENSER

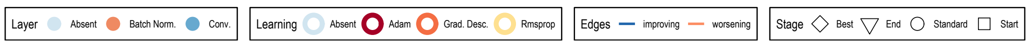


Figure 7.10: Neuroevolution Trajectory Networks showing *group 3* evo. filter for all variants of Fast-DENSER

7.7 CONCLUSIONS

The purpose of this work was to leverage NTNs in the study of transfer learning in CNNs, for image classification. This demonstrates that NTNs are a successful tool for in-depth studies of neuroevolution and could be further enhanced for application in both shallow and deep neural networks. This work strengthened the claim for an NAA practice, which will be discussed in the next and concluding Chapter 8.

The work presented in this chapter focused on the creation of two variants from the original Fast-DENSER algorithm with incremental development.

One variant, equipped with a standard one-point crossover, while another, based on local search principles, these two additional variants were a contribution, as specifically developed for this research. Particularly the local search variant entailed, a single individual selected as the best of five in the initial population was used to spawn an offspring through mutation for a subsequent generation. The motivations for this investigation were to observe whether reducing the population to a single individual could help find solutions faster without impacting accuracy, using fewer resources, and reducing training times. The second intention was to test the usefulness of crossover in this neuroevolution system, as previous studies have highlighted potential inefficiencies of this operator [7, 10]; our intuition is that crossover at the ES level will not improve the system any more that mutation alone. The results on incremental development in this research validated these hypotheses; confirming local search to be a powerful algorithm and that crossover struggles to improve neuroevolution of DNNs for image classification.

The NTNs results were instrumental at identifying the different levels of solutions (architectural) diversity produced by the variants algorithms compared to original Fast-DENSER. Specifically, the models of Local Search highlighted convergence and fewer nodes on average, indicating lower diversity. Additionally, this variant had the least negative exploration (worsening edges) and the average path lengths were notably shorter than those of other variants. This lack of negative exploration could be considered a positive outcome for tasks and domains as such, although it is still an open question, as QD algorithms very much benefit from the increase exploration rising from these learning stepping stones [195]. The original algorithm presented the highest diversity and the most transfer of learning, which often originated from all past datasets. Interesting observations were derived from the general lack and transfer of the pooling layers, which evolutionary processes of incremental development deemed superfluous to optimal CNN architectures.

A further noteworthy trait, emerging from these NTNs, is that the models were able to capture the successful and appropriate transfer of the convolutional unit. This has indicated that NTNs are an advanced tool for the assessment of salient and fundamental architectural components, and that evolution has successfully deemed convolution as a principal component of this network type.

This research claims to demonstrate the power of this analysis type and tool in better understanding DNNs from both architectural and learning perspectives. Information that can be useful to interpret the dynamics of neuroevolution for deep learning, but also in the practice of hand designed networks.

ACKNOWLEDGEMENTS This research is the result of the collaboration with the Departamento de Engenharia Informática of the University of Coimbra, Portugal; supported by the 2022 SPECIES scholarship. This work is funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

CONTRIBUTIONS ACKNOWLEDGEMENTS I hereby declare that the substantial portion of the work detailed in this chapter, as documented in the publications [22, 211], is the result of my own efforts. However, it is important to acknowledge the valuable contributions made by my co-authors in the order of their involvement in this research. Professor Gabriela Ochoa and Dr. Nuno Lourenço offered their strategic assistance towards the research design and their knowledge about STNs and the examined neurovolution algorithms respectively. In addition to offering valuable support in sense checking and proofreading. Dr. Jason Adair and Penousal Machado offered general scientific guidance, expertise in evolutionary computing, as well as proofreading and sense checking.

8

CHAPTER 8 — RESEARCH SYNOPSIS

This chapter provides a synopsis to the research presented in this thesis. Here, we present a concluding summary, reiterating the key contributions of our study on Neuroevolution Trajectory Networks (NTNs), and an answer for the main research hypothesis — presented in Chapter 1. Furthermore, throughout this thesis a discussion was promised concerning the fundamentals of Network Architecture Analysis (NAA), a proposed course of study induced by the principals of this research. Additionally, some concluding remarks will be drawn at the end of this chapter, including a brief reflection on the research journey taken since the start of this Ph.D. degree.
Finally, an outlook on future research is discussed, offering some foreseeable directions that NTNs will likely assume in the future, suggesting longevity to this complex network visualisation practice.

8.1 SUMMARY OF CONTRIBUTIONS

All these work, from the inception of STNs applied to neuroevolution, has been achieved through logical successions of intuitive lines of enquiry and hypotheses that slowly made it possible to converge on the NTNs technique.

Let us walk through the various stages of this research, highlighting once more, the fundamental contributions that this work offered to advance knowledge in this specific field.

BRINGING COMPLEX NETWORKS TO NEUROEVOLUTION

During the initial stages of the work described in this thesis, the analysed literature presented in Chapter 3 highlighted that such visualisation approaches were yet to be leveraged for understanding the complex dynamics occurring in neuroevolution. This work, published in [7], was instrumental in demonstrating the adaptability of STNs visualisations to this realm; modelling through complex networks, the evolution of dense ANNs. The study highlighted inefficiencies of the recombination operator in NEAT. This was the first step towards NTNs. Providing a novel contribution, filling the literature gap.

NTNs to study the role of recombination in Novelty Search

Once established that this tool brought novel and interesting findings related to the inner evolutionary dynamics of NEAT and its crossover operator, the subsequent hypotheses and tentative of the research was to capture the role that crossover plays in more deceiving domain, when a divergent search strategy, that seeks diversity of solutions, is deployed. Also in this scenario, the research work produced a successful publication [10], which revealed salient aspects of the inner dynamics of Novelty Search and its interplay with recombination. The study compared these inner mechanics to a standard, objective-driven strategy and highlighted that, in deceiving maze navigation domains, fitness search equipped with recombination can offer greater diversity. While Novelty Search, stripped of it crossover operator, can perform better in diversity generation. This was another tangible demonstration that NTNs are a useful contribution to the study of this EC field, and that their flexible characteristic can render them applicable to various aspects of neuroevolution.

Offering a topological analysis of Behavioural Characterisations (BC)

This contribution work [11] is what clearly established a boundary between NTNs and the parent STNs visualisation approach. This research allowed for the technique outlined in this thesis to fully concretise, as in this particular study, it is the first time that characteristics, which are solely unique to neuroevolution are modelled through complex networks. The study particularly focused on BCs alignment to the notion of quality. In the experiment, our analysis focused on a topological complexity assessment related to diversity generation. The findings highlighted that topologies fundamentally dictate how distinct behaviours are generated. Especially, shallower (simpler) networks seem to be those able of generating more diversity, ultimately reaching the goal of the reinforcement learning task. These findings led us to propose a *transitive* Behaviour Characterisation, one that is neither aligned or unaligned to the notion of quality, but relates to diversity of the topological structures.

