Neuro-Evolution Search Methodologies for Collective Self-Driving Vehicles

Chien-Lun (Allen) Huang

October 2019
Version: 1.02 (Final)

University of Cape Town

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Department of Computer Science

Computer Science

Neuro-Evolution Search Methodologies for Collective Self-Driving Vehicles

Chien-Lun (Allen) Huang

1. Reviewer Dr Geoff Nitschke
Department of Computer Science
University of Cape Town

Supervisors Dr Geoff Nitschke

October 2019

Chien-Lun (Allen) Huang
Neuro-Evolution Search Methodologies for Collective Self-Driving Vehicles
Computer Science, October 2019
University of Cape Town
Department of Computer Science
18 University Avenue
Rondebosch
Cape Town
7700

Abstract

Recently there has been an increasing amount of research into autonomous vehicles for real-world driving. Much progress has been made in the past decade with many automotive manufacturers demonstrating real-world prototypes.

Current predictions indicate that roads designed exclusively for autonomous vehicles will be constructed and thus this thesis explores the use of methods to automatically produce controllers for autonomous vehicles that must navigate with each other on these roads.

Neuro-Evolution, a method that combines evolutionary algorithms with neural networks, has shown to be effective in reinforcement-learning, multi-agent tasks such as maze navigation, biped locomotion, autonomous racing vehicles and fin-less rocket control.

Hence, a neuro-evolution method is selected and investigated for the controller evolution of collective autonomous vehicles in homogeneous teams.

The impact of objective and non-objective search (and a combination of both, a hybrid method) for controller evolution is comparatively evaluated for robustness on a range of driving tasks and collection sizes.

Results indicate that the objective search was able to generalise the best on unseen task environments compared to all other methods and the hybrid approach was able to yield desired task performance on evolution far earlier than both approaches but was unable to generalise as effectively over new environments.

Acknowledgement

This thesis has been a long journey for me as I transitioned through various stages of my life. It has taken far longer than I had originally planned and I am grateful to everyone who believed in me and supported me along the way.

First, thank you to all who have funded my Masters. This includes the *National Research Foundation* (NRF) and the *Department of Science and Innovation* (DSI) in collaboration with the *Council for Scientific and Industrial Research* (CSIR) for awarding me bursaries. My current employer, *Allan Gray Proprietary Limited* also provided additional funding.

I would also like to thank my supervisor, *Geoff* for providing insight, inspriation and for being patient as I slowly progressed with this thesis.

Finally, I am grateful to my family for always supporting me in all my endeavours and my girlfriend, *Suri* for always being there and providing constant motivation and encouragement. I would also like to thank all my friends who helped proof-read this thesis to ensure that it is of the highest quality.

Without you all, this research would not have been achievable.

Thank you.

Contents

1 Introduction

> *To be clear, Tesla is strongly in favour of people being allowed to drive their cars and always will be. However, when self-driving cars become safer than human-driven cars, the public may outlaw the latter. Hopefully not.*
>
> **— Elon Musk**
> (Entrepreneur, inventor and Tesla CEO)

Recently, autonomous vehicles have been of increased interest for various automotive companies. Future intelligent transport systems are envisaged to have thousands of autonomous vehicles which detect objects, avoid collisions and predict accidents, all whilst collectively traversing optimal paths through highways and road networks. Autonomous vehicles from different manufacturers will have to negotiate and navigate with each other and thus employing a distributed control method which does not rely on external control systems, leaving behavioural autonomy to individual vehicles (Martinoli et al., 2002) will be required.

For a distributed system of autonomous vehicles to operate effectively, controllers (behaviours) of individual vehicles play a critical role in the safe collective flow of traffic. Moreover, current engineering design methods are not appropriate for the design of autonomous vehicles that must elicit a distributed collective behaviour (Zhang et al., 2003) for the safe and constant flow of traffic at given speeds for a vast range of roads and highways.

Therefore, using traditional engineering methods, traffic systems can be viewed as complex systems where it is difficult to determine what the controller for each vehicles should be such that an optimal collective behaviour is synthesized.

This research thus investigates the evolution (adaptation) of controllers for autonomous vehicles in a simulated environment using neuro-evolution directed by objective (Stanley and Miikkulainen, 2002), non-objective search (Lehman and Stanley, 2011) and a combination of both (hybrid) to automate the autonomous vehicle controller design process.

1.1 Motivation and Problem Statement

Distributed autonomous vehicle control is a complex process, especially when collective behaviour needs to be synthesized. Moreover, current driving systems rely on vision, rather than more advanced technologies such as smart infrastructure or vehicle-to-vehicle communication. Current self-driving cars attempt to replicate human actions but at a more efficient level, thus still relying on traditional vision-based driving requirements and road rules.

In the future, it is predicted that all vehicles on roads will be autonomous. If this situation arises, the need for road rules that govern the way humans drive will no longer be appropriate and thus vehicles could rather travel in a more optimal fashion: simultaneously avoiding obstacles (other autonomous vehicles, road obstacles, pedestrians) and keeping within road barriers without the need for lanes or intersections that require stopping.

For this research we assume all vehicles will have fully-functioning sensor systems to properly detect other vehicles and obstacles around it (for example, LIDAR sensors (Levinson et al., 2011)) along with a vehicle tracking system (for example, GPS (Space-Based Positioning and Timing, 2017)) so that vehicles are aware of their positions relative to its destination.

Autonomous vehicles will be tasked with navigating a simulated environment with various static and dynamic obstacles. These autonomous vehicles will need to collectively navigate whilst avoiding collisions with each other and other obstacles whilst minimising travel time by following the optimal path at high speeds and thus utilising energy effectively.

Currently, there is no distributed control system for collective fully-autonomous vehicles that must navigate roads whilst accounting for other autonomous vehicles, obstacles and unpredictable events such as pedestrians or animals crossing. This research attempts to address this gap by presenting neuro-evolution methods for this application.

1.2 Methods

The main objective of this research is to automate the production of autonomous vehicle controllers that operate in a distributed fashion on any given environment. This means that controllers will be generalisable to unseen road networks that differ

from the training environment and be robust to different group sizes of autonomous vehicles.

Neuro-Evolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002) is used to adapt the vehicle controllers. NEAT is an approach to adapt artificial neural networks by evolving network connection weights and the network's structure (topology).

We investigate three methods: a traditional objective based approach (using a fitness function): NEAT Objective Controller Evolution (OCE), a non-objective based approach: NEAT Novelty Controller Evolution (NCE) (based on novelty search (NS) by (Lehman and Stanley, 2011)), and a hybrid (HCE) approach where the fitness function from OCE is combined with the novelty score from NCE.

In our experiments, vehicles act in homogeneous teams whereby morphology (shape, type, sensor configuration) and controller are the same for individual vehicles during evolution process. Homogeneous (clones) teams allow for many instances of a controller to be tested simultaneously, and is thus more efficient to adapt. Much of the literature uses homogeneous teams (Waibel et al., 2009) as they are easier to use (Trianni et al., 2006), scale more easily and are more robust to failures of individuals (Bryant and Miikkulainen, 2003) within teams when compared with heterogeneous teams (Floreano and Mattiussi, 2008). To adapt heterogeneous teams, each variant controller would compound the overall evaluation if it were to have been evolved with the same rigour as in the homogeneous team.

1.3 Contributions

NE has demonstrated the ability to produce ANN controllers that yield behaviours successful at a wide variety of complex control tasks such as maze navigation, biped locomotion (Lehman and Stanley, 2011) and pole balancing (Gomez et al., 2006), along with a variety of autonomous driving research (Togelius and Lucas, 2006; Ebner and Tiede, 2009; Drchal and Koutník, 2009).

A non-objective search approach for NE, NS has also been demonstrated to outperform (in terms of search speed and ability to avoid getting trapped in local optima) traditional objective based methods in certain domains (Lehman and Stanley, 2011).

Furthermore, the combination of the objective and non-objective methods (a hybrid), has demonstrated the ability to outperform pure approaches (Huang et al., 2015).

In this research, we assess the efficacy of traditional objective, novelty search and a hybrid methods on the collective autonomous driving task.

Prior research on autonomous driving using NE have focused on adapting controllers to navigate a single vehicle through a track using either objective or non-objective methods, rarely combining the two or adapting controllers to yield collective driving behaviours.

Given past research demonstrating the capability of hybrid approaches outperforming traditional (pure) methods, we hypothesize that the hybrid approach will adapt controllers that will generalise more effectively across various unseen environments when compared with the pure-objective and pure-novelty methods in this collective self-driving task.

The ability to successfully synthesize collective vehicle control behaviours could aid in designing future transportation systems where autonomous vehicle manufacturers develop safe and efficient autonomous fleets that do not rely on costly centralised control systems.

1.4 Thesis Structure

Chapter 2

This chapter provides the background of this research and introduces neuro-evolution, evolutionary search methods and autonomous vehicles.

Chapter 3

This chapter gives an overview of the specific implementations were employed for this research including details of the simulator, implementation and parameters for Neuro-Evolution of Augmenting Topologies (NEAT), the evaluation functions defined for each evolutionary search method and further details on our implementations for novelty search (NS) and hybrid approaches.

Chapter 4

This chapter describes the experiments, simulated vehicles and environments along with all permutations of experiments undertaken to assess performance between each search method. Results are then presented with graphs and statistical tests.

Chapter 5

The penultimate chapter presents a detailed analysis of the results and relates them back to the initial hypothesis.

Chapter 6

Finally, this chapter discusses the results and describes some limitations with this work and future research.

2 Background

This chapter provides background on methods used in our research and a literature review of existing research on autonomous vehicle controller design. It is divided into three sections.

The first section introduces Neuro-Evolution (NE) and explores the underlying technologies: artificial-neural networks (ANN) and evolutionary algorithms (EA). Neuro-Evolution of Augmenting Topologies (NEAT) is discussed in detail as it is the method used to adapt vehicle controllers in this research. In section 2, evolutionary search methods are discussed with focus on objective, non-objective and hybrid methods to guide NE. The non-objective method, Novelty Search (NS) is elaborated in more detail as it is the focus of this research.

In the final section, research in controller design for autonomous vehicles is surveyed and the current state-of-the-art self-driving technology in production and prototype vehicles outlined.

2.1 Neuro-Evolution

Neuro-Evolution (NE) is the evolution of Artificial Neural Networks using Evolutionary Algorithms. NE thus combines the power of two biologically inspired methods: the brain and the evolutionary process that derived the brain over generations. NE presents an alternative method to traditional reinforcement learning (RL) problems as solutions are not modified during evaluations (ontogenetic) but rather, through recombination of individuals in a population. This population-based or *phylogenetic* learning gradually moves the population towards the solution where an individual that exhibits a desired behaviour on a given task (highest fitness) is found.

2.1.1 Artificial Neural Networks

Artificial Neural Networks (ANN) is a computational model based on the biological neural networks found in the natural brain. ANNs are powerful problem solvers as they are universal function approximators, are generalisable and can retain memory (through recurrent networks) (Gomez, 2003). Similar to how a brain's neural

network is comprised of a vast array of neurons and their *connections* (axons and dendrites), an ANN is created by combining multiple artificial neurons.

The artificial neuron (depicted in the inset in figure 2.1) is a referred to as a *node* and it receives inputs via its input connections and produces a single output based on the input connection's weights and the node's activation function. The input values from the input connections are multiplied by the connection weights and a bias value is added to create a net input value using the following equation:

$$z = \sum_i w_{ij} O_i + \theta_j \tag{2.1}$$

where w_{ij} is the weight between current node j and incoming node i, O_i is the output value of node i and θ_j is the bias.

The net input, z is then passed to the activation function. Figure 2.2 depicts four commonly used activation functions. The output from this node thus depends on its inputs and the type of activation function used (for example, if a sigmoid function was selected, the output would be: $\sigma(z)$).

Multiple neurons are connected together to form an ANN. In the application of such networks, the structure, connection weights or both are modified and tuned to enable it to *learn* and create desirable outputs based on given inputs. Figure 2.1 depicts a fully-connected feed-forward neural network with a total of 10 nodes (4 input nodes, 5 hidden substrate nodes and one output node). Layers are used to describe where nodes reside in the network. Input nodes are in the *input layer* and these nodes have no incoming connections from other nodes. Output nodes are in the output layer (they terminate and have no outgoing connections into other nodes) and nodes that have both incoming and outgoing connections to and from other nodes are referred to hidden nodes in the hidden layer. There can be any number of hidden layers but only one input and one output layer. The network structure (number of nodes and connections between them) are referred to as the *network topology*.

Each input node can receive some numerical *feature* or signal of the task or environment. In a supervised learning task, an example in image processing would map each input node to a cluster of pixels on an image. Similarly, in reinforcement learning such as control systems, inputs would be signals from sensors that receive information about its environment. Outputs from these networks would either be classification of an image or some action that is taken by the control system to influence the environment.

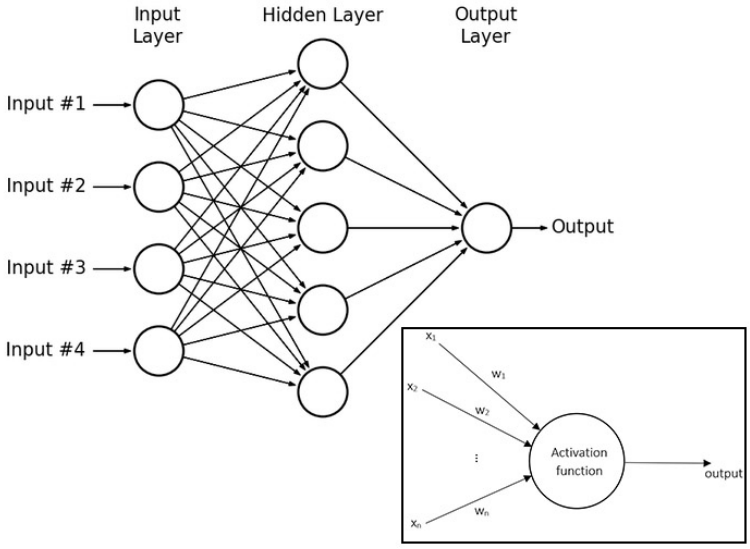


Fig. 2.1: An example fully-connected feedforward **artificial neural network**. The inset depicts an **aritficial neuron** or *node* from the hidden layer consisting of inputs x_1 through x_n. A connection weighting w_1 through w_n for each input connection, an activation function and an output. Inputs are multiplied by their weights, combined and passed through the *activation function* which then creates an output signal. Based on diagram from Yegnanarayana (2009).

ANNs have been applied in a wide-array of supervised, unsupervised and reinforcement learning tasks with examples such as image recognition, machine translation and control systems such as autonomous vehicles, respectively (Ciregan et al., 2012; Bahdanau et al., 2014; Pomerleau, 1991).

Different ANN representations are used depending on the learning type and task. Examples include: the recurrent neural network or the feed-forward neural network both of which can be partially or fully-connected. Figure 2.1 depicts the most straight-forward representation, a fully-connected feed-forward neural networks which has no cycles in the network.

Various techniques exist in adapting an ANN to solve problems. Traditional methods focus on tuning parameters of a single ANN repeatedly until some criteria is satisfied. These developmental processes are referred to as being *ontogenetic*. The most basic form of an ontogenetic method is the *Gradient Descent Algorithm* (Baldi, 1995). Evolutionary methods adapts the whole population of ANNs towards a solution through reproductive and mutative processes, instead of adapting parameters of individual ANNs. This *phylogenetic* method recombines and mutates individual ANNs to produce desirable *offspring* which eventually moves the population towards the solution where an individual in that population yields the desired behaviour.

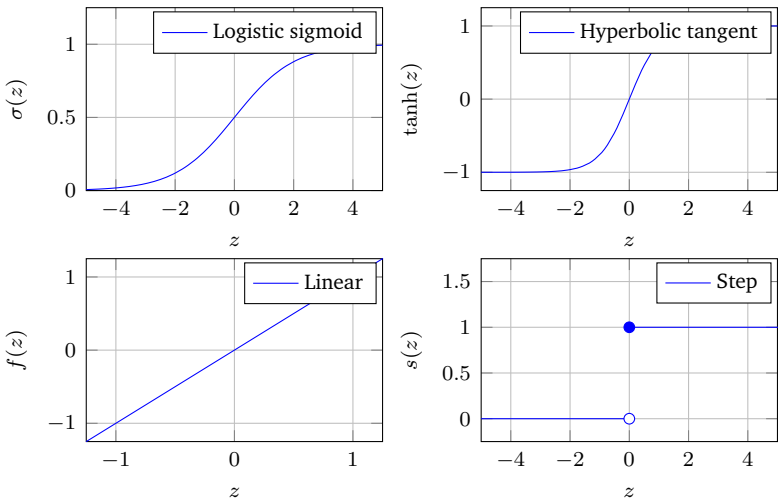


Fig. 2.2: **Activation Functions** commonly used include the logistic sigmoid $\sigma(z)$ and the hyperbolic tangent $\tanh(z)$. Other activation functions include the linear function $f(z)$ and the step function $s(z)$. (Krishna, 2018)

In this research, evolutionary algorithms are used to adapt ANNs. The next subsection thus describes this phylogenetic method and its process on ANN adaptation in more detail.

2.1.2 Evolutionary Algorithms

Evolutionary Computation (EC) and *Evolutionary Algorithms* (EA) are optimisation methods inspired by the process of natural evolution. The power of evolution in nature is evident in the diverse species that make up our world and how they survive in its own niche. The fundamental process of natural evolution is trial-and-error and it has proved to be a powerful method and thus is used as the main metaphor that EC uses for problem solving in computing (Eiben and Smith, 2003).

EA methods maintain a population of candidate solutions encoded in a chromosome. The chromosome represents properties of the solution depending on the encoding scheme. Depending the the specific EA and task, solution representations vary:

- Genetic Algorithm (GA; (Goldberg and Holland, 1988)) methods represent solutions with a vector of bits or real-valued numbers.

- Evolutionary Programming (EP; (Fogel and Fogel, 1995)) uses *finite state machines* (FSM) and its chromosomes encode the various components in a transition table.

- Genetic Programming (GP; (Koza, 1994)) uses tree-structures and thus chromosomes encode each node and branch.

- Neuro-Evolution (NE; (Wieland, 1991)) evolves ANNs and thus chromosomes encode the network structure and connection weights in some way and is discussed in further detail in section 2.1.3.

For example, using a GA to solve the *Travelling Salesman Problem* (Applegate et al., 2006) (a travelling salesman visiting multiple cities has to minimise travel distance. The task is to find the shortest path), an appropriate encoding of this problem would be to store each city's identifier as a gene in the solution's genotype. Each subsequent gene would store the *next* city the salesman should travel to and thus this representation maps directly to the problem whereby cities and order of travel is encoded (Grefenstette et al., 1985).