NTNs to track transfer learning in deep neural networks

Another fundamental contribution, which strengthened the case for NTNs and solidified the discipline that will be discussed next, is when the application of such technique shifted from the analysis of shallow/dense neural networks, mainly used in neurocontrolling task, to modelling CNNs for image classification. Additionally, in this research — successfully published and subject of awards [6] — we were able to further augment NTNs to capture the dynamics of a neuroevolution system that incrementally develops solutions. Doing so, to observe the transfer of evolutionary units (layers and learning) from the evolution on past classification datasets. This, not only proved the powers of this complex network modelling and visualisation instrument, but it enabled to confirm DNNs principles, and discover new findings on how neuroevolution favours certain network characteristics over others. Promising results which will be beneficial for the field of Neuroevolution, but also for manually designed DNNs.

8.2 NETWORK ARCHITECTURE ANALYSIS (NNA)

The slow and consistent establishment of the NTNs technique, due to the foundation work on STNs [19, 35] not only allowed for a specialised approach of *search trajectory* visualisations for neural networks evolution; it also catalysed the potential evidence for studies focusing primarily on the analysis of principal architectural components, used in neural networks evolution.

Neural Architecture Search (NAS) [207, 208, 209, 210] is the discipline aimed at developing a wide variety of algorithms for the evolution of structural neural networks components.

This could be the start of an important direction, which has the scope of using visualisations and complex network metrics to demystify the underlying workings and characteristics of architecture compositions. Essentially, NAA can be defined as follows.

> *"The illumination of fundamental neural networks building blocks and their determinative influential power on evolutionary search dynamics"*

The definition above is intended to offer a direct scope for these types of investigations, which can be classified as a branch of *explainability*. This arises from a lack of literature surrounding the topic, and the work provided in this thesis should be seen as a favourable starting point. One that focuses on the analysis of neuroevolution, perfecting the already existing meta-heuristic principles, rather than the constant creation of a plethora of algorithms, which are essentially equal but change only in their name. The tools that have been used throughout this research and the analysis provided, should incentivise the scientific community in neuroevolution to focus more on the "why" of a phenomena producing fitness successfulness, rather than finding more ways to achieve better performance, without fully understanding the reasons for such evolved intelligence.

The work of this thesis, is put together in the hope that NAA may grow to discover more advanced methods of examination and offer more plausible explanations for neuroevolved AI. This could have the potential to also lead towards the discovery of some fundamental principles behind the evolution of intelligence in general. NAA should be seen as closely aligned to the notion of *open-endedness* [91, 203]. Giving particular phenomena sufficient time and freedom of coming to life, float to the surface and reveal the true mechanics of power that are behind them.

In Section 1.2 of Chapter 1 we clearly defined the *main hypothesis* that this research has tested. Successfully, we can state that the thesis presented so far has favourably supported this claim.

The application of STNs will result in a more detailed and informative exploration of the highly-dimensional Search Space of Neuroevolution, compared to traditional fitness performance methods.

8.3 FINAL REMARKS

The intention of this section is to underline concluding takeaway remarks that encapsulate the journey of this thesis, resonating its significance.

Visualisations are a critical medium for understanding our surrounding world; both the directly visible and tangible, but also in the more abstract and distant domains. As we have seen from the Introduction Chapter 1, some historical examples were provided, where observation gathered via visualisation tools, were fundamental in advancing our knowledge in many different scientific fields.

We believe the work of this thesis on STNs [35] and NTNs [7, 10, 11, 22, 211], can be seen as novel advanced tools provided in the quest for understanding search and optimisation algorithms. The ultimate intent is to animate, and pictorially describe the unobservable, shining a light on the evolution of intelligence, or at least set the origin for this.

Through famous cities, it is custom to see panoramic view points, to elevate ourselves on higher grounds, getting vantage over the land below us. We innately do this to understand what surrounds us, and get a better perception of where we are. Sending satellites into space, to observe earth via spectral imaging, so that we can perceive our land, predict the weather via meteorological models and comprehend our intrinsic differences and similarities. Shooting probes through the vastness of our solar system, capturing data to visualise the composition of other planets. On the ground, we use altitude measurements to form contour maps for geological enquiries, to model earths formations through time, and often reach the unperceivable ocean abyss. In more abstract worlds, subatomic particles are accelerated, colliding them together, to form mathematical models and visualisation of their collisions [212].

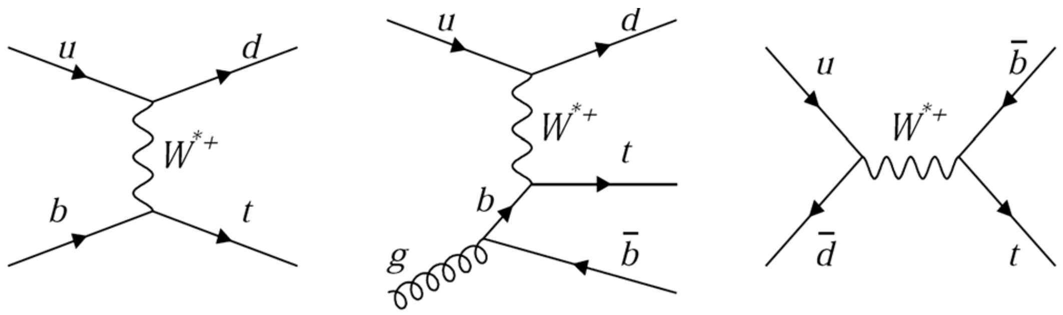


Figure 8.1: A representation of the Feynmann diagrams. Sourced from [12]

There is no better way to further this point across, on modelling and visual observation than the revolutionary work by Richard P. Feynman on the space-time approach to quantum electrodynamics [213]. This work, generated the famously known *Feynman Diagrams* of which examples are illustrated in Figure 8.1. Another striking case where visualisation (in conjunction with mathematics) have been able to guarantee a reliable tool to explain the behaviours of subatomic particles, which are unobservable and can otherwise only be imagined through complex equations.

Feynman diagrams were considered an ingenious idea to describe the interactions of elementary particles. Introduced as an aid for visualising and calculating the effects of electromagnetic interplay, between electrons and photons. Feynman diagrams were later adopted to represent all kinds of particles, and they do so with an astounding precision.

It is safe to say that Feynman developed a visualisation and calculation tool that gave life to unobservable phenomena, with a spatial and temporal component.