The EA process is a guided-random (*stochastic*) search which proceeds as follows:

1. A population of candidate solutions are initialised - either randomly or using a heuristic. Invalid solutions are removed or prevented from being created by some rule.

2. Each individual is evaluated against a criteria. A measure of fitness is assigned to the individuals.

3. Individuals are selected by a selection criteria (for example, fitness proportionate, elitism) which are then recombined to produce offspring (Eiben and Smith, 2003). Invalid or damaged offspring are either discarded and recreated or prevented from being produced.

4. Offspring are mutated and then evaluated where fitness is assigned to each offspring.

5. Using a replacement criteria (for example: fitness based, elitism), offspring replace parents in the population.

6. Steps 2 - 5 are repeated until a stopping condition (solution found or maximum number of evaluations) is met.

Selection pressure in steps 3 (recombining solutions that perform well) and 5 (replacing weak parents with fit offspring) forces newer generations to have high fitness and thus guides the search in finding good solutions in a reasonable time.

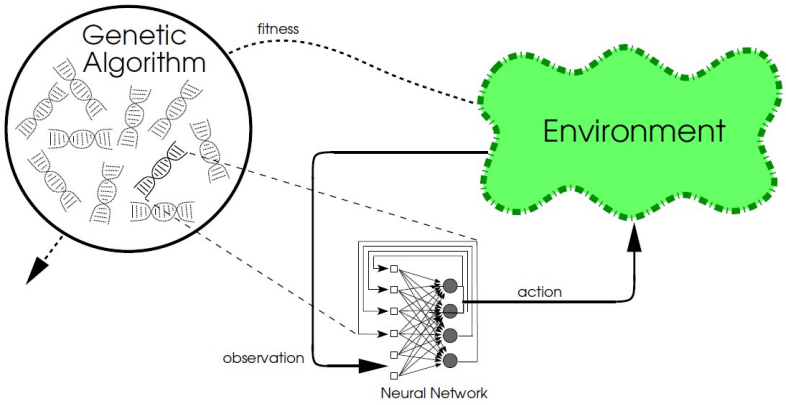


Fig. 2.3: The feedback nature of neuro-evolution (Gomez et al., 2008). Genotypes encode an individual neural network's structure. The neural networks are then evaluated in an environment which provides feedback on a genotype's fitness. Following the evolutionary process described in section 2.1.2, networks adapt to the environment.

The use of a population of candidate solutions gives EAs the advantage in that it can sample many points in the search space simultaneously which makes them less susceptible to local optima than ontogenetic methods. It also means that they are capable of climbing up multiple peaks in high dimensional search spaces, increasing the chances of reaching the global optima. These attributes make EAs suitable for solving complex problems (Gomez, 2003).

2.1.3 Evolving Neural Networks

EAs can be used to adapt different features of an ANN including connection weights, activation functions and topology or a combination of these. The most common way of using EAs for evolving ANNs is by evolving connection weights, as this is the most straightforward method and can also proxy topology modification (as weights set to 0 disconnects parts of a network) (Gomez, 2003). Evolving only a networks connection weights limits the complexity of solutions as networks are limited to a maximum predefined number of nodes and connections. Effective network topologies have to be predetermined and designed by a human with in-depth domain knowledge.

A genotype representing an ANN's properties can be encoded using a *direct encoding* scheme wherein each node, their connections and weights are represented (Wieland, 1991; Moriarty and Mikkulainen, 1996; Stanley and Miikkulainen, 2002) or via an *indirect encoding* scheme where rules for generating or developing a network is encoded instead (Gruau, 1993; Clune et al., 2011; Stanley et al., 2009). Directly encoded genotypes map to the ANN phenotype which makes it easier to adapt and evaluate. However, upper bounds of the network size is limited as large network

representations need to be stored in memory. Indirectly encoded genotypes can thus be more compact as only the *plans* for generating a network is stored, allowing for much larger ANN phenotypes. Figure 2.4 shows the indirect scheme used in Cellular Encoding by Gruau (1993) and direct encoding used in NEAT (Stanley and Miikkulainen, 2002).

NE searches through the behaviour space for a network that performs well at a given task (Stanley and Miikkulainen, 2002). Figure 2.3 shows how a NE algorithm proceeds. Each genotype is transformed into an artificial neural network (phenotype) and evaluated on the task. In a reinforcement learning task, the neural network receives input from the environment and produces an output signal that affects the environment. Thereafter, a fitness is assigned to the network according to its performance on the problem. Therefore, networks that perform well according to the objective (and thus have high fitness) are recombined to generate new networks in the following generations (see section 2.1.2) (Gomez et al., 2008).

By combining the capabilities of ANNs (universal function approximation, generalisability, memory) with the efficient EA search method, NE is a powerful tool for solving complex, noisy and partially observable control tasks (Gomez, 2003), such as autonomous vehicle control.

Below is a survey of relevant NE methods for RL tasks:

- Conventional Neuro-Evolution (CNE; (Wieland, 1991)) uses a single population and each chromosome represents a complete neural network. Networks are encoded with binary numbers and fully recurrent neural networks are used. Only weight values are evolved in this case.

- Symbolic, Adaptive Neuro-Evolution (SANE; (Moriarty and Mikkulainen, 1996)) evolved two separate populations simultaneously in a cooperative co-evolutionary fashion. A population of neurons and a population of *network blueprints* that form complete networks are evolved and combined randomly at each generation. Neurons and blueprints that yield high fitness networks are rewarded independently in their own populations for further evolution. Topologies are fixed in SANE and not adapted by evolution.

- Enforced SubPopulations (ESP; (Gomez and Miikkulainen, 1997)) is an extension of SANE and also uses cooperative co-evolution but it uses split populations of neurons instead of network blueprints. One neuron from each sub-population is used for each hidden unit in the network. This allows neurons to specialise within their own sub-population and thus optimising the performance of each part of the network simultaneously.

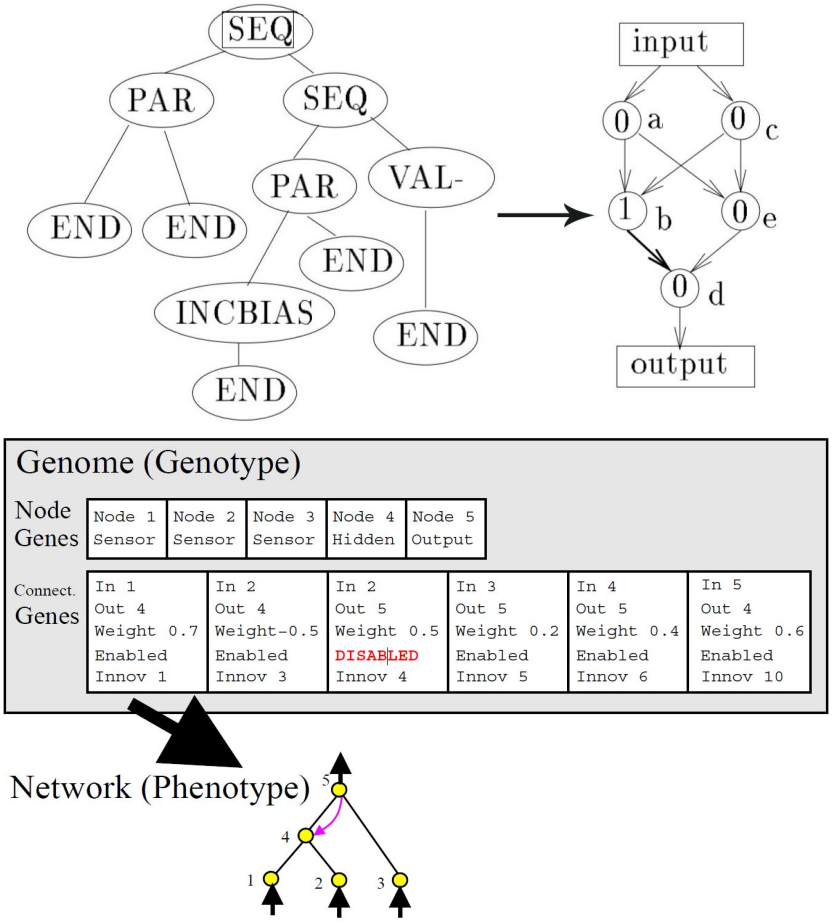


Fig. 2.4: **TOP:** Indirect encoding used in Cellular Encoding (Gruau, 1993). Grammar trees are encoded in the genotype and are used to generate network structure. **Bottom:** Direct encoding used in NEAT (Stanley and Miikkulainen, 2002). Genotypes explicity encode each node and connection for the network structure.

- GeNeralized Acquisition of Recurrent Links (GNARL; (Angeline et al., 1994)) is a Topology and Weight Evolving Artificial Neural Networks (TWEANN) that specifically adapts recurrent networks. GNARL only performs mutation on genotypes as authors question the efficacy of recombination.

- Neuro-Evolution of Augmenting Topologies (NEAT; (Stanley and Miikkulainen, 2002)) is also a TWEANN. It uses a direct-encoding scheme and starts off with minimal networks, gradually increasing the network complexity by either adding more nodes or connections. NEAT employs both mutation and recombination and solves many issues plaguing topology adaptation in past TWEANN algorithms. NEAT is described in further detail in the next section, as it is the focus of this research.

2.1.4 Neuro-Evolution of Augmenting Topologies (NEAT)

Neuro-Evolution of Augmenting Topologies (NEAT) is a Topology and Weight Evolving Artificial Neural Networks (TWEANNs) by Stanley and Miikkulainen (2002). It is a *complexification* algorithm in that it starts out with very simple, minimal neural networks and progressively increases the number of neurons and connections between them. This process is analogical to biological evolution where species increase in complexity over evolutionary generations.

The advantage of using a complexification algorithm, such as NEAT for evolving ANNs, is due to the progressive difficulty of training increasingly complex (many nodes, layers and connections between them) networks. Furthermore, a simple network might achieve similar results to a complex network, but with the advantage of being easier to evolve without changing their behaviour drastically and thus avoiding large networks that are slower to adapt.

Representation

The NEAT algorithm uses a direct genetic encoding that is designed to allow corresponding genes to be easily lined up during recombination. A genotype[1] consists of multiple genes (see figure 2.4 Bottom). The authors chose direct over indirect encoding as it does not restrict the phenotype networks and they demonstrate experimentally that an indirect encoded algorithm (Cellular Encoding (Gruau, 1993)) is not necessarily more efficient than a direct encoded algorithm (Stanley and Miikkulainen, 2002). Furthermore, upper-bound network size limitations for direct

[1] A genotype's gene in NEAT specifies one connection between nodes whilst the whole genotype represents the complete ANN.

encoding scheme does not affect this method as NEAT is biased towards solutions that have minimal networks.

Each gene has historical markings known as *innovation numbers* which denote when the gene was added. This unique feature in NEAT solves two problems experienced by TWEANN algorithms.

A common issue with TWEANNs is that structural changes (additions of connections or nodes) usually cause a drop in fitness as the addition is unlikely to bring utility as soon as it they are introduced and a few generations are required before they are optimized. Such *innovations* to the structure are often discarded by the EA as they are ranked lower in terms of fitness.

To address this and ensure topological diversity, many TWEANNs initialise the population with a random collection of topologies. Solutions in these random populations may produce infeasible networks where nodes have no connections from its input and outputs and require additional effort to clean up. These random populations also take longer to optimize as they have larger number of parameters (often many unnecessary nodes and connections to achieve behaviours that can be elicited with smaller networks).

Instead of initialising a random population of large solutions, NEAT starts with minimal networks and protect innovations by using its *innovation numbers* to *niche* or speciate genotypes (group individuals with genetic similarity) to prevent them from being evaluated with the whole population. Fitness sharing in a species of genotypes is employed by NEAT and giving new genes some time to adapt, protecting innovative genes from being discarded. The ability to protect innovations also allows NEAT to minimise network sizes *throughout evolution* without having to incorporate a penalty on network size in the fitness function as in Zhang and Muhlenbein (1993), which may have undesirable effects of on the search and requires further tuning as it is unclear what the penalty should be in relation to task performance.

Since NEAT uses a direct encoding scheme, solutions that are genetically similar are topologically similar and thus niches are groups of networks that are similar in structure (see 2.1.4).

The second issue in TWEANNs is the *Competing Conventions Problem* where there are many genotypic ways to express a solution. When genotypes that are functionally equivalent crossover, they are likely to produce damaged offspring. Innovation numbers allow genes that are equivalent in two different genotypes to *line-up* solving the problem.

Speciation

The innovation numbers allow us to line-up genes in a genotype that are similar and determine disjoint and excess genes. The number of disjoint and excess genes between solutions (see figure 2.7) can thus be used as a measure of compatibility distance between a pair of genotypes:

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

where E and D is the number of *excess* and *disjoint* genes and $\overline{W}$, the average weight differences of matching genes (including disabled genes). c_1, c_2 and c_3 are constants that are adjusted to weight importance of each of the three factors and N is the number of genes in the larger genotype, used for normalisation. The distance measure δ allows speciation of genotypes using a compatibility threshold δ_t.

The the end of a generation, each genotype (g) in the population is placed into species. Compatibility is determined for each species s sequentially (via a maintained list of species, S) by using a random representative genotype (s_g). The genotype is placed in first the specie where genotype's $\delta(g, s_g) \leq \delta_t$. The average compatibility of g compared with all the genotypes in s, but practically, using a single genotype (s_g) is sufficient and faster (constant time). If g is not compatible with any species ($\forall s \in S, \delta(g, s_g) > \delta_t$), a new species is created and g becomes a member of the new s.

Explicit fitness sharing is used within species, this means that all individuals in the species share the niche's fitness. This fitness of an individual, i is calculated according to the distance δ from every other individual j.

Competing Conventions

The competing conventions problem also known as the *permutations problem* is when there is more than one way to express a neural network solution. When genotypes representing the same solution do not have the same encoding, crossover is likely to produce damaged offspring. Figure 2.5 depicts the problem for a solution with three hidden nodes in two permutations. The hidden nodes can be configured in $3! = 6$ different permutations (in general, for n hidden nodes in a network, there are $n!$ functionally equivalent permutations) and when one of these permutations recombines with another, critical information in the network is lost. For example,

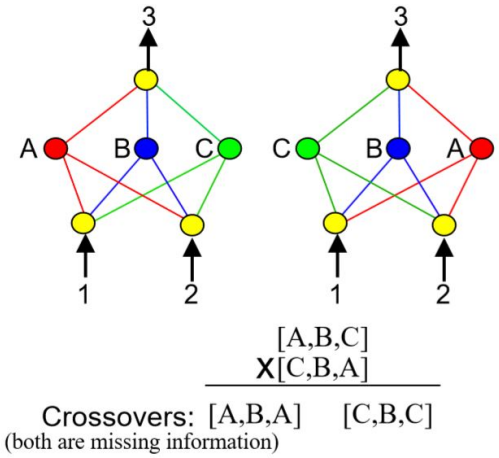


Fig. 2.5: Competing conventions problem: Two possible permutations (of 6) of a networks that compute a specific function but with hidden nodes appearing in different orders. Below the networks are two single-point crossover children between these two permutations, both missing one of the three components present in both parents (Stanley and Miikkulainen, 2002).

recombining *[A, B, C]* with *[C, B, A]* can result in *[A, B, A]*, a solution that has lost one-third of the information that both parents had: *C*.

NEAT takes inspiration from nature's gene alignment in sexual reproduction so that the correct genes are crossed with their counterparts in another genotype. In nature, a special protein, RecA ensures *homology* between two genes (genes are homologous if they are alleles of the same trait) before crossover (Radding, 1982). Homology can be found in the *historical origin* (via innovation numbers) of two genes if they originated at the same time. This allows NEAT to line these genes up for crossover without losing functional information in the resultant child (see figure 2.7).

Mutation and Recombination

Mutation in NEAT can occur in two different ways in NEAT: connection weights mutation and structural mutation and as with any other NE method, mutation of connection weights either occur or do not occur on a gene in each generation.

Structural mutation in NEAT occurs by either adding new connections or adding new nodes to the genotypes. When adding a new connection, a connection is made between two previously unconnected nodes. A gene representing the new connection is added to the end of the sequence and assigned the next incremental *innovation number* (see figure 2.6).

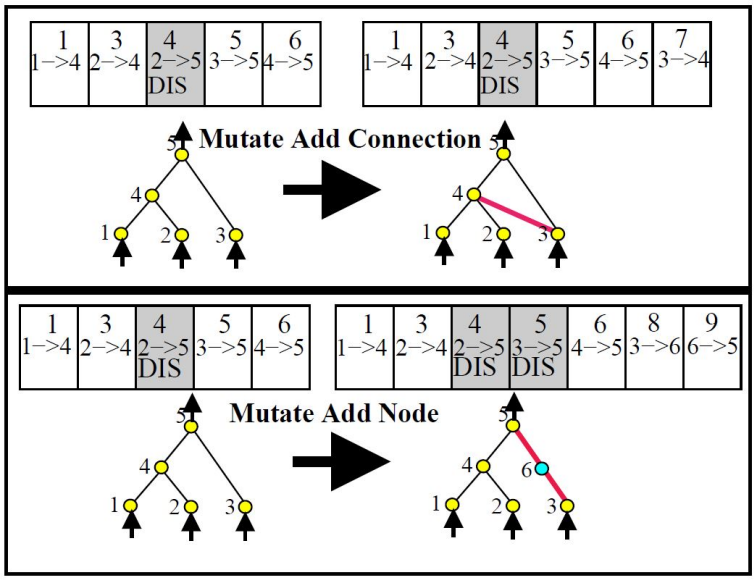


Fig. 2.6: Mutations in NEAT: The top number in each gene represents the innovation number, below that the connection between two nodes is represented. A gene can either be enabled or disabled (which are denoted by *DIS* and shaded in gray) (Stanley and Miikkulainen, 2002).

When adding a new node, an existing connection is split and the new node is placed where the connection used to be. The old connection is disabled and two new connections are added to the genotype. The new connection leading into the new node receives a weight of 1, and the new connection leading out receives the same weight as the old connection (see node 6 in figure 2.6).

Both of these mutations (adding a new node or adding a new connection) expand the size of the genotype by adding new genes (Stanley and Miikkulainen, 2002).

During mutation in a generation, there is a chance that an identical mutation occurs. To prevent an explosion of innovation numbers (added to each new gene), a list of innovations for the current generation is maintained. When the same mutation occurs again within a generation, the new gene is assigned the same innovation number as before.

When recombination occurs in NEAT, parent genomes are recombined by matching up their genes' innovation numbers. Genes with matching innovation numbers are inherited in the offspring randomly from either parent. Disjoint genes (those that do not match in the middle) and excess genes (those that do not match in the ends) are inherited from the parent with higher fitness (see figure 2.7). In the case that fitness of both parents are the same, random inheritance of these genes also occurs (Stanley and Miikkulainen, 2002).