The intent here is not to directly compare NTNs to the *Diagrams* — that would be overambitious — but to present a case that these types of explorative visualisations, and NAA could be the means to fuel further pioneering ideas in the field of evolutionary computation for neural networks.

Network visualisations, such as NTNs, in these realms demonstrate that, as seen during the work of this dissertation, such a tool can be highly malleable and dynamic. If the NTNs principles declared in Section 2.1.3 are followed, and the signature used in the modelling is correctly conceptualised based on the underlying principles of the neuroevolution algorithm, the tool can be applied to a vast array of systems that are yet to be analysed. The characteristic of universal applicability is a great benefit for this tool, as the simple principles, strongly rooted in mathematical theory, allow it to be generalisable and useful for many researchers. For example, the tool is well-suited for understanding algorithms such as CPPNs [118] and derivations such as HyperNEAT; as well as *Adaptive* HyperNEAT, which includes pattern or local plasticity rules. Among other examples of the applicability of this tool, it has been used in NAS, where some tests were conducted during this research but not reported, as incomplete. Furthermore, the multidimensional heat map of algorithms such as MAP-Elites and its derivations could be contrasted and analysed further using NTNs.

Furthermore, a benefit of this technique, which emerged from this dissertation, is the computed metrics derivable from the models. That is, if complex networks like NTNs are too complex to visualise, as they include too many data points and algorithmic runs, a simpler sub-selection of data can be visualised — as it has been done in this thesis — while still offering metrics of the NTNs for the full dataset, which includes all the available runs/data.

Despite the reported successes of this technique in its application to the Neuroevolution realm, several fundamental limitations exist, which, if addressed by further extensions, may improve this approach. One of the principal limitations highlighted by this research, which is yet to be fully tested and confirmed, is whether this technique would break at the limit and fail to provide consistent and useful analysis results. This limit is considered to be the transition from what are perceived to be *toy* problems and simple benchmarks to complex and real-world tasks and domains where novelty descriptors and solutions are a lot harder to conceptualise, and domains are vastly more highly dimensional.

Furthermore, although the networks supply us with useful statistical information on the evolutionary dynamics, depicted by trajectories of multiple runs, including neutrality, convergence of solutions, and diversity; it is not currently possible to drill into the finer details of what makes solutions different in terms of architectural or tunable components. The dimensionality is large, and displaying this granular level of detail is still an open issue, which, if resolved, could enrich the visualisations even further.

Secondly, we have seen throughout this thesis that the signature choice is highly dependent on the underlying neuroevolutionary system in analysis, and it is heavily dictated by empirical processes. There are several choices such as design, layout, and parameters related to the complex network's construction and visualisation, which are dictated by the researcher. It would be a favourable step towards strengthening and standardising these techniques if stricter postulates were established for specific research cases.

A further related limitation is that these controllable experimental settings and design choices are not benefiting from automation or heuristic search. If there were ways of conceptualising a fitness function based on the aforementioned postulates and pre-defined rules, we could let evolution dictate these choices and evolve the visualisations, reaching optimal and meaningful constructs, depicting the true nature of the underlying neuroevolution systems.

Finally, the trade-off between too little and too much information has been discussed at several stages in this dissertation. Said delicate balance prevents us from introducing a higher number of runs into the visualisations. This is a limitation that, if solved, could introduce much greater statistical strength to the NTNs. The point on evolution mentioned earlier could possibly help to solve this problem through meta-heuristic search.

8.4 FUTURE OUTLOOK

A synopsis would not be complete without a view on the future. An outlook on the opportunities and challenges that this research might witness flourish. Some of the points for future work are expressed in each of the publications [6, 7, 10, 11, 211]. Several of these lines of enquiry have already been explored, some successfully, others without considerable results, and meanwhile some are currently being investigated.

In favour of being concrete and concise, three main avenues of interest could further enhance this research.
(i) Explore the possibility of increasing the amount of information that NTNs can convey within a single plot. Intelligently, orchestrate a blend of decorators, labels and metrics that would maximise the information delivery of these complex networks.
(ii) Discover effective methods for integrating temporal information in the plots, emphasising the importance of time/iteration components in the search dynamics. Utilise the spatial coordinates of the plot (something that

has not been used much in the layouts) to convey intrinsic information about the evolution of neural network architectures.
(iii) Investigate the development of NTNs that evolve through co-evolutionary means. A conceptual way of bringing evolutionary optimisation to NTNs. The idea is to allow the layout, decorators and overall composition of NTNs to evolve and optimise, generating visualisation networks that are incrementally constructed to include essential information, determining the underlying structure of the search process. This would, in turn, generate artefacts which include evolutionary dynamics, directly into the plots. Our hypothesis is that this would generate more comprehensible visualisations, dictated by what evolution selects to present.

"We do not know what we are capable of, until we do it"

BIBLIOGRAPHY

[1] R. Brooks and J. P. Matelski, "The dynamics of 2-generator subgroups of psl (2, c)," in *Riemann surfaces and related topics: Proceedings of the 1978 Stony Brook Conference, Ann. of Math. Stud*, vol. 97, 1981, pp. 65–71.

[2] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[3] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary Computation*, vol. 19, no. 2, pp. 189–222, 2011.

[4] J. Mégane, N. Lourenço, and P. Machado, "Probabilistic grammatical evolution," in *Genetic Programming*, T. Hu, N. Lourenço, and E. Medvet, Eds. Cham: Springer International Publishing, 2021, pp. 198–213.

[5] F. Assunção, N. Lourenço, B. Ribeiro, and P. Machado, "Incremental evolution and development of deep artificial neural networks," in *Genetic Programming*, T. Hu, N. Lourenço, E. Medvet, and F. Divina, Eds. Cham: Springer International Publishing, 2020, pp. 35–51.

[6] S. Sarti, N. Laurenço, J. Adair, P. Machado, and G. Ochoa, "Under the hood of transfer learning for deep neuroevolution," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2023, pp. 640–655.

[7] S. Sarti and G. Ochoa, "A NEAT Visualisation of Neuroevolution Trajectories," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12694 LNCS, 2021, pp. 714–728.

[8] E. M. Reingold and J. S. Tilford, "Tidier Drawings of Trees," *IEEE Transactions on Software Engineering*, vol. SE-7, no. 2, pp. 223–228, 1981.

[9] I. Omelianenko, *Hands-On Neuroevolution with Python*. Packt Publishing, Limited, 2019.

[10] S. Sarti, J. Adair, and G. Ochoa, "Recombination and novelty in neuroevolution: A visual analysis," *SN Computer Science*, vol. 3, no. 3, p. 185, Mar 2022.