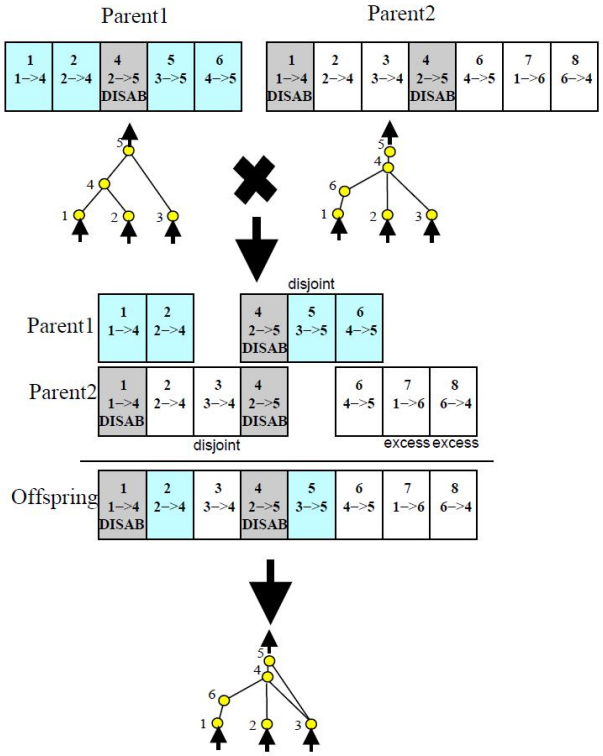


Fig. 2.7: **Recombination:** Genes in parent genotypes are matched up using the innovation numbers to ensure offspring produced retain functional aspects of both parents (Stanley and Miikkulainen, 2002).

2.1.5 Why NEAT?

Experiment results on various tests by Stanley and Miikkulainen (2002) demonstrate that NEAT outperforms other NE (TWEANN and WEANN) methods: CE, SANE and ESP. Results from the pole balancing task, a useful proxy for control systems showed NEAT not only adapted controllers in far less evaluations, but also adapted desired control behaviours with smaller network configurations.

These results along with the long list of related research (Cardamone et al., 2010; Willigen et al., 2013; Jallov, 2014; Watson and Nitschke, 2015; Parker and Nitschke, 2017) using NEAT to adapt controllers and its suitability for Novelty Search (section 2.2.2) used in our research provides evidence of NEAT's suitability for our purpose of adapting controllers collective autonomous vehicle control and using it as a foundation to assess impacts of objective and non-objective search.

2.2 Evolutionary Search Methods

Different search methods are be used to guide the evolutionary process. In this section, objective and non-objective search methods are described.

2.2.1 Objective and Non-Objective search

EAs require some criteria to evaluate individuals candidate solutions during the search process, as described in steps 2 - 4 in section 2.1.2 above.

This is often described as the objective or fitness function. The reason for this nomenclature is due to a solution's success measurement at a given task or *objective*. It measures how *well* a solution performs at a given task based on some *function* predetermined by the researcher. Furthermore, the choice of function to use can vary an EAs performance significantly.

Non-objective search methods, on the other hand, do not require researchers to craft fitness functions to assess a genotype's performance. Many researchers (Gould (1996), Miconi (2008) and Sigmund (1995)) have argued that fitness functions, which induce selection pressure (pressure to adapt in a certain way) actually restricts the search and opposes innovation.

In many problem domains, it is also extremely difficult to craft effective fitness functions, as it requires an *a priori* understanding of the fitness landscapes and stepping stones to the objective (Woolley and Stanley, 2011). An oversight made by the researcher when designing a fitness function could cause the search to prematurely converge or be trapped in local optima.

In a non objective method, stepping stones that would have been thrown away by an objective method because they appear to be far away from the objective, will be preserved. These stepping stones may lead to the ultimate objectives and prevent solutions from getting trapped by deception (Lehman and Stanley, 2011)[2].

Various non objective methods exist such as *novelty search* by Lehman and Stanley (2011) which explores the behaviour space instead of the problem space, *curiosity-driven exploration* by Pathak et al. (2017) which uses an *intrinsic* reward mechanism (prediction error as a proxy for curiosity) instead of *extrinsic* or *objective* rewards.

[2]Local optima that appear to be close to the objective, but prevents solutions from ever reaching the ultimate goal.

Despite the advantage of not having to craft fitness functions, non-objective search still requires the researcher to craft and use some other metric to determine solution *quality* in a non-objective manner. One example would be the **novelty metric** in a non-objective method *novelty search* which is used to define how to measure *solution novelty* or *behavioural distance*.

2.2.2 Novelty Search

Novelty search is a non objective neuro-evolution algorithm, proposed by Lehman and Stanley (2011) where solutions are rewarded by a novelty metric based on how significantly different or *novel* their behaviours (phenotypes) are in respect to previous solutions.

Novelty search presents a new way of traversing the search space. It operates on the premise that novel solutions provide the stepping stones (ones that objective-based methods would have discarded as they appear to be too far away from the objective) to the final objective. This way, novelty search ensures that solutions are not trapped or deceived into a local optima, which plagues objective-based methods.

Novelty search has been experimentally demonstrated by Lehman and Stanley (2011) to vastly outperform objective search in the maze navigation (see figure 2.8) and biped locomotion domains.

The novelty search algorithm encourages novelty and diversity in its genotypes via the maintenance of a novelty archive. This is achieved by keeping a permanent archive of past genotypes whose behaviours were highly novel when they originated. Newly produced genotypes are measured against its own population and the archive. The aim is to characterize how far away in the behaviour space (rather than the solution space) new genotypes are from the rest of the population and its predecessors. Thus, a good novelty metric should compute the sparseness at any point in the behaviour space.

The Novelty Search NEAT algorithm has been adapted by Lehman and Stanley (2011). The adaptation is straightforward as all that was required was to replace the fitness function with a novelty metric and an addition of a novelty archive because the underlying algorithm of NEAT ensured that the ANNs became more complex as solutions were being explored. Implicitly, this means that once simple ANNs have been explored, more complex ones will create novel behaviours (that were not producible by simpler ANNs), ensuring that the search is not random and revisiting already explored areas.

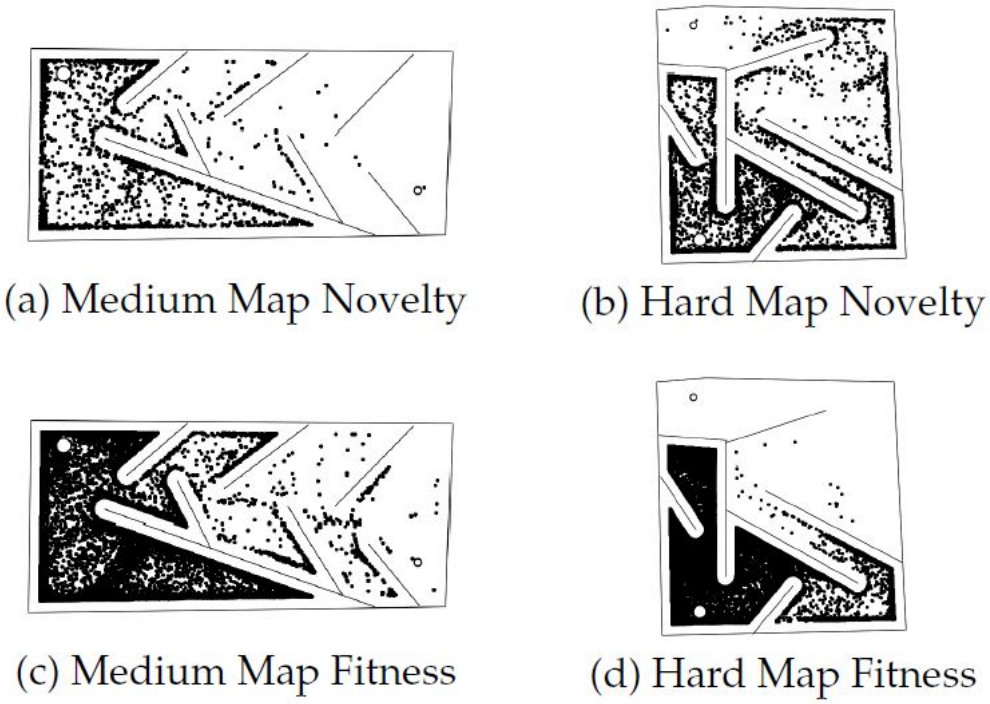


Fig. 2.8: **Novelty Search Maze Navigation Resuts:** starting positions in the medium (a, c) and hard (b, d) maps are located at the top left and bottom left, respectively. The objective or goal for the medium and hard maps are located at the bottom right and top left, respectively. Each black dot represents the final locations of maze navigation robots at the end of each evolution generation until either the goal or when the maximum number of evaluations were reached. In both maps, novelty was able to traverse far more of the maze space earlier when compared to the fitness-based method. All methods were able to reach the goal with the exception of the hard map (d). The fitness function used was a distance of the navigators position from the goal (straight-line distance). Novelty metric used to measure behavioural sparsity: location (x, y co-ordinates) of the navigator (Lehman and Stanley, 2011).

This implication shows that NEAT is an effective algorithm for novelty search and encourages exploration of the search space effectively.

2.2.3 Hybrid Search

Researchers (Cuccu and Gomez, 2011; Inden et al., 2013; Huang et al., 2015) have demonstrated that combining objective and non-objective search methods can yield superior performance over just using pure approaches.

Combination between objective and non-objective methods can be implemented in multiple ways. For purposes of illustration and due to our non-objective choice, Novelty Search, we will discuss combination methods between fitness and novelty search.

Selection by fitness and novelty: individuals in this hybrid method are selected using fitness-based tournament selection and novelty. A proportion of the population are selected for based on their fitness and the remaining, novelty. (Inden et al., 2013)

Novelty-based speciation: instead of speciating individuals based on fitness or genotypic similarity, species are maintained based on solutions' behavioural similarity. (Inden et al., 2013)

Weighted-sum of novelty and fitness: novelty scores and fitnesses are combined linearly to create a hybrid measure of performance. (Inden et al., 2013; Huang et al., 2015)

The weighted-sum approach was selected for its simplicity and its ability to outperform nearly all other more complex methods in Inden et al., 2013 on the pole balancing, four patterns task.

2.2.4 Applications

Objective and non-objective search methods, specifically novelty search have been used as evolutionary search methods across many applications. These include a range of simulated and physical task environments in the field of evolutionary robotics (ER).

Many (Gomez and Miikkulainen, 1999; Gomez et al., 2008; Igel, 2003) have demonstrated the efficiency of using objective-based evolutionary search methods to

successfully solve the non-markovian double pole-balancing task. Later, it was shown that non-objective and *hybrid* search methods are able to successfully solve the same task with greater efficiency (Mouret and Doncieux, 2012; Cuccu and Gomez, 2011; Inden et al., 2013; Huang et al., 2015).

Although these tasks demonstrate the method's feasibility in single-agent control, researchers have extended the literature with experiments using objective, non-objective and hybrid approaches for multi-agent and collective swarm ER tasks.

Gomes et al. (2013a), Gomes et al. (2016), and Nitschke and Didi (2017) have shown that NE was suitable for multi-agent ER tasks and hybridizing objective and non-objective search methods can yield better results.

Our research focuses on a multi-agent collective self-driving task where all vehicles have homogeneous controllers testing various search methods.

2.3 Autonomous Vehicles

Recently there has been increasing research attention focused on overcoming the technical challenges for producing self-driving vehicles (KPMG, Center for Automotive Research, 2012). Some companies have demonstrated working production prototypes (Ackerman, 2015), the successors of which are speculated as eventually displacing the need for human drivers on public roads (Motavalli, 2012; Bilger, 2013; Bamonte, 2013).

Our research focus is on automated methods for deriving optimal controllers in self-driving vehicles that are able to generalise across new environments such as different road networks and traffic conditions. In this section we explore past research on controller and morphology design as well as the current state-of-the-art in terms of self-driving vehicles in the real-world.

2.3.1 Controller Design

There has been a significant amount of work on evolving driving behaviours for simulated (Togelius and Lucas, 2006; Ebner and Tiede, 2009; Drchal and Koutník, 2009; Talamini et al., 2018) and physical vehicles (Beeson et al., 2008; Kesting et al., 2010; Furda and Vlacic, 2011) though such studies produce controllers that are suitable for the given task environment and evolved driving behaviours rarely function across a broad range of environments (Togelius and Lucas, 2005; Togelius and Lucas, 2006; Cardamone et al., 2010).

In Togelius and Lucas (2006), ANNs were utilised and a 3-layer neural network with fixed topology was used. The input layer consisted of at least three input nodes, a bias input of value 1, an input for the speed of the vehicle and an input for the waypoint sensor. Each vehicle had a waypoint sensor which informed the network at what angle the vehicle was approaching the next waypoint on the track. Further inputs were from range-finder or *laser* sensors with values of distances to obstacles recorded by each sensor. The network finally had two outputs, which were for the vehicle's throttle and steering. Only the connection weights were adapted by the EA. The test environment was a 2-dimensional top-down simulation, with no physics simulation other than moving the vehicle in the direction of travel and collision detection with walls.

Ebner and Tiede (2009) demonstrated a method of evolving controllers using Genetic Programming. Genetic programming is another EA that is out of the scope of this research, however, of interest in this particular research is that they used *The open race car simulator* (TORCS), which is a more realistic 3-dimensional racing simulator with more realistic physics (gravity, friction) simulation. Similarly, Cardamone et al. (2010) also used the TORCS framework but utilized NE, specifically online-NE to learn *on-the-fly* so that trained controllers can learn to drive on new tracks. In both instances however, the task was limited to driving a single vehicle around a racetrack as fast as possible, avoiding the track barriers.

Drchal and Koutník (2009) used ANNs for their vehicle controllers and HyperNEAT as the EA. Here, both network weights and topologies were evolved with HyperNEAT, a variant of NEAT that uses indirect encoding. Simple two-wheeled vehicles were trained to drive around an inter-connected roadway. Vehicles had to stay within the boundaries of the road, and not collide with each other if they had to cross paths. The output of the network controlled each wheel individually, thus allowing turns and rotations to be made. Figure 2.9 provides an overview of past research.

Talamini et al. (2018) assessed the impact of ANN evolution using global (system-wide) versus local (individual vehicle) fitness in terms of speed, safety and efficiency for a collective driving task using NEAT in a simulated two-dimensional environment. Individual vehicles had five laser sensors spread out in 180° which provided input into the network for three distances: to the closest car, to the closest roadside, and to the closest intersection. Besides these three inputs from each sensor, three additional inputs were added: current car speed, distance to the target and direction of the target resulting in 18 input nodes. There were two outputs from the network: steering and acceleration (or braking, in the case of negative acceleration). Results did not show statistical significance between local (selfish) and global optimisation but did show that selfish fitness may yield more robust controllers, generalisable to different environments. Although this task was collective, vehicles started at

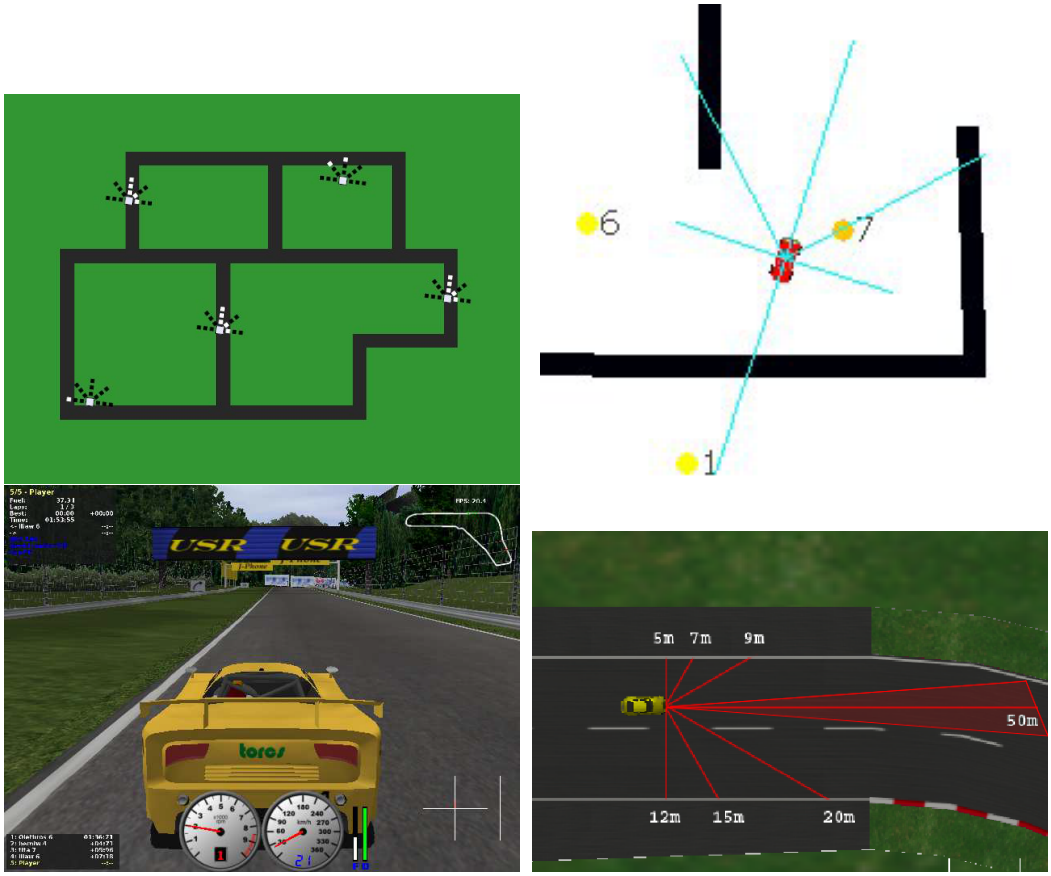


Fig. 2.9: **Clockwise from Top Left: 1.** Simplistic simulated 2D environment, road network and test vehicles with sensors in Drchal and Koutník (2009). **2.** 2D simulated environment in (Togelius and Lucas, 2006) with limited physics, shown here is the sensor configuration on the test vehicle at a corner of a track. **3.** Ray-cast sensor configuration used in Cardamone et al. (2010) for online-NE in TORCS, a high fidelity simulation framework. Sensors used to determine distance to track boundaries. **4.** TORCS framework used by Cardamone et al. (2010) and Ebner and Tiede (2009) and various other researchers to test controller designs.

different points in the road network and only encountered each other later on, namely, they were not tasked with travelling in a group.

To the best of our knowledge and the presented past research above, most vehicles travelled in isolation and had to avoid either static or occasionally dynamic obstacles. Collective behaviour evolution has received significant attention in the context of swarm robotics (Beni, 2004; Werfel, 2007) and collective robotics Kube and Zhang, 1994; Watson and Nitschke, 2015, however, there has been relatively little research on the evolution on the evolution of collective driving behaviours on road networks where vehicles are tasked with travelling in a group.