[11] ——, "Neuroevolution trajectory networks of the behaviour space," in *Applications of Evolutionary Computation*, J. L. Jiménez Laredo, J. I. Hidalgo, and K. O. Babaagba, Eds. Cham: Springer International Publishing, 2022, pp. 685–703.

[12] T. Aaltonen, S. Amerio, D. Amidei, and e. a. Anastassov, A., "Measurement of the single top quark production cross section and $|V_{tb}|$ in 1.96 tev $p\overline{p}$ collisions with missing transverse energy and jets and final cdf combination," *Phys. Rev. D*, vol. 93, p. 032011, Feb 2016.

[13] T. E. H. T. Collaboration, K. Akiyama, A. Alberdi, and e. a. Walter Alef, "First m87 event horizon telescope results. i. the shadow of the supermassive black hole," *The Astrophysical Journal Letters*, vol. 875, no. 1, p. L1, apr 2019.

[14] B. Mandelbrot and R. Hudson, *The Misbehaviour of Markets: A Fractal View of Risk, Ruin and Reward*. Profile, 2004.

[15] D. Saupe and R. Hamzaoui, "A review of the fractal image compression literature," *SIGGRAPH Comput. Graph.*, vol. 28, no. 4, p. 268–276, nov 1994.

[16] M. Barnsley, *Fractals Everywhere*, ser. Dover Books on Mathematics. Dover Publications, 2012.

[17] M. E. J. Newman, *Networks: an introduction*. Oxford; New York: Oxford University Press, 2010.

[18] K. Sörensen, "Metaheuristics - the metaphor exposed," *Int. Trans. Oper. Res.*, vol. 22, pp. 3–18, 2015.

[19] G. Ochoa, K. M. Malan, and C. Blum, "Search Trajectory Networks of Population-Based Algorithms in Continuous Spaces," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12104 LNCS, pp. 70–85, 2020.

[20] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "DENSER: deep evolutionary network structured representation," *Genetic Programming and Evolvable Machines*, vol. 20, no. 1, pp. 5–35, 2019.

[21] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Fast denser: Efficient deep neuroevolution," in *Genetic Programming*, L. Sekanina, T. Hu, N. Lourenço, H. Richter, and P. García-Sánchez, Eds. Cham: Springer International Publishing, 2019, pp. 197–212.

[22] S. Sarti, N. Laurenço, J. Adair, P. Machado, and G. Ochoa, "Under the hood of transfer learning for deep neuroevolution," in *Applications of Evolutionary Computation*, J. Correia, S. Smith, and R. Qaddoura, Eds. Cham: Springer Nature Switzerland, 2023, pp. 640–655.

[23] D. Ha, "Neural network evolution playground with backprop neat," *blog.otoro.net*, 2016.

[24] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[25] J. Lehman, K. O. Stanley *et al.*, *Exploiting open-endedness to solve problems through the search for novelty.*, 2008.

[26] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local comp," no. Gecco, 2011, pp. 211–218.

[27] J.-B. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evolutionary computation*, vol. 20, no. 1, pp. 91–133, 2012.

[28] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," pp. 1–15, 2015.

[29] K. Chatzilygeroudis, A. Cully, V. Vassiliades, and J.-B. Mouret, "Quality-diversity optimization: a novel branch of stochastic optimization," in *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. Springer, 2021, pp. 109–135.

[30] S. Risi and K. O. Stanley, "Indirectly encoding neural plasticity as a pattern of local rules," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6226 LNAI, no. Sab, pp. 533–543, 2010.

[31] A. Soltoggio, K. O. Stanley, and S. Risi, "Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks," *Neural Networks*, vol. 108, pp. 48–67, 2018.

[32] E. Papavasileiou, J. Cornelis, and B. Jansen, "A systematic literature review of the successors of "neuroevolution of augmenting topologies"," *Evolutionary Computation*, vol. 29, no. 1, pp. 1–73, 2021.

[33] G. Ochoa, M. Tomassini, S. Verel, and C. Darabos, "A study of nk landscapes' basins and local optima networks," pp. 555–562, 2008.

[34] G. Ochoa, N. Veerapen, F. Daolio, and M. Tomassini, “Understanding phase transitions with local optima networks: Number partitioning as a case study,” in *Evolutionary Computation in Combinatorial Optimization, EvoCOP*, ser. Lecture Notes in Computer Science, vol. 10197, 2017, pp. 233–248.

[35] G. Ochoa, K. M. Malan, and C. Blum, “Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics,” *Applied Soft Computing*, vol. 109, no. May, 2021.

[36] V. Narvaez-Teran, G. Ochoa, and E. Rodriguez-Tello, “Search Trajectory Networks Applied to the Cyclic Bandwidth Sum Problem,” *IEEE Access*, vol. 9, pp. 1–1, 2021.

[37] S. Sarti and G. Ochoa, “A NEAT visualisation of neuroevolution trajectories,” in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science, vol. 12694. Springer, 2021, pp. 714–728.

[38] T. Lewis, *Network Science: Theory and Practice*. Wiley, 2009.

[39] L. Euler, “Solutio problematis ad geometriam situs pertinentis,” *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140, 1741.

[40] N. L. Biggs, “T. p. kirkman, mathematician,” *Bulletin of the London Mathematical Society*, vol. 13, no. 2, pp. 97–120, 1981.

[41] E. C. Kirby, R. B. Mallion, P. Pollak, and P. J. Skrzyński, “What kirchhoff actually did concerning spanning trees in electrical networks and its relationship to modern graph-theoretical work,” *Croatica Chemica Acta*, vol. 89, no. 4, pp. 403–417, 2016.

[42] D. Rouvray, “The pioneering contributions of cayley and sylvester to the mathematical description of chemical structure,” *Journal of Molecular Structure: THEOCHEM*, vol. 185, pp. 1–14, 1989.

[43] W. Whewell, *On the philosophy of discovery: chapters historical and critical*. JW Parker and son, 1860.

[44] M. E. J. Newman, “The structure and function of complex networks,” *SIAM Rev.*, vol. 45, pp. 167–256, 2003.

[45] S. L. Thomson, “Anatomy of the local optima level in combinatorial optimisation,” 2020.

[46] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos, “A study of nk landscapes' basins and local optima networks,” in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 555–562.

[47] G. Ochoa, K. M. Malan, and C. Blum, “Search trajectory networks of population-based algorithms in continuous spaces,” in *Applications of Evolutionary Computation, EvoApps*, ser. Lecture Notes in Computer Science, vol. 12104. Springer, 2020, pp. 70–85.

[48] D. Davendra, *Traveling salesman problem: Theory and applications*. BoD–Books on Demand, 2010.

[49] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Springer Publishing Company, Incorporated, 2015.

[50] N. J. Radcliffe, “Genetic set recombination and its application to neural network topology optimisation,” *Neural Computing & Applications*, vol. 1, pp. 67–90, 1993.

[51] J. M. Sobel, G. F. Chen, L. R. Watt, and D. W. Schemske, “The biology of speciation,” *Evolution*, vol. 64, no. 2, pp. 295–315, 2010.

[52] J. Lehman and K. O. Stanley, “Abandoning objectives: Evolution through the search for novelty alone,” *Evolutionary Computation*, vol. 19, no. 2, pp. 189–222, 2011.

[53] S. Doncieux, A. Laflaquière, and A. Coninx, “Novelty search: a theoretical perspective,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 99–106.

[54] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality diversity: A new frontier for evolutionary computation,” *Frontiers Robotics AI*, vol. 3, no. JUL, pp. 1–17, 2016.

[55] E. Meyerson, J. Lehman, and R. Miikkulainen, “Learning behavior characterizations for novelty search,” 2016, pp. 149–156.

[56] K. Xu, Y. Ma, and W. Li, “Dynamics-aware novelty search with behavior repulsion,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1112–1120. [Online]. Available: https://doi.org/10.1145/3512290.3528761

[57] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, “Denser: Deep evolutionary network structured representation,” *Genetic Programming and Evolvable Machines*, vol. 20, 03 2019.

[58] C. Ryan, J. Collins, and M. O. Neill, “Grammatical evolution: Evolving programs for an arbitrary language,” in *Genetic Programming*, W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 83–96.

[59] F. Rothlauf and M. Oetzel, “On the locality of grammatical evolution,” in *Genetic Programming*, P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Ekárt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 320–330.

[60] P. Whigham, G. Dick, J. Maclaurin, and C. Owen, “Examining the ?best of both worlds? of grammatical evolution,” *Proceedings of the 2015 Genetic and Evolutionary Computation*, vol. 2015, pp. 1111–1118, 2015.

[61] N. Lourenço, F. B. Pereira, and E. Costa, “Sge: A structured representation for grammatical evolution,” in *Artificial Evolution*, S. Bonnevay, P. Legrand, N. Monmarché, E. Lutton, and M. Schoenauer, Eds. Cham: Springer International Publishing, 2016, pp. 136–148.

[62] N. Lourenço, F. Assunção, F. B. Pereira, E. Costa, and P. Machado, *Structured Grammatical Evolution: A Dynamic Approach*. Cham: Springer International Publishing, 2018, pp. 137–161.

[63] M. Valueva, N. Nagornov, P. Lyakhov, G. Valuev, and N. Chervyakov, “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,” *Mathematics and Computers in Simulation*, vol. 177, pp. 232–243, 2020.

[64] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[65] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[66] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[67] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[68] X. Yao, "A Review of Evolutionary Arti cial Neural."

[69] E. Ronald and M. Schoenauer, "Genetic lander: An experiment in accurate neuro-genetic control," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 866 LNCS, pp. 452–461, 1994.

[70] F. Gruau and U. C. B.-l. I, "Thesis Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm," *Synthesis*, 1994.

[71] J. C. Figueira Pujol and R. Poli, "Evolving the Topology and the Weights of Neural Networks Using a Dual Representation," *Applied Intelligence*, vol. 8, no. 1, pp. 73–84, 1998.

[72] K. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019, cited By 77.

[73] G.-A. Vargas-Hákim, E. Mezura-Montes, and H.-G. Acosta-Mesa, "A review on convolutional neural network encodings for neuroevolution," pp. 12–27, 2022.

[74] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, p. 47–62, Jan 2008.

[75] S. van Steenkiste, J. Koutník, K. Driessens, and J. Schmidhuber, "A wavelet-based encoding for neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 517–524.

[76] E. Hastings, R. Guha, and K. Stanley, "Automatic content generation in the galactic arms race video game," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 245–263, 2009.

[77] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, 2005.

[78] A. Hoover and K. Stanley, "Exploiting functional relationships in musical composition," *Connection Science*, vol. 21, no. 2-3, pp. 227–251, 2009.

[79] H. Dinh, N. Aubert, N. Noman, T. Fujii, Y. Rondelez, and H. Iba, "An effective method for evolving reaction networks in synthetic biochemical systems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 374–386, 2015.

[80] G. Wang, G. Cheng, and T. Carr, "The application of improved neuroevolution of augmenting topologies neural network in marcellus shale lithofacies prediction," *Computers and Geosciences*, vol. 54, pp. 50–65, 2013.

[81] J. Nadkarni and R. Ferreira Neves, "Combining neuroevolution and principal component analysis to trade in the financial markets," *Expert Systems with Applications*, vol. 103, pp. 184–195, 2018.

[82] T. Aaltonen, J. Adelman, T. Akimoto, and e. a. Albrow, "Measurement of the top-quark mass with dilepton events selected using neuroevolution at CDF," *Physical Review Letters*, vol. 102, no. 15, pp. 1–7, 2009.

[83] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *Journal of artificial intelligence research*, vol. 21, pp. 63–100, 2004.

[84] F. Silva, P. Urbano, L. Correia, and A. L. Christensen, "odneat: An algorithm for decentralised online evolution of robotic controllers," *Evolutionary Computation*, vol. 23, no. 3, p. 421–449, Sep 2015.

[85] F. Silva, L. Correia, and A. L. Christensen, "Evolutionary online behaviour learning and adaptation in real robots," *Royal Society Open Science*, vol. 4, no. 7, p. 160938, Jul 2017.

[86] N. T. Siebel and G. Sommer, "Evolutionary reinforcement learning of artificial neural networks," *International Journal of Hybrid Intelligent Systems*, vol. 4, no. 3, pp. 171–183, 2007.

[87] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2902–2911.

[88] V. Costa, N. Lourenço, and P. Machado, "Coevolution of generative adversarial networks," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2019, pp. 473–487.

[89] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 2019, pp. 293–312.

[90] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[91] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nature Machine Intelligence*, vol. 2, pp. 24–35, 2019.

[92] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," pp. 1–13, 2017.

[93] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[94] V. Mnih, A. P. Badia, L. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *33rd International Conference on Machine Learning, ICML 2016*, vol. 4, pp. 2850–2869, 2016.

[95] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning," 2017.

[96] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 3215–3222, 2018.

[97] J. Lehman, J. Chen, J. Clune, and K. O. Stanley, "Safe mutations for deep and recurrent neural networks through output gradients," *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*, pp. 117–124, 2018.

[98] T. Gangwani, C. Science, J. Peng, and C. Science, "Policy Optimisation by Genetic Distillation," 2018.