2.3.2 Current Self-Driving Vehicles

Due to the rapid interest by automotive companies to produce autonomous vehicles, significant progress has been made in the past few years. This section will provide a survey of current production and prototype vehicles by various automotive and technology companies.

Furthermore, the National Highway Traffic Safety Administration (NHTSA) has proposed a formal classification system (Highway and Administration, 2013) for different levels of vehicle autonomy.

- **No-Automation (Level 0)**: The driver is in complete and sole control of the primary vehicle controls – brake, steering, throttle, and motive power – at all times.

- **Function-specific Automation (Level 1)**: Automation at this level involves one or more specific control functions. Examples include electronic stability control or pre-charged brakes, where the vehicle automatically assists with braking to enable the driver to regain control of the vehicle or stop faster than possible by acting alone. Almost all modern vehicles have this level of autonomy as ESC and Anti-lock braking systems (ABS) are now mandatory for all new vehicles sold in the United States. (NHTSA, 2016)

- **Combined Function Automation (Level 2)**: This level involves automation of at least two primary control functions designed to work in unison to relieve the driver of control of those functions. This level of autonomy is often achieved by utilizing radar technology combined with camera imaging techniques to determine road bounds and "follow" traffic at safe distances. An example of combined functions enabling a Level 2 system is adaptive cruise control in combination with lane centering. Tesla's autopilot in its Model S, Model 3 and Model X's is currently considered Level 2 autonomy. (Kenwell, 2018)

- **Limited Self-Driving Automation (Level 3)**: Vehicles at this level of automation enable the driver to cede full control of all safety-critical functions under certain traffic or environmental conditions and in those conditions to rely heavily on the vehicle to monitor for changes in those conditions requiring transition back to driver control. The driver is expected to be available for occasional control, but with sufficiently comfortable transition time. This level requires a larger suite of sensors, cameras and advanced image recognition software (often using deep-learning) to determine the bounds of the road and road obstacles. Highly-detailed maps are often required for accurate naviga-

tion and pre-planning to aid with the rest of the control systems. Waymo's self-driving car is an example of limited self-driving automation. Recently, Audi has launched an A8 with Level 3 autonomy capabilities. (Wasef, 2018)

- **Full Self-Driving Automation (Level 4)**: The vehicle is designed to perform all safety-critical driving functions and monitor roadway conditions for an entire trip. Such a design anticipates that the driver will provide destination or navigation input, but is not expected to be available for control at any time during the trip. This includes both occupied and unoccupied vehicles.

Almost all new vehicles today have Level 1 as a safety standard and Level 2 is often an optional safety feature available from many manufacturers. A few manufacturers have released either production (such as Tesla's vehicles for a short period) or experimental vehicles (Waymo's self-driving vehicles) and Audi demonstrating Level 3 capabilities.

2.4 Conclusion

In this chapter, evolutionary algorithms, neuro-evolution with a focus on NEAT and Novelty Search were reviewed. These are algorithms employed in this research for the collective self-driving task. A survey of existing literature on self-driving research using evolutionary methods were listed as well as state-of-the-art technology used in production self-driving capabilities of today's road vehicles.

Current research using evolutionary methods is largely limited to training individual vehicles to drive and testing them in isolation and technologies used in today's production vehicles have achieved level 3 autonomy. Furthermore, existing evolutionary methods only use objective-based search.

Our research is focused on Level 4 or *full automation* by NHTSA's classifications for self-driving vehicles traversing road and highway networks in a collective fashion. We utilize the NEAT neuro-evolution algorithm and investigate the impact of using objective-based and non-objective-based search methods to elicit collective self-driving behaviours where all vehicles traverse multiple routes simultaneously whilst avoiding each other and other obstacles en-route to their destination.

3 Methods

The focus of this research is on adapting autonomous vehicles to traverse various road environments in a collection and assessing the performance impact of the search method used to guide evolution. This chapter describes the implementation details for the simulator used in this research, algorithmic implementations of NEAT (objective, non-objective and hybrid) and the evaluation functions used for each search method.

3.1 Simulator and NEAT Implementation

An extension of *UnityNEAT* (developed by Jallov (2014)) based on SharpNEAT by Green (2003) (written in C#) was used to simulate physically realistic 3D vehicles, sensors, roads and obstacles. Unity[1] is a multi-platform game engine that allows for easy development of 2D or 3D games.

The vehicle controllers are evolved with the goal of being able to maximize the average distance traversed (measured by way of checkpoints passed) on tracks with obstacles, perpendicular traffic and oncoming traffic whilst minimizing collisions with obstacles and other vehicles. Sensory input and motor output layers are fixed and NEAT adapts the number of hidden layer nodes and connectivity between sensory inputs and motor outputs.

The simulation is performed for homogenous teams where groups of vehicles travelling together on the track have identical controllers.

NEAT Parameters

The parameters used in NEAT for the experiments are outlined in table 4.1 and discribed in more detail in the next chapter. The default activation function employed by NEAT, the *steepened* sigmoid function is used for our research:

[1] http://unity3d.com

$$\sigma z = \frac{1}{1 + e^{-4.9 \cdot z}} \tag{3.1}$$

The sigmoid function $\sigma(z)$ is graphed in figure 2.2, top-left.

3.2 Evaluation Functions

A focus area of this research is to investigate the different methods used to direct evolutionary search in NEAT. To determine the impact these methods have on search efficiency, this section will discuss the three different evolutionary search methods used: objective search (fitness), novelty search (NS) and hybrid search. Experimental set-up and parameters are described in more detail in the next chapter.

3.2.1 Fitness Function

In the objective search method, controllers were awarded a fitness equalling the portion of the track's length covered (via checkpoints) over 45 simulation (task trial) iterations:

$$\text{fitness}(x) = \frac{1}{cars} \sum_{i=0}^{cars} \left(\frac{cp_{passed}}{cp_{total}} * 0.9^{coll} \right) \tag{3.2}$$

where $cars$ represents the number of cars in a group, cp_{passed} denoting the number of checkpoints vehicles successfully pass, cp_{total} is the total number of check-points on that track, and $coll$ is the number of collisions[2] that a vehicles was involved in. Collisions cause an exponential decay to the fitness of an individual, thus allowing the evolution to be lenient individuals that collide rarely but penalise them exponentially as collision counts increase.

Thus vehicle controllers that minimized the number of collisions and maximized the number of check-points passed, were selected for by NEAT.

3.2.2 Novelty Metric

The non-objective approach, NS, requires a *novelty metric* to determine an controller's behavioural novelty. This novelty is described by *sparseness* (equation 3.3), which

[2]The value of 0.9 for the base was selected after experimenting with a few other values during preliminary experiments. Values lower than 0.9 (in increments of 0.1) resulted in slower evolution and often caused evolution to stagnate.

is a combination of a behaviour characterisation (which describes an controller's behaviour) and a distance metric. An individual's novelty is thus a relative score on how different its behaviour is compared to others in the population and novelty archive.

A novelty archive is maintained to store *novel* controllers as they appear ensuring that previously novel behaviours are not lost and reappear during evolution.

K-nearest neighbours (composed of other behaviourally similar solutions in the novelty archive and solutions in the population at the same given generation) are used to compute a controller's sparseness and thus novelty. Sparseness is defined as:

$$\text{Sparseness}(x) = \frac{1}{k}\sum_{i=0}^{k} dist(x, \mu_i) \tag{3.3}$$

where μ is the ith-nearest neighbor of x with respect to the novelty metric, and where the distance component in equation 3.3 uses the Euclidean distance derived by the Pythagorean theorem (Gower, 1982).

Novelty Archive

The top fifteen of the population (100) most novel solutions at each generation are added to the novelty archive. The novelty archive is unbounded which means the maximum size of the novelty archive is $15 \times n_{generations}$ at the final generation. Table 4.1 presents the archive size and addition rate for novelty search and population size for the experiments.

Behaviour Characterisations

The behaviour characterisation (BC) describes the dimensions(s) chosen for the novelty metric and has an impact on the search as it induces pressure to exhibit novel behaviour based given attributes. To ascertain which BC yields the highest performance (relating to the task), three BCs were tested and compared:

- **Speed**: Individual vehicle's velocity measured in meters per second (m/s).

- **Speed and Cohesion**: Individual vehicle's velocity and the distance between the vehicles. The distance between the vehicles is the line-of-sight distance in $meters$.

- **Location**: Vehicle locations (x and z coordinates) in the simulated environment.

For each of the metrics, values were sampled at fixed time-steps in the simulation and vectors of behaviours were compared with each other to determine an individual solution's sparseness.

Behaviour Sampling

The behaviour characterisation is a vector comprised of values that are sampled at fixed intervals of 1/100th of total simulation time-steps per generation. Each value in the vector is a dimension that is used to compare against other individuals using the pythagoras equation.

One hundred samples are collected for each vehicle (sampling rate, table 4.1) in a group and since each vehicle has their own values during sampling, the vector of behaviour characterisation values are combined for each vehicle in the group making a final vector used for sparseness calculation.

To ensure that an individual vehicle's behaviour is compared to the most similar vehicle's behaviour in another solution's behaviour vector, the final behaviour vector is sorted by the aggregate values of the sub-vectors.

3.2.3 Hybrid Function

The hybrid search method used in this research linearly combines novelty and fitness as in Huang et al. (2015) to create a weighted sum. The score that each individual receives for the hybrid method is defined as:

$$\text{score}(i) = \rho.\overline{fit}(i) + (1-\rho).\overline{nov}(i) \tag{3.4}$$

Where, $\rho = 0.5$, combining fitness and novelty equally which are normalized according to:

$$\overline{fit}(i) = \frac{fit(i) - fit_{min}}{fit_{max} - fit_{min}}, \overline{nov}(i) = \frac{nov(i) - nov_{min}}{nov_{max} - nov_{min}} \tag{3.5}$$

Where, nov_{min}, fit_{min} are the lowest novelty and fitness values in the population, respectively and nov_{max}, fit_{max} are the highest.

It was shown in Huang et al. (2015) increasing or decreasing the ρ value biased the results heavily to either objective performance or NS performance. As it is also unclear for general tasks whether objective or NS will perform better and only with *a priori* experiments for either pure approached will an experimenter know if it is better to bias a weighted-sum hybrid approach to either objective or fitness. To remove this bias and leave ρ as a parameter-tuning excercise, we selected $\rho = 0.5$.

3.3 Conclusion

In this chapter, the simulator used for these experiments along with the NEAT implementation was outlined and the three different evaluation function implementations for fitness, novelty search and hybrid described.

The behaviour characterisation for novelty search and hybrid may have effect on search performance and thus we comparatively assess three different metrics, *speed*; *speed and cohesion* and *location*.

For simplicity and supported by previous research, we create a equally-weighted linearly-combined fitness-novelty hybrid approach that will be used to assess whether our hypothesis that hybrid approaches can generalise over various task environments is correct.

4 Experiments and Results

The ANN controllers are adapted by NEAT for each evolutionary search method, objective, non-objective and hybrid. Thereafter, generalisability tests (evaluations) are performed for each of the adapted controllers to determine performance differences for each search method.

This chapter describes the simulated vehicles, task environments and experiments that will be run followed by results.

4.1 Vehicle Simulation

The vehicles in this simulation have a BMW E46 M3 body model. The simulation vehicle has similar acceleration and braking to that of a real-world vehicle. Maximum steering angle is $25°$ and top speed is capped at $120km/h$.

Each vehicle has five radar sensors that surround the front of the vehicle as depicted in figure 4.1. These simulated sensors have a pyramidal shape and when an obstacle is in range, the distance between the vehicle and the obstacle is fed into the ANN controller. Each sensor represents an input node in the ANN controller. Three more inputs are fed into the ANN controller, the bias input θ, angle to the next way-point and current vehicle velocity.

ANN controllers manoeuvre the vehicle along the track and avoids obstacles using inputs from its sensors. When a vehicle reaches a way-point, its destination is updated to the next way-point along the track. A total of 8 input nodes and two outputs are predefined in the network. Hidden nodes and connections between all nodes are adapted by NEAT. An example evolved network is depicted in figure 4.2.

For training, groups of three vehicles are adapted together in homogeneous teams.

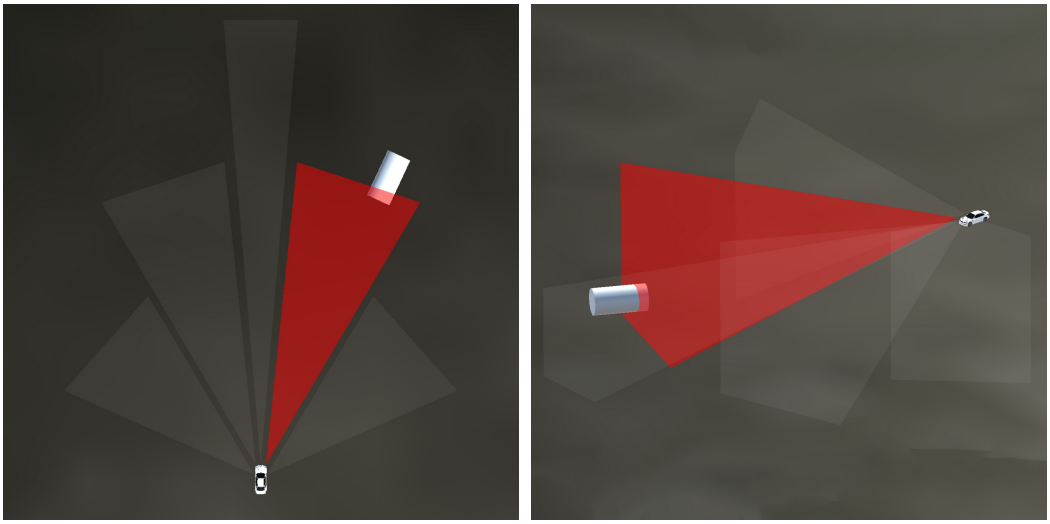


Fig. 4.1: Sensor Configuration: Each vehicle has five pyramidal sensors covering the forwards direction of the vehicle. An example of the sensor detects objects is depicted in red where sensor no. 4 has detected an obstacle. This fan layout surrounding the front of the vehicle is consistent with past research (Drchal and Koutník, 2009; Togelius and Lucas, 2006; Cardamone et al., 2010) and was thus selected.

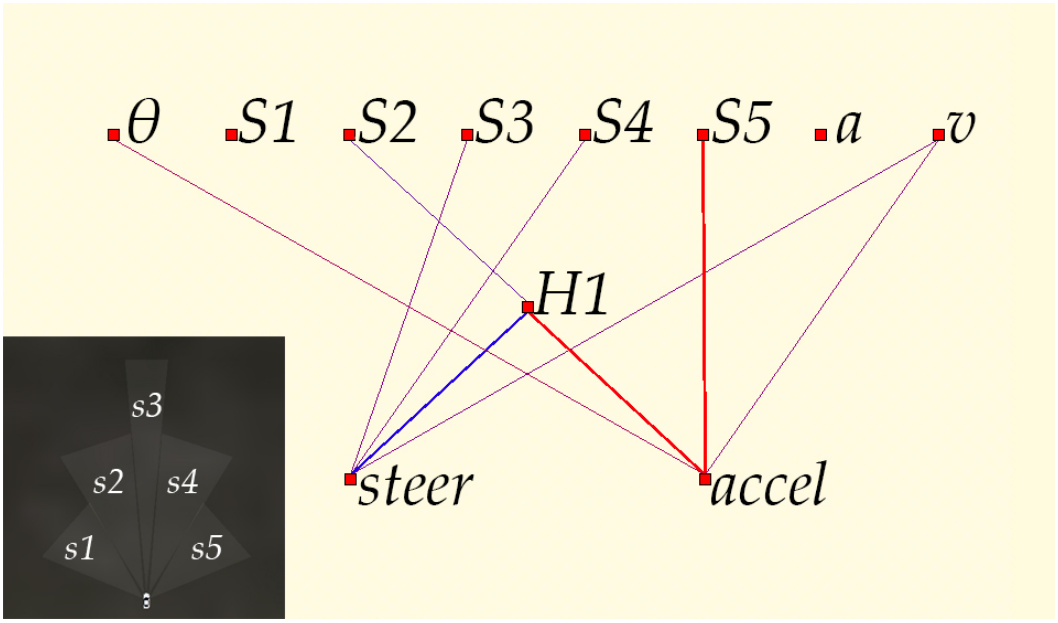


Fig. 4.2: Example ANN controller: each sensor on the vehicle correspond to an input node in the ANN ($S1$ to $S5$). Other inputs include the bias input, θ, angle to the next way-point, a and current speed of the vehicle, v. This example has one hidden node, $H1$. The controller outputs control behaviour via its output $steer$ and $acceleration$ nodes.

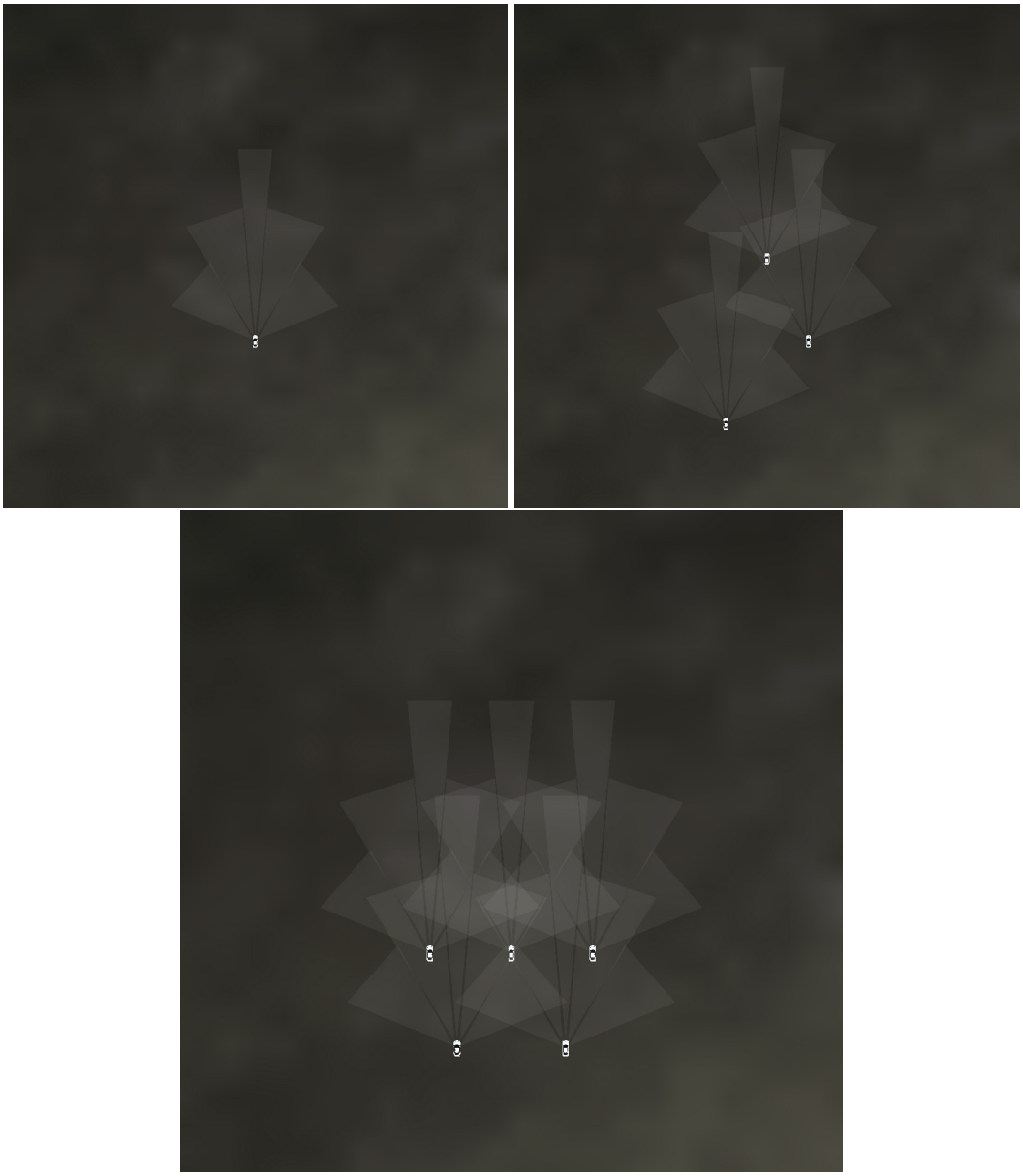


Fig. 4.3: Vehicle group layouts for 1, 3 and 5 vehicle set-ups. The three vehicle set-up is used for controller adaptation by NEAT whilst all set-ups are used for evaluation on unseen tracks.