[99] D. S. Reilstad, "Cultivating diversity: a comparison of diversity objectives in neuroevolution," Master's thesis, 2023.

[100] J.-B. Mouret and S. Doncieux, "Using behavioral exploration objectives to solve deceptive problems in neuro-evolution," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 627–634.

[101] J. Lehman and K. O. Stanley, "Exploiting open-endedness to solve problems through the search for novelty," no. Alife Xi, 2008, pp. 329–336.

[102] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," *Advances in Neural Information Processing Systems*, vol. 2018-December, no. Nips, pp. 5027–5038, 2018.

[103] D. Gravina, A. Liapis, and G. N. Yannakakis, "Surprise search: Beyond objectives and novelty," *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, no. July, pp. 677–684, 2016.

[104] H. Mengistu, J. Lehman, and J. Clune, "Evolvability search: Directly selecting for evolvability in order to study and produce it," *GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, no. July, pp. 141–148, 2016.

[105] C. Stanton and J. Clune, "Deep Curiosity Search: Intra-Life Exploration Can Improve Performance on Challenging Deep Reinforcement Learning Problems," 2018.

[106] A. Cully, J. Clune, D. Tarapore, and J. B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[107] J. Huizinga, J. B. Mouret, and J. Clune, "Does aligning phenotypic and genotypic modularity improve the evolution of neural networks?"

GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference, pp. 125–132, 2016.

[108] A. Nguyen, J. Yosinski, and J. Clune, "Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning," *Evolutionary Computation*, vol. 24, no. 3, pp. 545–572, 2016.

[109] B. Hodjat, H. Shahrzad, and R. Miikkulainen, "Distributed age-layered novelty search," *Proceedings of the Artificial Life Conference 2016, ALIFE 2016*, 2016.

[110] J. C. Brant and K. O. Stanley, "Minimal criterion coevolution: A new approach to open-ended search," *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, no. Gecco, pp. 67–74, 2017.

[111] L. K. Le Goff, E. Hart, A. Coninx, and S. Doncieux, "On pros and cons of evolving topologies with novelty search," *The 2020 Conference on Artificial Life*, 2020.

[112] A. M. Turing, "The chemical basis of morphogenesis," *Bulletin of Mathematical Biology*, vol. 52, no. 1-2, pp. 153–197, 1990.

[113] J. C. Bongard and R. Pfeifer, "Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny," no. 1998, 2000.

[114] F. Gruau, "Automatic De nition of Modular Neural Networks Abstract matic de ntion of sub-neural networks ." *Writing*, pp. 1–44, 1995.

[115] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," *Artificial Life*, vol. 9, no. 2, pp. 93–130, 2003.

[116] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development."

[117] J. Gauci and K. O. Stanley, "Autonomous evolution of topographic regularities in artificial neural networks," *Neural Computation*, vol. 22, no. 7, pp. 1860–1898, 2010.

[118] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, no. 2, pp. 185–212, 2009.

[119] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria, "On the performance of indirect encoding across the continuum of regularity," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 346–367, 2011.

[120] J. Huizinga, J. B. Mouret, and J. Clune, "Evolving neural networks that are both modular and regular: Hyperneat plus the connection cost technique," *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, no. July, pp. 697–704, 2014.

[121] E. Buchanan, L. K. Le Goff, W. Li, E. Hart, A. E. Eiben, M. De Carlo, A. F. Winfield, M. F. Hale, R. Woolley, M. Angus, J. Timmis, and A. M. Tyrrell, "Bootstrapping artificial evolution to design robots for autonomous fabrication," *Robotics*, vol. 9, no. 4, pp. 1–24, 2020.

[122] L. K. L. Goff, E. Buchanan, E. Hart, A. E. Eiben, W. Li, M. De Carlo, A. F. Winfield, M. F. Hale, R. Woolley, M. Angus, J. Timmis, and A. M. Tyrrell, "Morpho-evolution with learning using a controller archive as an inheritance mechanism," pp. 1–15, 2021.

[123] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 427–436, 2015.

[124] F. Liu, Q. Song, G. Wen, J. Cao, and X. Yang, "Bipartite synchronization in coupled delayed neural networks under pinning control," *Neural Networks*, vol. 108, pp. 146–154, 2018.

[125] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Using GP Is NEAT: Evolving compositional pattern production functions," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10781 LNCS, no. January, pp. 3–18, 2018.

[126] S. Risi and K. O. Stanley, "An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density, and Connectivity of Neurons," vol. 363, pp. 331–363, 2012.

[127] ——, "A unified approach to evolving plasticity and neural geometry," *Proceedings of the International Joint Conference on Neural Networks*, no. Ijcnn, 2012.

[128] D. Floreano and F. Mondada, "Evolution of Plastic Neurocontrollers for Situated Agents," *From Animals to Animats 4*, no. May 2014, 1994.

[129] D. Floreano and J. Urzelai, "Evolutionary robots with on-line self-organization and behavioral fitness," *Neural Networks*, vol. 13, no. 4-5, pp. 431–443, 2000.

[130] F. Attneave, M. B., and D. O. Hebb, "The Organization of Behavior; A Neuropsychological Theory," *The American Journal of Psychology*, vol. 63, no. 4, p. 633, 1950.

[131] B. Inden, "Neuroevolution and complexifying genetic architectures for memory and control tasks," *Theory in Biosciences*, vol. 127, no. 2, pp. 187–194, 2008, cited By 4.

[132] F. Gallego-Durán, R. Molina-Carmona, and F. Llorens-Largo, "Experiments on neuroevolution and online weight adaptation in complex environments," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8109 LNAI, pp. 131–138, 2013, cited By 0.

[133] P. Tonelli and J. B. Mouret, "On the relationships between generative encodings, regularity, and learning abilities when evolving plastic artificial neural networks," *PLoS ONE*, vol. 8, no. 11, 2013.

[134] J. B. Mouret and P. Tonelli, "Artificial evolution of plastic neural networks: A few key concepts," *Studies in Computational Intelligence*, vol. 557, pp. 251–261, 2015.

[135] R. Greve, E. Jacobsen, and S. Risi, "Evolving neural turing machines for reward-based learning," F. T., Ed. Association for Computing Machinery, Inc, 2016, pp. 117–124, cited By 14.

[136] B. Lüders, M. Schläger, A. Korach, and S. Risi, "Continual and one-shot learning through neural networks with dynamic external memory," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10199 LNCS, pp. 886–901, 2017, cited By 6.

[137] K. Nguyen and Y. Choe, "Dynamic control using feedforward networks with adaptive delay and facilitating neural dynamics," vol. 2017-May. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 2987–2994, cited By 1.

[138] K. Olav Ellefsen, J.-B. Mouret, and J. Clune, "Neural Modularity Helps Organisms Evolve to Learn New Skills without Forgetting Old Skills," *Comput Biol*, vol. 11, no. 4, p. 1004128, 2015.

[139] J. Clune, J. B. Mouret, and H. Lipson, "Summary of the evolutionary origins of modularity," *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion*, p. 23, 2013.

[140] R. Velez and J. Clune, "Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks," 2017.

[141] A. Sboev, A. Serenko, R. Rybka, D. Vlasov, and A. Filchenkov, "Estimation of the influence of spiking neural network parameters on classification accuracy using a genetic algorithm," S. A.V., Ed., vol. 145. Elsevier B.V., 2018, pp. 488–494, cited By 3.

[142] I. Showalter and H. Schwartz, "Neuromodulated multiobjective evolutionary neurocontrollers without speciation," *Evolutionary Intelligence*, 2020, cited By 0.

[143] H. Qiu, M. Garratt, D. Howard, and S. Anavatti, "Towards crossing the reality gap with evolved plastic neurocontrollers." Association for Computing Machinery, 2020, pp. 130–138, cited By 0.

[144] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pp. 4780–4789, 2019.

[145] A. Rawal and R. Miikkulainen, "From Nodes to Networks: Evolving Recurrent Neural Networks," 2018.

[146] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, 2017.

[147] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 4278–4284, 2017.

[148] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Fast-denser++: Evolving fully-trained deep artificial neural networks," *arXiv preprint arXiv:1905.02969*, 2019.

[149] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pp. 1–16, 2017.

[150] P. Feldmeier, “Fully automated game testing via neuroevolution,” in *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, 2023, pp. 486–488.

[151] D. Zimmermann, P. Deubel, and A. Koziolek, “Evaluating the effectiveness of neuroevolution for automated gui-based software testing,” in *2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, 2023, pp. 119–126.

[152] E. Otović, J. Lerga, D. Kalafatovic, and G. Mauša, “Neuroevolution for the sustainable evolution of neural networks,” in *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, 2023, pp. 1045–1051.

[153] D. D. Kulkarni and S. B. Nair, “Transfer learning for embodied neuroevolution,” in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO ’23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 2128–2135.

[154] Y. Qiao and M. Gallagher, “Modularity based linkage model for neuroevolution,” in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO ’23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 675–678.

[155] M. Merten, R. Krauss, and R. Drechsler, “Scalable neuroevolution of ensemble learners,” in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO ’23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 667–670.

[156] M. Le Clei and P. Bellec, “Generative adversarial neuroevolution for control behaviour imitation,” in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO ’23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 663–666.

[157] Z. Shuai and F. Chen, “Automatic lightweight yolo construction through neuroevolution for white shrimps detection,” in *2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, 2023, pp. 675–682.

[158] A. J. Alburghaif, M. A. Balafar, and J. P. Tanha, "Neae: Neuroevolution autoencoder for anomaly detection in internet traffic data," *The Journal of Supercomputing*, Oct 2023.

[159] J. Karns and T. Desell, "Local stochastic differentiable architecture search for memetic neuroevolution algorithms," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 2123–2127.

[160] J. P. Doye, "Network topology of a potential energy landscape: A static scale-free network," *Physical review letters*, vol. 88, no. 23, p. 238701, 2002.

[161] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger, "Barrier trees of degenerate landscapes," *Zeitschrift für Physikalische Chemie*, vol. 216, no. 2, Jan 2002.

[162] O. M. Becker and M. Karplus, "The topology of multidimensional potential energy surfaces: Theory and application to peptide structure and kinetics," *The Journal of Chemical Physics*, vol. 106, no. 4, p. 1495–1517, Jan 1997.

[163] J. P. K. Doye, M. A. Miller, and D. J. Wales, "The double-funnel energy landscape of the 38-atom lennard-jones cluster," *The Journal of Chemical Physics*, vol. 110, no. 14, p. 6896–6906, Apr 1999.

[164] J. Hallam and A. Prugel-Bennett, "Large barrier trees for studying search," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, p. 385–397, Aug 2005.

[165] S. L. Thomson, N. Veerapen, G. Ochoa, and D. van den Berg, "Randomness in local optima network sampling," in *GECCO'23 Companion: Genetic and Evolutionary Computation Conference Companion*, 2023.

[166] S. L. Thomson, G. Ochoa, N. Veerapen, and K. Michalak, "Channel configuration for neural architecture: Insights from the search space," in *GECCO'23: Genetic and Evolutionary Computation Conference*, 2023.

[167] N. M. Rodrigues, K. M. Malan, G. Ochoa, L. Vanneschi, and S. Silva, "Fitness landscape analysis of convolutional neural network architectures for image classification," *Information Sciences*, vol. 609, pp. 711–726, 2022.

[168] G. Ochoa and N. Veerapen, “Neural architecture search: A visual analysis,” in *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I.* Springer, 2022, pp. 603–615.

[169] P. Mitchell, G. Ochoa, Y. Lavinas, and R. Chassagne, “Local optima networks for assisted seismic history matching problems,” in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar).* Springer, 2023, pp. 86–101.

[170] S. L. Thomson, G. Ochoa, and S. Verel, “Fractal dimension and perturbation strength: A local optima networks view,” in *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I.* Springer, 2022, pp. 562–574.

[171] D. Whitley and G. Ochoa, “Local optima organize into lattices under recombination: an example using the traveling salesman problem,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 757–765.

[172] I. Zelinka, D. Davendra, V. Snášel, R. Jašek, R. Šenkeřik, and Z. Oplatková, “Preliminary investigation on relations between complex networks and evolutionary algorithms dynamics,” in *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM).* IEEE, 2010, pp. 148–153.

[173] L. Skanderova, T. Fabian, and I. Zelinka, “Small-world hidden in differential evolution,” in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 3354–3361.

[174] P. Gajdo, P. Kromer, and I. Zelinka, “Network visualization of population dynamics in the differential evolution,” in *2015 IEEE Symposium Series on Computational Intelligence*, 2015, pp. 1522–1528.

[175] L. Taw, N. Gurrapadi, M. Macedo, M. Oliveira, D. Pinheiro, C. Bastos-Filho, and R. Menezes, “Characterizing the social interactions in the artificial bee colony algorithm,” in *2019 IEEE Congress on Evolutionary Computation (CEC).* IEEE, 2019, pp. 1243–1250.

[176] M. Oliveira, C. J. Bastos-Filho, and R. Menezes, “Towards a network-based approach to analyze particle swarm optimizers,” in *2014 IEEE Symposium on Swarm Intelligence.* IEEE, 2014, pp. 1–8.