Parameter	Value
NEAT Population size	100
NEAT Species count	10
NEAT Activation scheme	Acyclic
NEAT Activation function	SteepenedSigmoid
NEAT Complexity regulation strategy	Absolute
NEAT Complexity threshold	21
Novelty Search archive size	Unbound
Novelty Search archive addition rate	15 per generation
Novelty Search K-nearest neighbours	15
Behaviour Characterisation Sampling Rate	1/100th run length (100 Samples)
Hybrid Weighted-Sum Proportion (ρ)	0.5

Tab. 4.1: Neuro-Evolution (NE) and Experiment Parameters. *Parameters with minimal impact on evolution are excluded.*

4.2 Task Environments

The different tracks used in this research are divided up into training and evaluation tracks. The training track is where vehicles controllers are adapted by NEAT. The controllers are then evaluated on *unseen* evaluation tracks that present different challenges for the controllers.

Further to evaluating controllers in unseen environments of varying difficulty, vehicle group sizes are also varied to simulate increased traffic.

4.2.1 Checkpoints

Checkpoints are placed along the tracks to guide the vehicles (in their default driving heuristic) along the track's path, similar to how GPS navigates a route. Checkpoints are also used to determine how far a vehicle has travelled. The fitness function (see section 3.2.1) accounts for the number of checkpoints each vehicle has passed, where the number of checkpoints a vehicles passes, relative to the total checkpoints, contributes to controller fitness.

To ensure normalisation across tracks, all tracks have a total of 10 checkpoints that are spread equally apart.

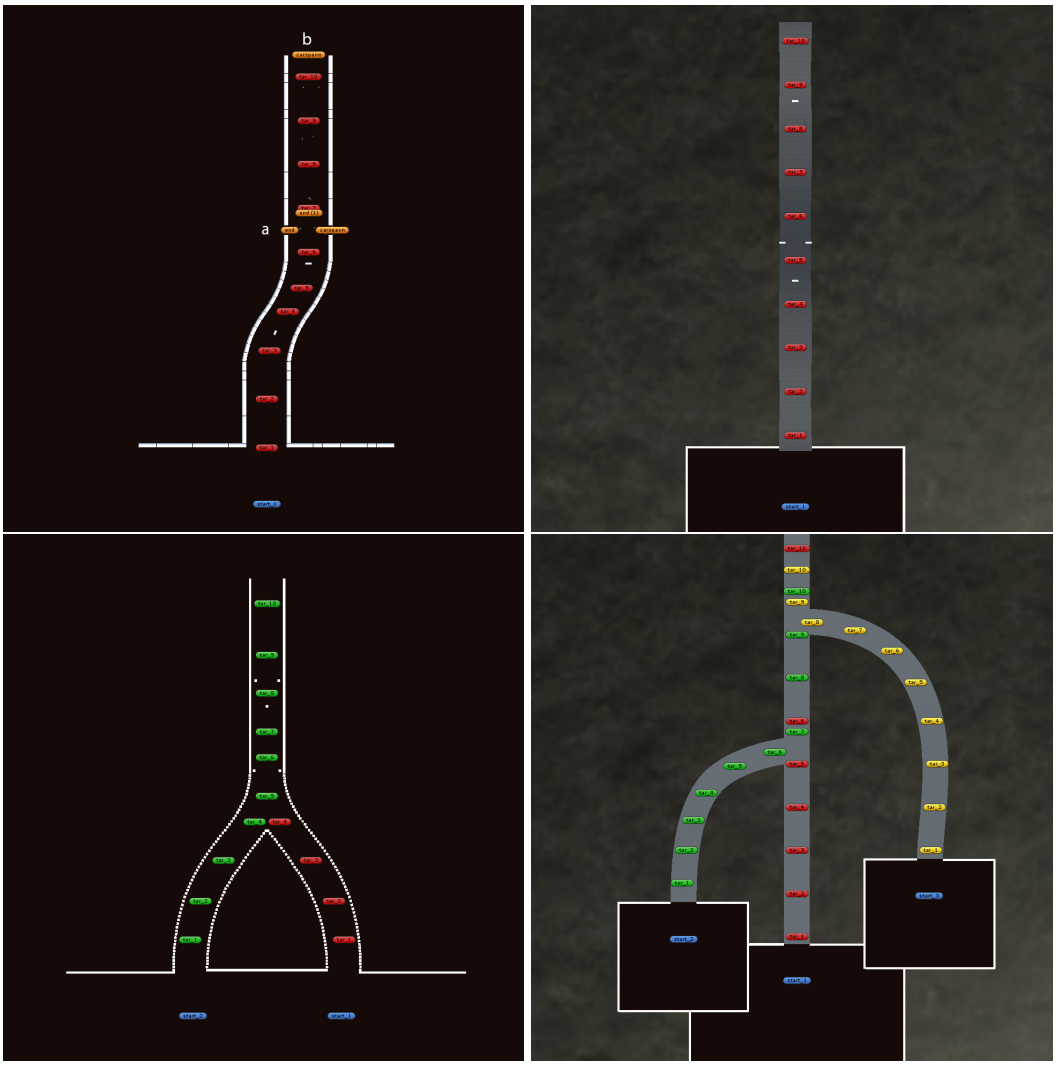


Fig. 4.4: Tracks used for evolution and evaluation of ANN controllers. Each track has ten checkpoints denoted in red (green and yellow for secondary checkpoints) for each starting position (denoted by blue). **Top-left:** *Training track* used for adapting ANN controllers. Two static obstacles are placed between the third and forth target and fifth and sixth targets. Dynamic obstacles (denoted by *a* and *b*) in the form of vehicles crossing the road and oncoming traffic also make this track more difficult to complete. The other three tracks (top-right: 2, bottom-left: 3 and bottom-right: 4) are unseen by controllers and present different challenges for controllers. Each unseen track has three variants with increasing number of obstacles to vary difficulty.

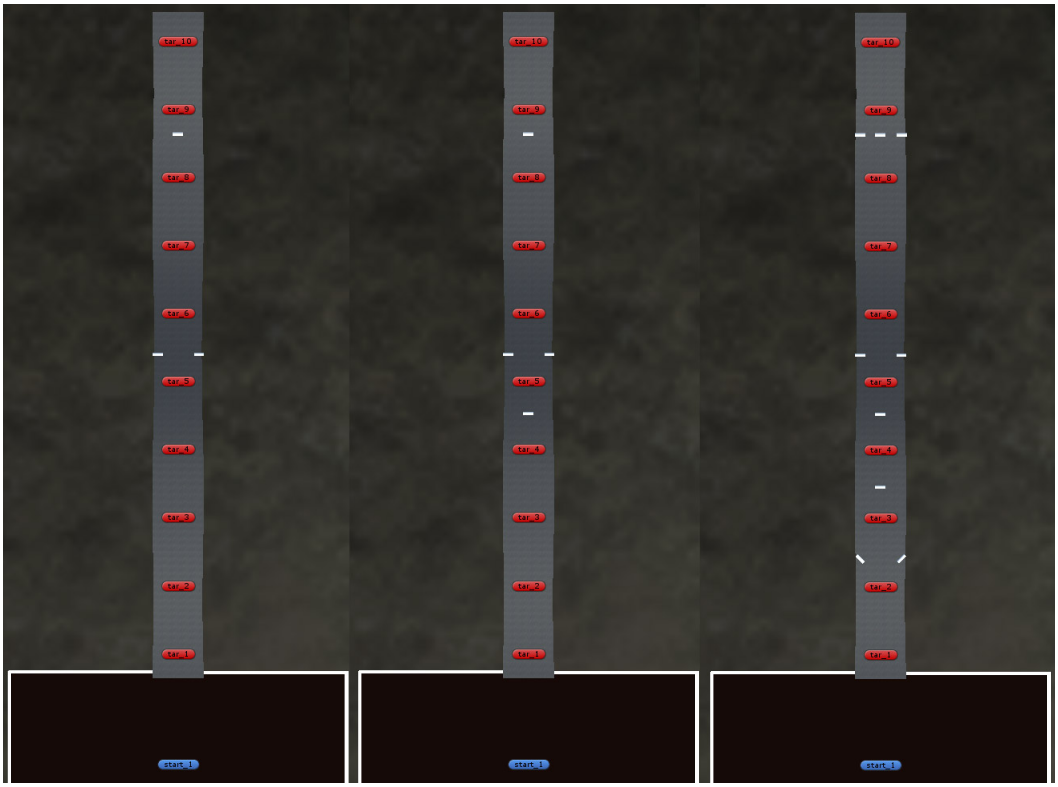


Fig. 4.5: Track 2. Left to right: easiest track contains three obstacles, medium track contains four obstacles and most difficult track containing nine obstacles.

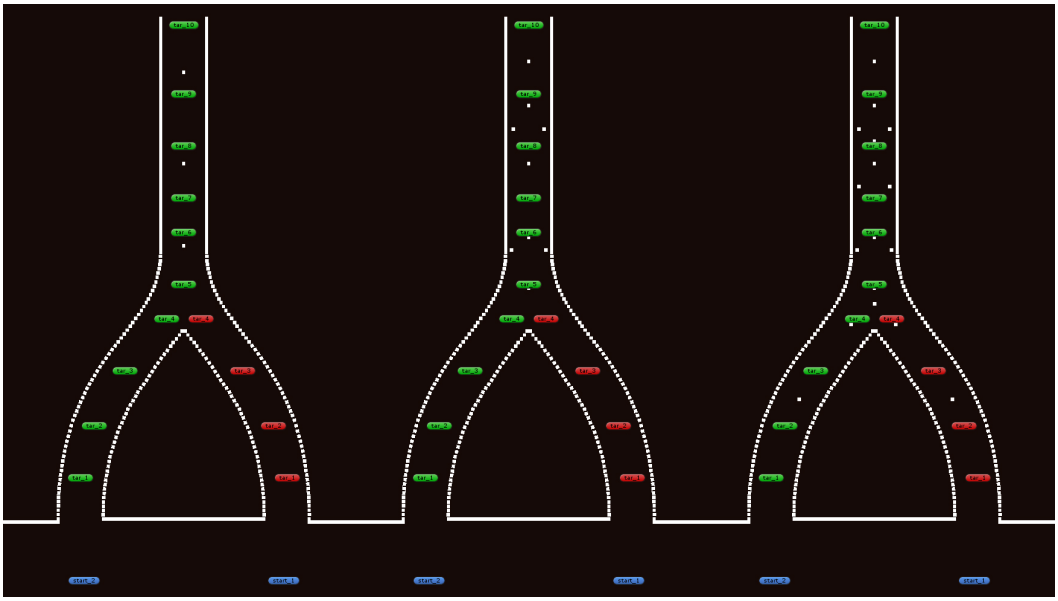


Fig. 4.6: Track 3. Left to right: easiest track contains three obstacles, medium track contains nine obstacles and most difficult track containing seventeen obstacles.

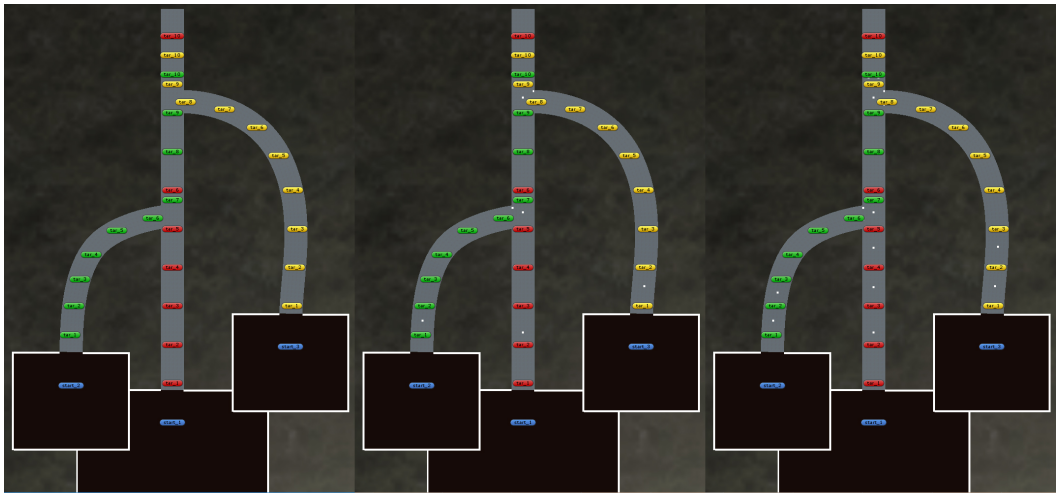


Fig. 4.7: Track 4. Left to right: easiest track contains no obstacles, medium track contains seven obstacles and most difficult track containing sixteen obstacles.

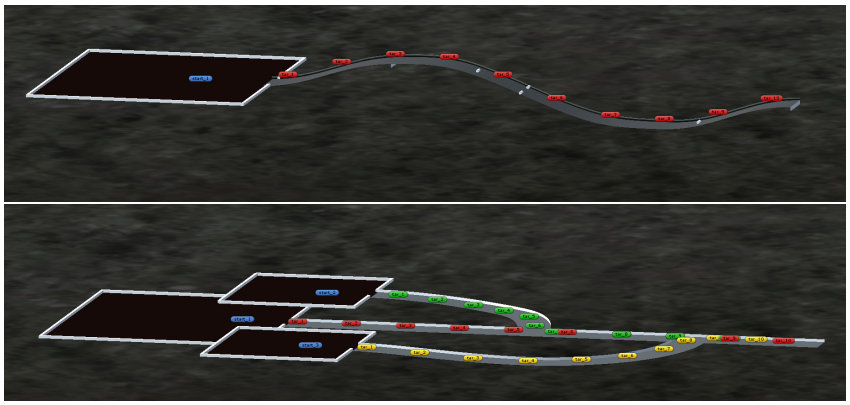


Fig. 4.8: Unlike the training track and track three, tracks two and four have height variances which could affect sensor coverage. These tracks simulate real-world hill scenarios. Track four has three starting points which means three groups of vehicles start at the different locations and they all end up meeting in the center lane. The starting positions all start on varying heights.

Evolution Parameters	
Parameter	**Value**
Number of vehicles	3
Number of runs	20
Task trial duration (seconds)	45
Generations	100
Evolutionary search methods	3 + 3 BC for NS
Tracks	1
Total Runs	$20 * 6 = 120$
Generalisability Evaluation Parameters	
Parameter	**Value**
Number of vehicles	[1, 3, 5]
Number of runs	20
Maximum trial duration (seconds)	100
Controllers Tested	60
Tracks	1 + (3 * 3) = 10
Total Runs	$20 * 60 * 3 * 10 = 36000$

Tab. 4.2: **Top:** Experiment runtime parameters for NE controller adaptation. **Bottom:** Generalisability test runtime parameters.

4.3 Neuro-Evolution Experiments

Twenty runs of 10 000 (100 generations, 100 population size) (see tables 4.1 & 4.2) evaluations are performed on the training track (figure 4.4, top-left) using vehicle group configuration 3 (figure 4.3, top-right) for each evolutionary search method described in chapter 3. To ascertain the best-performing behaviour characterisation (section 3.2.2) for use in the hybrid search method and as the candidate to represent NS results, three non-objective search method characterisations are run.

4.4 Generalisability Evaluations

These experiments evaluate an adapted controller's ability to traverse unseen environments. Each track (figure 4.4) has checkpoints which contribute to controllers' fitness as vehicles pass. However, the *fitness* earned by controllers when passing checkpoints in these evaluations differ from that described by equation 3.2. Instead of penalising controllers on collisions, the vehicle is immediately stopped on collision, preventing controllers from earning more points by passing checkpoints. If a controlled vehicle collides with another, the other vehicle is also stopped. This is to mimic real-world instances where vehicles should completely avoid collisions.

Given that the *fitness* measurement differs in these experiments, the original training track is included here so that results can be fairly compared with unseen tracks. Table 4.3 outlines all the generalisability experiments that will be run. Since each evolutionary search method produces twenty champions, each evaluation will be run sixty times as described in table 4.2.

4.5 Results

Overall experimental results are discussed in the proceeding section. Neuro-Evolution Results including results behaviour characterisation tests for NS are first presented, followed by controller generalisability results. Objective-based search refers to the search method utilizing the only the fitness function. Non-objective refers to the NS search method and hybrid refers to the combination of both. Refer to section 3.2 for more information on each.