[177] E. Hart and P. Ross, "Gavel - a new tool for genetic algorithm visualization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 335–348, 2001.

[178] T. Tusar and B. Filipic, "Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, p. 225–245, Apr 2015.

[179] J. E. Fieldsend, T. Chugh, R. Allmendinger, and K. Miettinen, "A feature rich distance-based many-objective visualisable test problem generator," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 541–549.

[180] T. D. Collins, "Applying software visualization technology to support the use of evolutionary algorithms," *Journal of Visual Languages and Computing*, vol. 14, no. 2, p. 123–150, Apr 2003.

[181] K. Michalak, "Low-dimensional euclidean embedding for visualization of search spaces in combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 232–246, 2019.

[182] H. Pohlheim, "Multidimensional scaling for evolutionary algorithms—visualization of the path through search space and solution space using sammon mapping," *Artificial Life*, vol. 12, no. 2, p. 203–209, Mar 2006.

[183] A. De Lorenzo, E. Medvet, T. Tušar, and A. Bartoli, "An analysis of dimensionality reduction techniques for visualizing evolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 1864–1872.

[184] G. Ochoa, K. M. Malan, and C. Blum, "Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics," *Applied Soft Computing*, vol. 109, p. 107492, 2021.

[185] Y. Lavinas, C. Aranha, and G. Ochoa, "Search trajectories networks of multiobjective evolutionary algorithms," in *Applications of Evolutionary Computation*, J. L. Jiménez Laredo, J. I. Hidalgo, and K. O. Babaagba, Eds. Cham: Springer International Publishing, 2022, pp. 223–238.

[186] Y. Lavinas, M. Ladeira, G. Ochoa, and C. Aranha, "Component-wise analysis of automatically designed multiobjective algorithms on constrained problems," in *Proceedings of the Genetic and Evolutionary Com-*

putation Conference, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 538–546.

[187] G. Ochoa, A. Liefooghe, Y. Lavinas, and C. Aranha, "Decision/objective space trajectory networks for multi-objective combinatorial optimisation," in *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*. Springer, 2023, pp. 211–226.

[188] T. Hu, G. Ochoa, and W. Banzhaf, "Phenotype search trajectory networks for linear genetic programming," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2023, pp. 52–67.

[189] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[190] A. McIntyre, M. Kallada, C. G. Miguel, and C. F. da Silva, "neat-python," https://github.com/CodeReclaimers/neat-python.

[191] ——, "neat-python," https://github.com/CodeReclaimers/neat-python.

[192] C. S. Wetherell and A. Shannon, "Tidy drawings of trees," *IEEE Transactions on Software Engineering*, vol. SE-5, pp. 514–520, 1979.

[193] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991.

[194] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006.

[195] K. O. Stanley and J. Lehman, *Why greatness cannot be planned : the myth of the objective*.

[196] R. Hinterding, "Representation, mutation and crossover issues in evolutionary computation," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 2, 2000, pp. 916–923 vol.2.

[197] D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," 1987.

[198] H. G. Rice, "Classes of recursively enumerable sets and their decision problems," *Transactions of the American Mathematical Society*, vol. 74, no. 2, pp. 358–366, 1953.

[199] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 184–192.

[200] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2018.

[201] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," *Information. Processing Letters*, vol. 31, pp. 7–15, 1989.

[202] "Open-endedness: The last grand challenge you've never heard of – O'Reilly."

[203] J. Lehman and K. O. Stanley, "Beyond open-endedness: Quantifying impressiveness," in *Artificial Life Conference Proceedings*. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2012, pp. 75–82.

[204] I. H. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, no. 6, Aug 2021.

[205] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 1, Mar 2021.

[206] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search: Framework and Applications*. Cham: Springer International Publishing, 2019, pp. 129–168.

[207] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 1997–2017, 2019.

[208] P. Ren, Y. Xiao, X. Chang, P.-y. Huang, Z. Li, X. Chen, and X. Wang, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, may 2021.

[209] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 550–570, 2023.

[210] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv preprint arXiv:1905.01392*, 2019.

[211] S. Sarti, "Neuroevolution trajectory networks: Revealing the past of incrementally neuroevolved cnns," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 41–42.

[212] A. D. Rosso, "Higgs: the beginning of the exploration. Étude du Higgs: ce n'est que le début," no. 47/2012, p. 3, 2012.

[213] R. P. Feynman, "Space-time approach to quantum electrodynamics," *Physical Review*, vol. 76, no. 6, p. 769–789, Sep 1949.

A

REFLECTIONS ON RESEARCH

The start of a long research project, such as a doctoral degree, is full of ambitions and often slight overoptimism. At the beginning, there is the expectation that shining a light towards darkness will reveal what you aimed to find, but that confidence soon disappears, leaving you to find that it takes more effort and resilience. This darkness is also filled with many obstacles that are not immediately visible.

Nevertheless, it is through this challenging journey, with the help of colleagues and extraordinary supervision, that you learn to grow and constantly improve. As a researcher, a scientist, a friend and someone who is willing to share the knowledge learned and land a hand to those needing it.

Throughout my Ph.D. research, I have learned some sound principles that have helped me to this date, which I would like to share with you.

- *Overcome the departure difficulty* — starting a 4 year-long project can feel like a daunting task, which might make you struggle to think how it should be started. Similarly to a white page, generating the *writer's block*. The best way is to not hesitate, and start the work; a path will reveal very quickly.

- *Swiftly move on* — recognise when a project or research avenue is not fruitful, or it is not producing the intended results. When something was overambitious, or if it is taking unfeasibly too long to accomplish. If so, dynamically move on to a secondary plan. If that does not work either, keep on moving, until things settle in the correct way.

- *Being proud* — Recognise when you have done a good job, be proud of the accomplishments, even the small ones. These are the fuel necessary to keep you going. The motivation to produce better and better research, with the necessary enthusiasm.

- *Breathing* — The difficult times are always right around the corner. Research can be stressful and it can make you spiral into confidence problems and imposter syndrome issues. Face it, chin up. Breathing is fundamental, might this be via physical activity, meditation or simply walking. It can get you unstuck. Go outside and breathe.

- *Enjoy it all!* — This is an excellent opportunity to make a significant contribution to knowledge, and after all the hurdles you might have had to overcome, you need to feel happy and enjoy the feeling of reaching this stage. Appreciate the collaborations, conferences, the riveting conversations, the friends and long-lasting relationships you make along the way. Like an artist at the end of a painting, stand back, admire your work and pat yourself on the back. Enjoy all those moments.