4.5.1 Neuro-Evolution (NE) Results

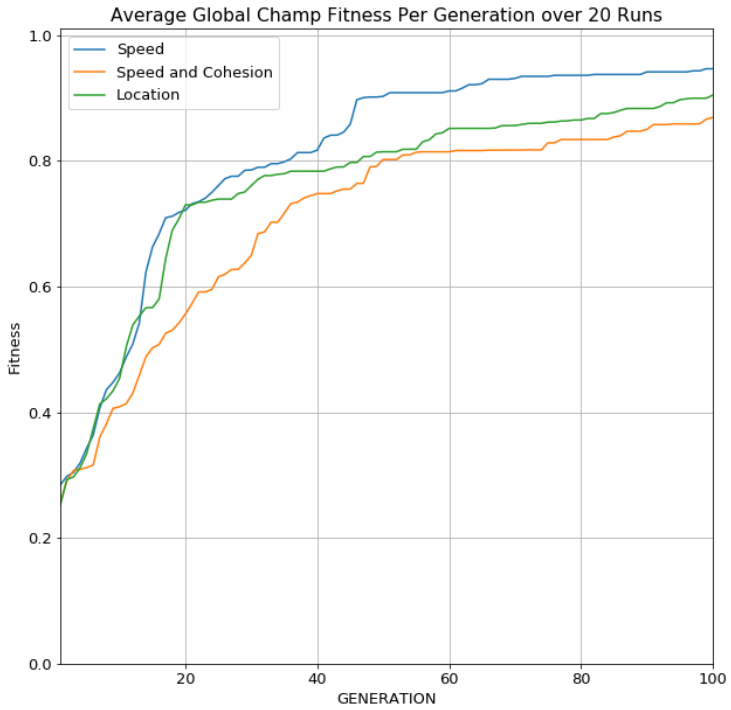
Three behaviour characterisations (see 3.2.2) were tested for the non-objective NS search method. The best-performing was selected to be the candidate to represent the non-objective search method and to be implemented for the hybrid search. Thus, these results will be presented and discussed first before a comparison is made between objective, non-objective and hybrid results.

Behaviour Characterisation Comparison

Results from NE controller adaptation comparing the three behaviour characterisations *speed*, *speed & cohesion* and *location* indicate that there was no statistical significant difference between them (see figure 4.9). Thus, *speed* was selected for comparison between the other search methods and for use in the hybrid method since it had the highest mean fitness overall. A simpler version of the training track (figure 4.4 top-left) was utilized for these these experiments (namely, the obstacle between target 3 and target 4 removed).

Overall NE Results

Results from NE controller adaptation comparing the three search methodologies, objective (fitness), non-objective (NS) and hybrid show that there was a statistical



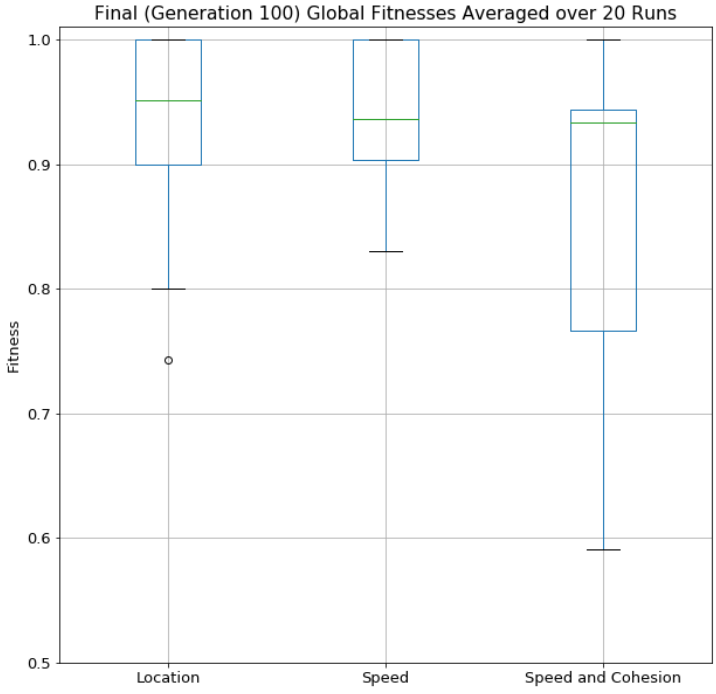


Fig. 4.9: Novelty Search Behaviour Characterisations: Average fitness of the fittest individual over 20 runs for *top:* each generation and *bottom:* final generation. *Mann-Whitney U*, $p \leq 0.05$ statistical tests indicated no statistical difference between characterisations in the final generation.

significance between hybrid (it outperformed both) and the other two methods but no statistical significance between fitness and novelty search (refer to table 4.4).

To determine the effectiveness of each method in adapting NE controllers, heat-maps visualising the portion of genotypes in each task performance bucket per generation is shown in figure 4.11. The hybrid approach has a more even spread of genotypes throughout all generations except the first generation, where a large concentration of genotypes are found at 0.2 task performance. It is also clear that for the objective-based search approach, a larger proportion of the genotypes start at higher task performance in early generations compared to the other methods and most genotypes remain in this region with few high-performers in later generations (this is evidenced in figure 4.10 right, where a large variability of solutions in the final generation exist for the objective-based approach). For almost all generations for novelty search, genotypes are spread evenly across the performance space. As outlined in related work (Lehman and Stanley, 2011; Velez and Clune, 2014), this could be due to the more explorative nature of novelty search, maintaining genotype diversity throughout generations but unfortunately at the cost of task performance in this study.

4.5.2 Generalisability Results

The generalisability evaluations test the ability of the different search methodologies' controllers to navigate completely new environments.

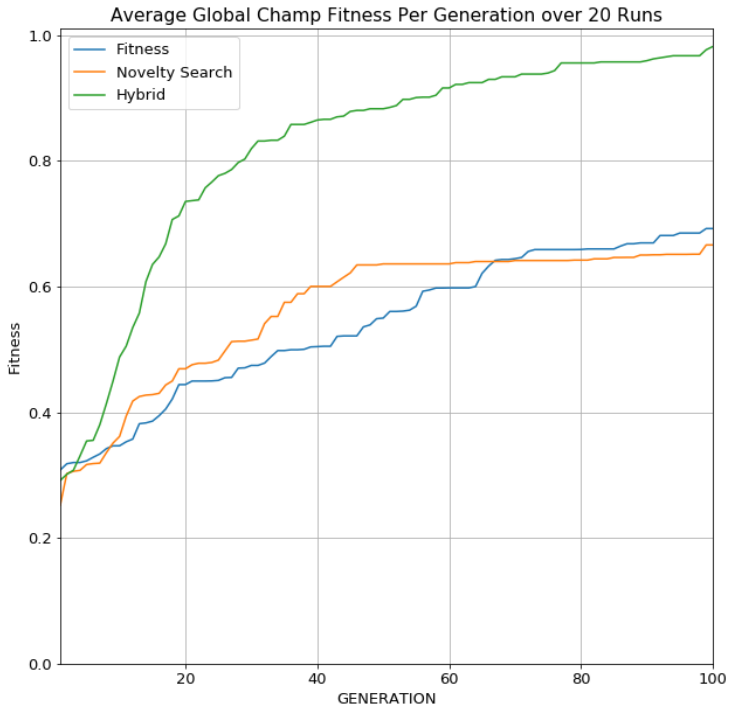
Controllers were tested in unseen track environments as well as varying vehicle configurations to assess their robustness. The three methodologies' controllers were tested on 10 different tracks (9 unseen) and 3 different vehicle configurations (group sizes). Refer to section 4.4 for more details on these experiments.

These results present overall performance and is followed by performance grouped by vehicle configuration sizes and performance grouped by track.

4.5.3 Overall Evaluation Results

All three search methods yielded above 60% task performance on the NE experiments with Hybrid significantly outperforming NS and Fitness.

However, the aggregate results for all evaluations indicate that the methods were only able to succeed between 15-30% of the desired fitness. This is expected as the evaluation tasks are significantly more difficult as controllers have to navigate



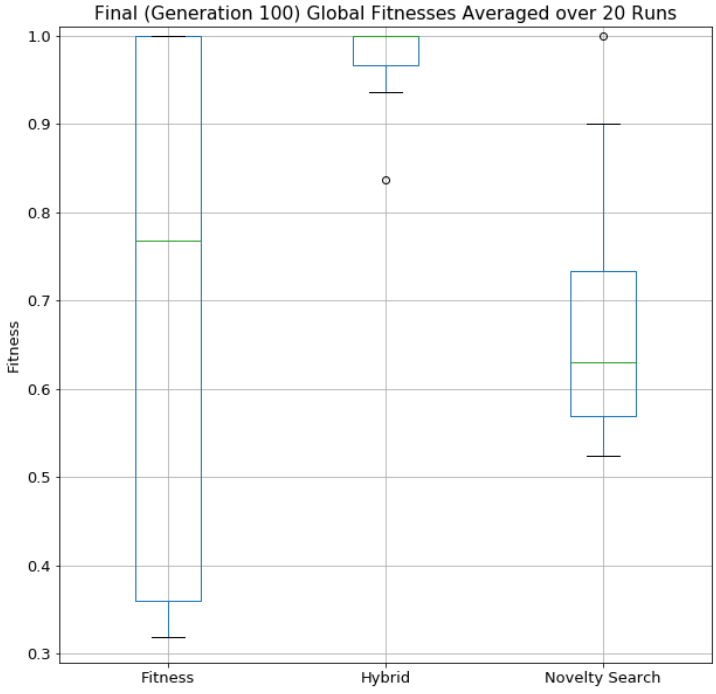


Fig. 4.10: **All NE Results:** Average fitness of the fittest individual over 20 runs for *top:* each generation and *bottom:* final generation. *Mann-Whitney U*, $p \leq 0.05$ statistical tests indicated statistical significance between search methods [hybrid, fitness] and [hybrid, novelty search] but no statistical significance between search methods [fitness, novelty search] in the final generation.

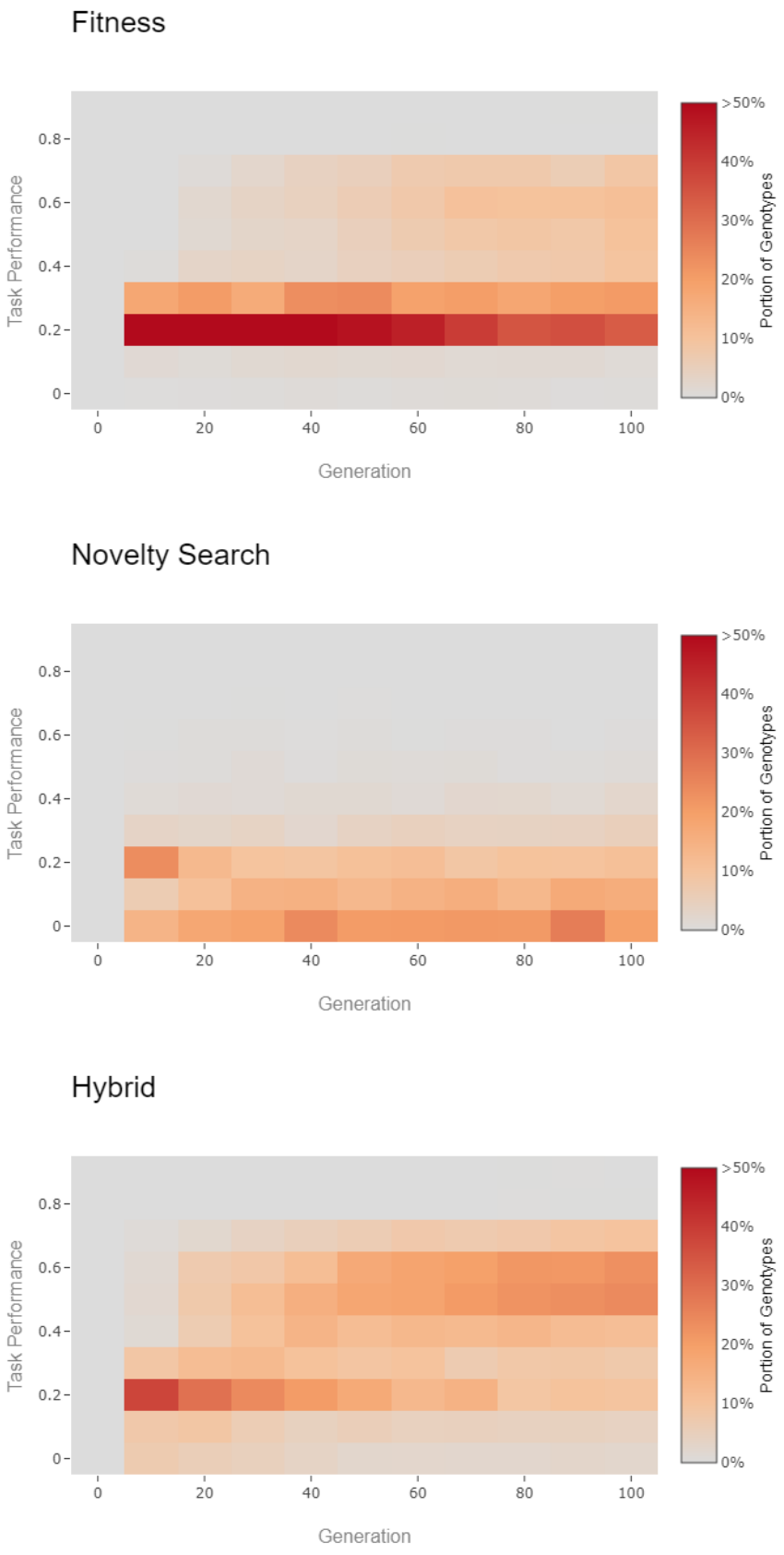


Fig. 4.11: Heat-maps showing portions of genotypes evolved via each method (Top: Fitness, Middle: Novelty Search and Bottom: Hybrid) per generation. Darker shading indicates a higher portion of genotypes.

unseen environments and vehicles are completely stopped when collisions occur. Figure 4.12 present all aggregate results with fitness outperforming both the NS and hybrid search methods, despite performing the worst (tied with NS) in the NE experiments. All results are statistically significant with hybrid performing the worst (see table 4.4).

4.5.4 Evaluation Results by Vehicle Configuration

To ascertain whether some vehicle configurations performed better than others for specific search methods, results were grouped by vehicle configuration sizes and presented in figure 4.13. Table 4.5 outlines all statistical tests between each method for each vehicle configuration. All results are statistically significant at $p \leq 0.05$. NS outperformed all other methods for vehicle configuration 1 (see figure 4.3, *top-left*) for all tracks and fitness outperformed all other methods for vehicle configurations 3 and 5 for all tracks.

4.5.5 Evaluation Results by Track

To determine if some tracks were more difficult for controllers than others, results were grouped by track and aggregated over vehicle configurations and is presented in figure 4.14. Table 4.6 outlines all statistical tests between each method for each track. Besides statistical insignificance between fitness and NS on track 3 for all difficulty variants and statistical insignificance between fitness and hybrid on track 3 easy, all other results are statistically significant at $p \leq 0.05$ (see table 4.6). On the training track, the hybrid-based controllers were able to outperform and generalise over all vehicle configurations far better than objective-based and NS controllers. Otherwise, for all other tracks except track 3 easy (due to statistical insignificance), hybrid performed the worst. On tracks 2 and 4, fitness outperformed all other search methods for all vehicle configurations.

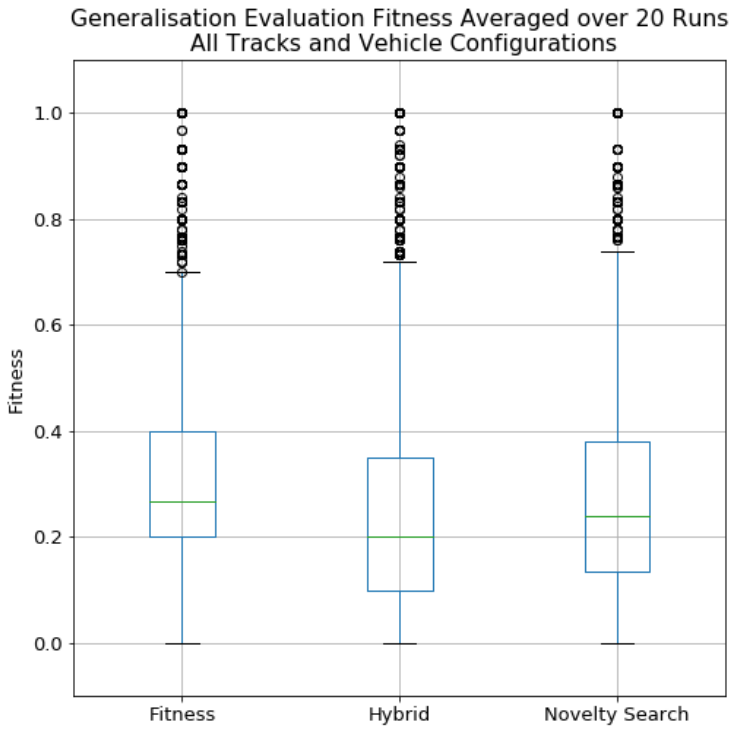


Fig. 4.12: All Generalisability Evaluation Results: Average fitness of the fittest individual over 20 runs for each search method on unseen tracks and different vehicle group sizes. *Mann-Whitney U*, $p \leq 0.05$ statistical tests indicated statistical significance between all search methods.

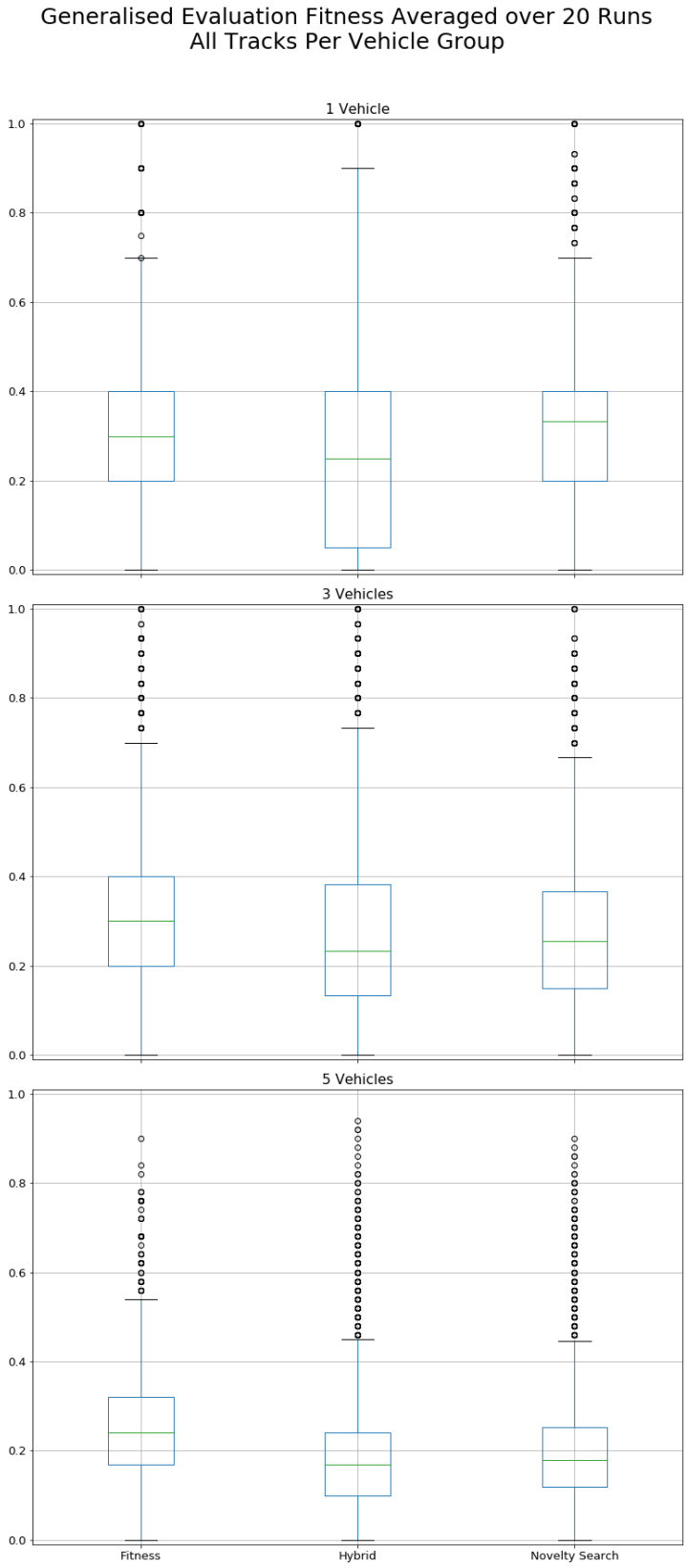


Fig. 4.13: Generalisability Evaluation Results per Vehicle Configuration: Average fitness of the fittest individual over 20 runs for each search method on unseen tracks grouped by vehicle group sizes.

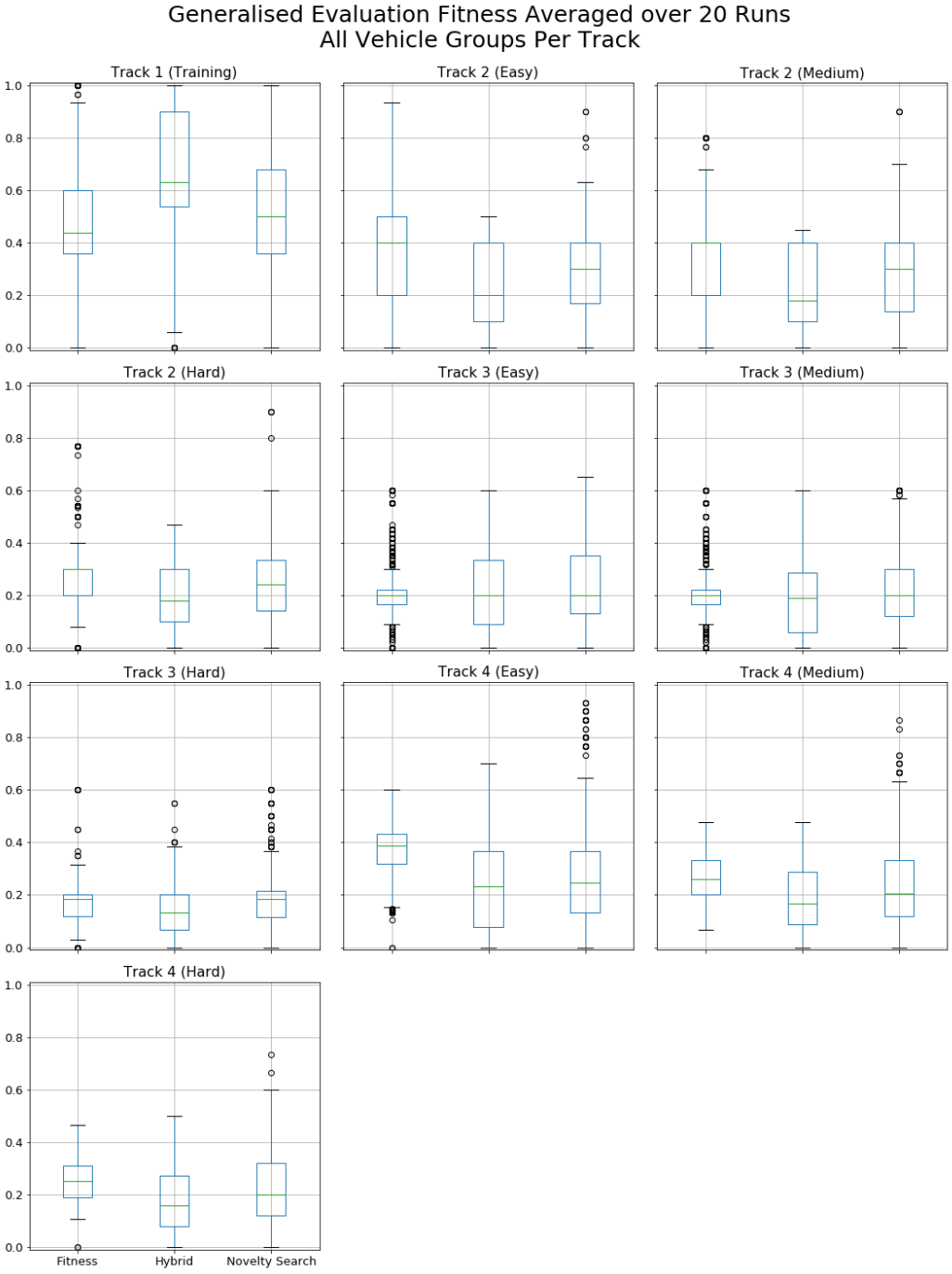


Fig. 4.14: **Generalisability Evaluation Results per Track:** Average fitness of the fittest individual over 20 runs for each search method for various vehicle group sizes, results grouped by performance per track.

Track	Difficulty Variant	Cars	Trials
Training (Track 1)	N/A	1	20
Training (Track 1) - same as evolution	N/A	3	20
Training (Track 1)	N/A	5	20
Track 2	Easy	1	20
Track 2	Medium	1	20
Track 2	Hard	1	20
Track 2	Easy	3	20
Track 2	Medium	3	20
Track 2	Hard	3	20
Track 2	Easy	5	20
Track 2	Medium	5	20
Track 2	Hard	5	20
Track 3	Easy	1	20
Track 3	Medium	1	20
Track 3	Hard	1	20
Track 3	Easy	3	20
Track 3	Medium	3	20
Track 3	Hard	3	20
Track 3	Easy	5	20
Track 3	Medium	5	20
Track 3	Hard	5	20
Track 4	Easy	1	20
Track 4	Medium	1	20
Track 4	Hard	1	20
Track 4	Easy	3	20
Track 4	Medium	3	20
Track 4	Hard	3	20
Track 4	Easy	5	20
Track 4	Medium	5	20
Track 4	Hard	5	20

Tab. 4.3: **Schedule of Generalisability Experiments:** Each of the listed experiments are run for all sixty controllers from the three evolutionary search methods.

Result	Search Methods	$p \leq 0.05$ (Statistically significant)
Controller Evolution (NE)	Fitness vs Hybrid	Y
Controller Evolution (NE)	Fitness vs NS	N
Controller Evolution (NE)	Hybrid vs NS	Y
Generalisability	Fitness vs Hybrid	Y
Generalisability	Fitness vs NS	Y
Generalisability	Hybrid vs NS	Y

Tab. 4.4: ***Mann-Whitney U* Statistical Tests** for each search method for controller evolution and generalisability evaluations.

Vehicles	Search Methods	$p \leq 0.05$ (Statistically significant)
1 Vehicle	Fitness vs Hybrid	Y
1 Vehicle	Fitness vs NS	Y
1 Vehicle	Hybrid vs NS	Y
3 Vehicles	Fitness vs Hybrid	Y
3 Vehicles	Fitness vs NS	Y
3 Vehicles	Hybrid vs NS	Y
5 Vehicles	Fitness vs Hybrid	Y
5 Vehicles	Fitness vs NS	Y
5 Vehicles	Hybrid vs NS	Y

Tab. 4.5: ***Mann-Whitney U* Statistical Tests** for each search method per vehicle configuration, aggregated over tracks.

Track	Search Methods	$p \leq 0.05$ (Statistically significant)
Track 1	Fitness vs Hybrid	Y
Track 1	Fitness vs NS	Y
Track 1	Hybrid vs NS	Y
Track 2 (Easy)	Fitness vs Hybrid	Y
Track 2 (Easy)	Fitness vs NS	Y
Track 2 (Easy)	Hybrid vs NS	Y
Track 2 (Medium)	Fitness vs Hybrid	Y
Track 2 (Medium)	Fitness vs NS	Y
Track 2 (Medium)	Hybrid vs NS	Y
Track 2 (Hard)	Fitness vs Hybrid	Y
Track 2 (Hard)	Fitness vs NS	Y
Track 2 (Hard)	Hybrid vs NS	Y
Track 3 (Easy)	Fitness vs Hybrid	N
Track 3 (Easy)	Fitness vs NS	N
Track 3 (Easy)	Hybrid vs NS	Y
Track 3 (Medium)	Fitness vs Hybrid	Y
Track 3 (Medium)	Fitness vs NS	N
Track 3 (Medium)	Hybrid vs NS	Y
Track 3 (Hard)	Fitness vs Hybrid	Y
Track 3 (Hard)	Fitness vs NS	N
Track 3 (Hard)	Hybrid vs NS	Y
Track 4 (Easy)	Fitness vs Hybrid	Y
Track 4 (Easy)	Fitness vs NS	Y
Track 4 (Easy)	Hybrid vs NS	Y
Track 4 (Medium)	Fitness vs Hybrid	Y
Track 4 (Medium)	Fitness vs NS	Y
Track 4 (Medium)	Hybrid vs NS	Y
Track 4 (Hard)	Fitness vs Hybrid	Y
Track 4 (Hard)	Fitness vs NS	Y
Track 4 (Hard)	Hybrid vs NS	Y

Tab. 4.6: ***Mann-Whitney U* Statistical Tests** for each search method per track, aggregated over vehicle configurations.

5 Discussion

This chapter presents a discussion of each method's (objective, NS and hybrid) capabilities, how they balance solution space exploration and exploitation, if they were able to produce controllers that had desired task performance and their ability to generalise over unseen task environments.

A detailed analysis of each along with evolved controller complexity is presented to show how it compared with the literature where hybrid approaches are more suitable at adapting controllers with desired task performance more efficiently and our original hypothesis that *the hybrid approach will adapt controllers that will generalise more effectively across various unseen environments when compared with the pure-objective and pure-novelty methods in this collective self-driving task* (see section 1.3).

5.1 Evolved Task Performance

The NE experiment results showed that all search approaches investigated in this work were appropriate for the collective self-driving task. That is, they were able to evolve controllers capable of achieving 80% task performance given the constrains of a population size of 100 and runs of 100 generations. Furthermore, it showed that the hybrid approach significantly outperformed the objective and NS approaches and was the most suitable for adapting controllers that can yield desired task performance at least 3 times quicker, that is, hybrid was able to achieve 80% task performance by generation 30 whereas the other methods only reached these levels close to generation 100 (see figure 4.10).

Although the NS approach initially outperformed the objective method (up to generation 50), it stagnated in producing more effective controllers between generations 50 and 90 with objective outperforming (but not statistically significantly) at around generation 70.

These results support previous research where hybrid approaches significantly outperform pure objective or non-objective methods in both single-agent tasks (Huang

et al., 2015; Inden et al., 2013; Gomes et al., 2015) and collective behaviour (multi-agent) tasks (Nitschke and Didi, 2017).

5.2 Behavioural Space Analysis

The genotype heat-maps shown in figure 4.11 present the behaviour space of each method's population. The objective approach had most of its population in the 20% - 30% task performance space for almost all generations with a few genotypes moving into higher task performance spaces in later generations. The population evolved by novelty search was more evenly spread with most genotypes in the 0% - 20% space consistently throughout all generations. Only outlying genotypes reached the desired task performance by generation 100. The hybrid approach evolved the largest spread of genotypes across the behaviour space as genotypes were spread in an upward task-performance trajectory. By the later generations, almost all genotypes (above 50%) achieved desired task performance. At the final generation, around 20% of all genotypes were able to achieve 70% task performance whereas less than 5% of genotypes evolved with objective search and novelty search were in this space.

The result where NS produced a more evenly spread population is consistent with previous research (Lehman and Stanley, 2011; Gomes et al., 2012; Velez and Clune, 2014) showing that NS results in more exploration of the behaviour space when compared with traditional objective approaches. Furthermore, the hybrid results are consistent with previous research which shows that NS is less effective in complex tasks with high-dimensional solution spaces and combining NS with an objective approach can help the exploitation of good regions in the broad exploration achieved by NS (Cuccu and Gomez, 2011; Gomes et al., 2013b).

5.3 Generalisability Evaluations

In order to determine whether adapted controllers were able to generalise over new unseen environments, evolved controllers were evaluated over unseen tracks with varying vehicle group sizes as described in section 4.4.

Results presented in figures 4.12, 4.13, 4.14 present the controller performance in the generalisability tests for all results aggregated, results grouped by various vehicle configurations and results grouped by various tracks respectively. In all tests except Track 1 in figure 4.14 (which is consistent with evolution results), the hybrid controllers performed the worst when compared with NS and fitness based controllers.

Our original hypothesis that controllers adapted by the hybrid approach would be able to generalise better over new environments is thus rejected as these controllers performed the worst when compared with controllers adapted via NS and objective search.

In order to understand why the hybrid controllers may have underperformed over unseen environments, we analyse controller complexity of each approach.

5.3.1 Controller Complexity

Network complexity is defined by the number of connections of a controller. Our results (see figure 5.1) showed that there was a statistical significance between the objective-based method (fitness) and NS. This is consistent with previous research demonstrating that NS can produce comparable task performance with simpler networks (Lehman and Stanley, 2011; Gomes et al., 2013b). Although there was no statistical significance between either pure (objective or NS) search methods and the hybrid method (see figure 5.1), the average (mean) complexity for hybrid controllers at 7.5 was lower than controllers yielded by the objective-based search (11.25) and NS (8.55)

Given that network complexity results for the hybrid search method and other methods were not statistically significant, further research will need to be done to ascertain if network complexity may be the cause of under-performance of the hybrid approach when compared with pure fitness and NS based controllers.

Limiting analysis to controllers which were to achieve desired task performance, that is, controllers that were able to achieve at least 80% task performance (over 20 runs, objective-based search yielded 9 such controllers, NS yielded 3 controllers and hybrid yielded 19 controllers), we are able to see a larger difference in controller complexity. Figure 5.2 presents the top three controllers from each method and it is clear from these that the controllers from the hybrid approach were far simpler than either fitness or NS.

A possible explanation may be that the hybrid was able to cover the behaviour space quickly given its NS component however may have over-specialised to the training track too aggressively, simplifying controllers along the way. The hybrid approach we implemented equally weighted the novelty score and fitness components equally (see section 3.2.3). As shown in previous research (Huang et al., 2015), varying the weighting biases the search to either NS or fitness. In our study, this may have yielded different results for hybrid controllers on unseen tracks and is an avenue for future research.

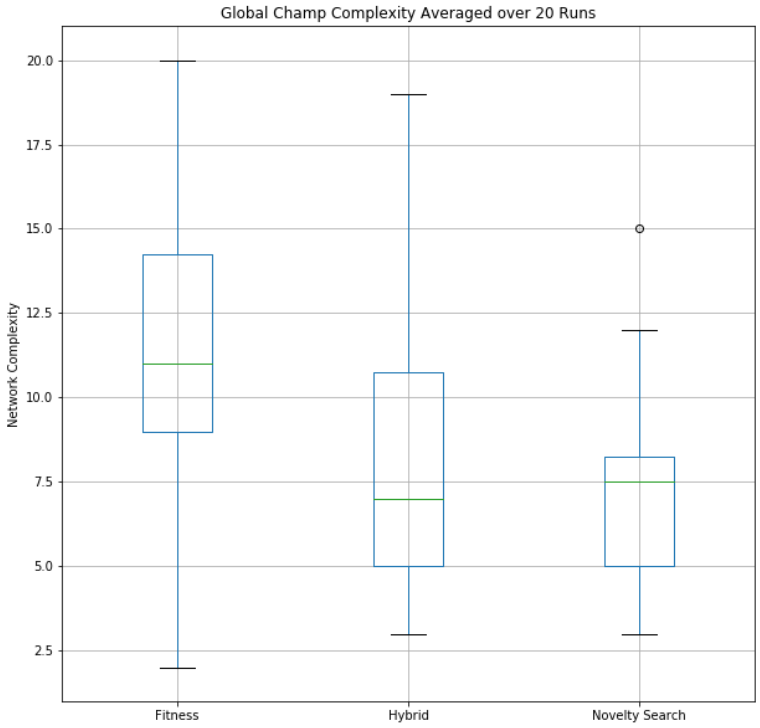


Fig. 5.1: Network Complexity of controllers: Average network complexity of champions at the final generation averaged over 20 runs. *Mann-Whitney U*, $p \leq 0.05$ statistical tests indicated statistical significance between search methods [fitness, novelty search] but no statistical significance between search methods [fitness, hybrid] or [hybrid, novelty search]

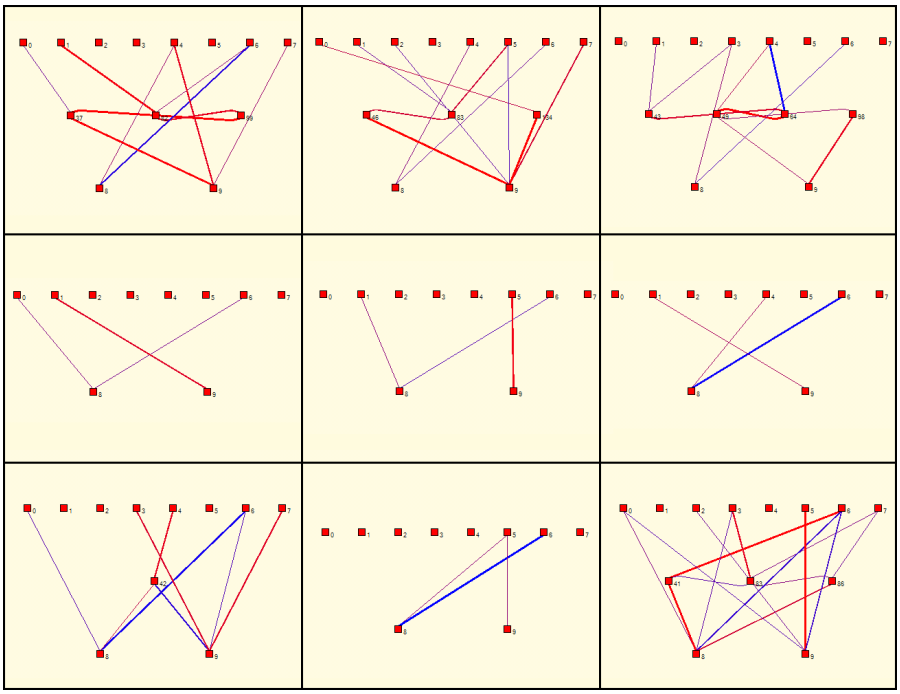


Fig. 5.2: **Controller Networks:** Top three networks from each search method. Networks displayed are selected for highest task performance and lowest complexity. **Top:** Fitness, **Middle:** Hybrid. **Bottom:** Novelty Search.

6 Conclusion

This final chapter summarises our findings and how they relate to our original hypothesis and concludes with known limitations and future work.

6.1 Summary of Findings and Results

This work aimed at applying Neuro-Evolution (NE) to the collective self-driving task (chapter 4), an area of vast scientific and commercial interest as it may present the next frontier of personal transportation systems.

We examined three different search methodologies for a popular and well-researched NE framework, NEAT. Our research supported previous work with consistent results where hybrid approaches are able to outperform pure objective and NS approaches in terms of task performance and evolution speed.

Furthermore, we presented generalisability experiments that aimed at contributing to the literature in terms of showing how each approach, Objective-based, NS and Hybrid perform on unseen task environments. Based on related work (Huang et al., 2015; Inden et al., 2013; Cuccu and Gomez, 2011), our hypothesis was that the hybrid approach will not only outperform both pure approaches in the *training* task, but also unseen tasks.

However, our results did not support this hypothesis and hybrid ultimately were outperformed by objective-based search and NS. This result contrasted the hybrid method's ability to outperform pure objective-based and NS methods, highlighting hybrid's inability to generalise to unseen environments.

6.2 Known Limitations

The track widths in the simulator were significantly wider than real-world roads. Each tracks' width was at least 15 vehicles wide which is unrealistic. This was due to controllers unable to adapt when track widths were narrower which would cause sensors to detect obstacles constantly.

Sensors also did not distinguish between detected object *types*. All objects that it detected (walls, static obstacles, other vehicles in its group, dynamic obstacles) were all treated the same way.

Since the unity game engine produces a physically realistic simulation, randomness is part of the simulation and our NE task trials were only run once per genotype per generation. This means that randomness could cause some genotypes to have higher fitness than others in the NE adaptation phase which we did not average over multiple runs (this was done in the evaluations, however).

6.3 Future Work

Given our simple (equal weighting) implementation of our hybrid approach, testing various degrees of weighting between fitness and novelty may yield significantly different results.

Improving the simulator with sensors which consider object types and narrower roads could also affect task performance and adapting controllers on more than just one track and varying the environment or vehicle sizes could possibly yield different results since the given track may have biased a specific search methodology.

Implementing the three different approaches in related collective behaviour tasks (such as, swarm robotics) may yield different results, especially our result where hybrid doesn't generalise over a range of unknown tasks and may indicate domain-specific reasons relating to poor task generalisability.

Finally, given our observed results, introducing a more robust evolution (training) track by introducing more noise (beyond traffic randomness) during evolution may yield different results by mitigating the risk of the various search approaches to overspecialising to the training track.

Bibliography

Angeline, Peter J, Gregory M Saunders, and Jordan B Pollack (1994). „An evolutionary algorithm that constructs recurrent neural networks“. In: *IEEE transactions on Neural Networks* 5.1, pp. 54–65 (cit. on p. 15).

Applegate, David L, Robert E Bixby, Vasek Chvatal, and William J Cook (2006). *The traveling salesman problem: a computational study*. Princeton university press (cit. on p. 11).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). „Neural machine translation by jointly learning to align and translate“. In: *arXiv preprint arXiv:1409.0473* (cit. on p. 9).

Baldi, Pierre (1995). „Gradient descent learning algorithm overview: A general dynamical systems perspective“. In: *IEEE Transactions on Neural Networks* 6.1, pp. 182–195 (cit. on p. 9).

Bamonte, T. (2013). „Autonomous Vehicles: Drivers for Change“. In: *Roads and Bridges* Summer, pp. 5–10 (cit. on p. 25).

Beeson, Patrick, Jack O'Quin, Bartley Gillan, et al. (2008). „Multiagent Interactions in Urban Driving“. In: *Journal of Physical Agents* 2.1. Special issue on Multi-Robot Systems, pp. 15–30 (cit. on p. 25).

Beni, G. (2004). „From Swarm Intelligence to Swarm Robotics“. In: *Proceedings of the International Workshop on Swarm Robotics*, pp. 1–9 (cit. on p. 27).

Bilger, B. (2013). „Auto Correct: Has The Self-Driving Car At Last Arrived“. In: *New Yorker* 25 November (cit. on p. 25).

Bryant, Bobby D and Risto Miikkulainen (2003). „Neuroevolution for adaptive teams“. In: *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. Vol. 3. IEEE, pp. 2194–2201 (cit. on p. 3).

Cardamone, Luigi, Daniele Loiacono, and Pier Luca Lanzi (2010). „Learning to drive in the open racing car simulator using online neuroevolution“. In: *Computational Intelligence and AI in Games, IEEE Transactions on* 2.3, pp. 176–190 (cit. on pp. 20, 25–27, 38).

Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber (2012). „Multi-column deep neural networks for image classification“. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3642–3649 (cit. on p. 9).

Clune, Jeff, Kenneth O Stanley, Robert T Pennock, and Charles Ofria (2011). „On the performance of indirect encoding across the continuum of regularity“. In: *IEEE Transactions on Evolutionary Computation* 15.3, pp. 346–367 (cit. on p. 12).

Cuccu, G. and F. Gomez (2011). „When novelty is not enough“. In: *Applications of Evolutionary Computation*. Springer, pp. 234–243 (cit. on pp. 24, 25, 58, 63).

Drchal, Jan, Jan Koutník, et al. (2009). „HyperNEAT controlled robots learn how to drive on roads in simulated environment“. In: *Evolutionary Computation, 2009. CEC'09. IEEE Congress on Evolutionary Computation*. IEEE, pp. 1087–1092 (cit. on pp. 3, 25–27, 38).

Ebner, M. and T. Tiede (2009). „Evolving driving controllers using Genetic Programming“. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Milano, Italy: IEEE Press, pp. 279–286 (cit. on pp. 3, 25–27).

Eiben, Agoston E and James E Smith (2003). *Introduction to evolutionary computing*. Springer Science & Business Media (cit. on pp. 10, 11).

Floreano, Dario and Claudio Mattiussi (2008). *Bio-inspired artificial intelligence: theories, methods, and technologies*. MIT press (cit. on p. 3).

Fogel, David B and Lawrence J Fogel (1995). „An introduction to evolutionary programming“. In: *European Conference on Artificial Evolution*. Springer, pp. 21–33 (cit. on p. 10).

Furda, Andrei and Ljubo Vlacic (2011). „Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making“. In: *Intelligent Transportation Systems Magazine, IEEE* 3.1, pp. 4–17 (cit. on p. 25).

Goldberg, David E and John H Holland (1988). „Genetic algorithms and machine learning“. In: *Machine learning* 3.2, pp. 95–99 (cit. on p. 10).

Gomes, Jorge, Paulo Urbano, and Anders Lyhne Christensen (2012). „Introducing novelty search in evolutionary swarm robotics“. In: *International Conference on Swarm Intelligence*. Springer, pp. 85–96 (cit. on p. 58).

– (Jan. 1, 2013a). „Evolution of swarm robotics systems with novelty search“. In: *Swarm Intelligence* 7.2–3, pp. 115–144. published (cit. on p. 25).

– (2013b). „Evolution of swarm robotics systems with novelty search“. In: *Swarm Intelligence* 7.2-3, pp. 115–144 (cit. on pp. 58, 59).

Gomes, Jorge, Pedro Mariano, and Anders Lyhne Christensen (2015). „Devising effective novelty search algorithms: A comprehensive empirical study“. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, pp. 943–950 (cit. on p. 58).

Gomes, Jorge, Miguel Duarte, Pedro Mariano, and Anders Lyhne Christensen (Sept. 1, 2016). „Cooperative Coevolution of Control for a Real Multirobot System“. In: *Parallel Problem Solving from Nature – PPSN XIV*. Springer, pp. 591–601. published (cit. on p. 25).

Gomez, F., J. Schmidhuber, and R. Miikkulainen (2006). „Efficient non-linear control through neuroevolution“. In: *Machine Learning: European Conference on Machine Learning*. Springer, pp. 654–662 (cit. on p. 3).

Gomez, Faustino and Risto Miikkulainen (1997). „Incremental evolution of complex general behavior“. In: *Adaptive Behavior* 5.3-4, pp. 317–342 (cit. on p. 13).

Gomez, Faustino, Jürgen Schmidhuber, and Risto Miikkulainen (2008). „Accelerated neural evolution through cooperatively coevolved synapses“. In: *The Journal of Machine Learning Research* 9, pp. 937–965 (cit. on pp. 12, 13, 24).

Gomez, Faustino J and Risto Miikkulainen (1999). „Solving non-Markovian control tasks with neuroevolution“. In: *IJCAI*. Vol. 99, pp. 1356–1361 (cit. on p. 24).

Gomez, Faustino John (2003). „Robust non-linear control through neuroevolution“. PhD thesis (cit. on pp. 7, 12, 13).

Gould, Stephen Jay (1996). „Full House: the Spread of Excellence from Plato to Darwin, 1996“. In: *See also excerpts of Stephen Jay Gould on Stanford Presidential Lectures in the Humanities and Arts*, p. 197 (cit. on p. 21).

Gower, John Clifford (1982). „Euclidean distance geometry“. In: *Math. Sci* 7.1, pp. 1–14 (cit. on p. 33).

Green, C (2003). *SharpNEAT homepage* (cit. on p. 31).

Grefenstette, John, Rajeev Gopal, Brian Rosmaita, and Dirk Van Gucht (1985). „Genetic algorithms for the traveling salesman problem“. In: *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, pp. 160–168 (cit. on p. 11).

Gruau, Frederic (1993). „Genetic synthesis of modular neural networks“. In: *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., pp. 318–325 (cit. on pp. 12–15).

Huang, Chien-Lun Allen, Geoff Nitschke, and David Shorten (2015). „Searching for novelty in pole balancing“. In: *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, pp. 1792–1798 (cit. on pp. 3, 24, 25, 34, 35, 57, 59, 63).

Igel, Christian (2003). „Neuroevolution for reinforcement learning using evolution strategies“. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* Vol. 4. IEEE, pp. 2588–2595 (cit. on p. 24).

Inden, Benjamin, Yaochu Jin, Robert Haschke, Helge Ritter, and Bernhard Sendhoff (2013). „An examination of different fitness and novelty based selection methods for the evolution of neural networks“. In: *Soft Computing* 17.5, pp. 753–767 (cit. on pp. 24, 25, 58, 63).

Jallov, Daniel Ingmer (2014). „Evolve-Introducing a Novel Game Mechanic Based on the Indirect Control Of Evolving Neural Networks“. In: (cit. on pp. 20, 31).

Kenwell, Bret (2018). *Will Tesla Be the First Automaker to Offer Fully-Autonomous Driving?* (Cit. on p. 28).

Kesting, Arne, Martin Treiber, and Dirk Helbing (2010). „Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity“. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 368.1928, pp. 4585–4605 (cit. on p. 25).

Koza, John R (1994). „Genetic programming as a means for programming computers by natural selection“. In: *Statistics and computing* 4.2, pp. 87–112 (cit. on p. 11).

KPMG, Center for Automotive Research (2012). *Self-Driving Cars: The Next Revolution*. https://www.kpmg.com/US/en/IssuesAndInsights/ArticlesPublications/Documents/self-driving-cars-next-revolution.pdf (cit. on p. 25).

Krishna (2018). *Introduction to Exponential Linear Unit* (cit. on p. 10).

Kube, R. and H. Zhang (1994). „Stagnation recovery behaviours for collective robotics“. In: *IEEE/RSJ IROS*. Cambridge, USA: MIT Press, pp. 1883–1890 (cit. on p. 27).

Lehman, Joel and Kenneth O Stanley (2011). „Abandoning objectives: Evolution through the search for novelty alone“. In: *Evolutionary computation* 19.2, pp. 189–223 (cit. on pp. 1, 3, 21–23, 47, 58, 59).

Levinson, Jesse, Jake Askeland, Jan Becker, et al. (2011). „Towards fully autonomous driving: Systems and algorithms“. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, pp. 163–168 (cit. on p. 2).

Martinoli, A., Y. Zhang, P. Prakash, E. Antonsson, and R. Olney (2002). „Towards Evolutionary Design of Intelligent Transportation Systems.“ In: *Eleventh International Symposium on New Technologies for Advanced Driver Assistance Systems*. Siena, Italy.: ATA Press, pp. 283–290 (cit. on p. 1).

Miconi, Thomas (2008). „Evolution and complexity: The double-edged sword“. In: *Artificial life* 14.3, pp. 325–344 (cit. on p. 21).

Moriarty, David E and Risto Mikkulainen (1996). „Efficient reinforcement learning through symbiotic evolution“. In: *Machine learning* 22.1-3, pp. 11–32 (cit. on pp. 12, 13).

Motavalli, J. (2012). „Self-Driving Cars Will Take Over By 2040“. In: *Forbes Magazine* 25 September (cit. on p. 25).

Mouret, J-B and Stéphane Doncieux (2012). „Encouraging behavioral diversity in evolutionary robotics: An empirical study“. In: *Evolutionary computation* 20.1, pp. 91–133 (cit. on p. 25).

NHTSA (2016). *Electronic Stability Control (ESC)* (cit. on p. 28).

Nitschke, Geoff and Sabre Didi (2017). „Evolutionary policy transfer and search methods for boosting behavior quality: Robocup keep-away case study“. In: *Frontiers in Robotics and AI* 4, p. 62 (cit. on pp. 25, 58).

Parker, Aashiq and Geoff Nitschke (2017). „Autonomous intersection driving with neuroevolution“. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, pp. 133–134 (cit. on p. 20).

Pathak, Deepak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell (2017). „Curiosity-driven exploration by self-supervised prediction“. In: (cit. on p. 21).

Pomerleau, Dean A (1991). „Efficient training of artificial neural networks for autonomous navigation“. In: *Neural Computation* 3.1, pp. 88–97 (cit. on p. 9).

Radding, Charles M (1982). „Homologous pairing and strand exchange in genetic recombination“. In: *Annual review of genetics* 16.1, pp. 405–437 (cit. on p. 18).

Sigmund, Karl (1995). „Games of life: explorations in ecology, evolution and behavior“. In: (cit. on p. 21).

Space-Based Positioning, Navigation National Coordination Office for and Timing (2017). *Roads and Highways* (cit. on p. 2).

Stanley, Kenneth O and Risto Miikkulainen (2002). „Evolving neural networks through augmenting topologies“. In: *Evolutionary computation* 10.2, pp. 99–127 (cit. on pp. 1, 3, 12–15, 18–20).

Stanley, Kenneth O, David B D'Ambrosio, and Jason Gauci (2009). „A hypercube-based encoding for evolving large-scale neural networks“. In: *Artificial life* 15.2, pp. 185–212 (cit. on p. 12).

Talamini, Jacopo, Giovanni Scaini, Eric Medvet, and Alberto Bartoli (2018). „Selfish vs. global behavior promotion in car controller evolution“. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, pp. 1722–1727 (cit. on pp. 25, 26).

Togelius, J. and S. Lucas (2005). „Evolving controllers for simulated car racing“. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. Edinburgh, UK: IEEE Press, pp. 1906–1913 (cit. on p. 25).

– (2006). „Evolving robust and specialized car racing skills“. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. Vancouver, Canada: IEEE Press, pp. 1187–1194 (cit. on pp. 3, 25–27, 38).

Trianni, Vito, Stefano Nolfi, and Marco Dorigo (2006). „Cooperative hole avoidance in a swarm-bot“. In: *Robotics and Autonomous Systems* 54.2, pp. 97–103 (cit. on p. 3).

Velez, Roby and Jeff Clune (2014). „Novelty search creates robots with general skills for exploration“. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, pp. 737–744 (cit. on pp. 47, 58).

Waibel, Markus, Laurent Keller, and Dario Floreano (2009). „Genetic team composition and level of selection in the evolution of cooperation“. In: *IEEE Transactions on Evolutionary Computation* 13.3, pp. 648–660 (cit. on p. 3).

Wasef, Basem (2018). *2019 Audi A8 L Review | Brilliant engineering in an unassuming wrapper* (cit. on p. 29).

Watson, J. and G. Nitschke (2015). „Evolving Robust Robot Team Morphologies for Collective Construction“. In: *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pp. 1039–1046 (cit. on pp. 20, 27).

Werfel, J. (2007). „Building Blocks for Multi-robot Construction“. In: *Distributed Autonomous Robotic Systems 6*. Tokyo, Japan: Springer, pp. 285–294 (cit. on p. 27).

Wieland, Alexis P (1991). „Evolving neural network controllers for unstable systems“. In: *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*. Vol. 2. IEEE, pp. 667–673 (cit. on pp. 11–13).

Willigen, Willem H van, Evert Haasdijk, and Leon JHM Kester (2013). „Evolving intelligent vehicle control using multi-objective neat“. In: *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2013 IEEE Symposium on*. IEEE, pp. 9–15 (cit. on p. 20).

Woolley, Brian G and Kenneth O Stanley (2011). „On the deleterious effects of a priori objectives on evolution and representation“. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, pp. 957–964 (cit. on p. 21).

Yegnanarayana, B (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd. (cit. on p. 9).

Zhang, Byoung-Tak and Heinz Muhlenbein (1993). „Evolving optimal neural networks using genetic algorithms with Occam's razor“. In: *Complex systems* 7.3, pp. 199–220 (cit. on p. 16).

Zhang, Yizhen, Alcherio Martinoli, and Erik K Antonsson (2003). „Evolutionary design of a collective sensory system“. In: *Proc. of the 2003 AAAI Spring Symposium on Computational Synthesis*. SWIS-CONF-2003-007, pp. 283–290 (cit. on p. 1).

Websites

Ackerman, Evan (2015). *Tesla Model S: Summer Software Update Will Enable Autonomous Driving*. URL: http://spectrum.ieee.org/cars-that-think/transportation/self-driving/tesla-model-s-to-combine-safety-sensors-to-go-autonomous (visited on Mar. 20, 2015) (cit. on p. 25).

Highway, National and Traffic Safety Administration (2013). *U.S. Department of Transportation Releases Policy on Automated Vehicle Development*. URL: http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development (visited on Apr. 1, 2016) (cit. on p. 28).

List of Figures

List of Tables

Declaration

I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

I have used the ..... convention for citation and referencing. Each contribution to, and quotation in this thesis, ***Neuro-Evolution Search Methodologies for Collective Self-Driving Vehicles***, from the work(s) of other people has been attributed, and has been cited and referenced.

This thesis, ***Neuro-Evolution Search Methodologies for Collective Self-Driving Vehicles***, is my own work.

I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Cape Town, October 2019

Chien-Lun (Allen) Huang