Neuroevolution for Decentralised Autonomous Intersection Management

Alexander Furman, Department of Computer Science, University of Cape Town
Supervised by Dr Geoff Nitschke

***Abstract*— This study compared two approaches to coordinated intersection traversal without traditional control mechanisms. In the first approach, a centralised system managed space and time in the intersection based on a heuristic control policy. In the second approach, *neuroevolution* evolved collision circumvention behaviour for a decentralised vehicle controller. An experimental comparison of the two approaches measured vehicle throughput and average traversal time in grid-based intersections that varied in terms of size and connected lanes. Results indicated that neuroevolution is capable of producing promisingly robust behaviours for intersection traversal, outperforming the centralised approach in the vast majority of tested scenarios, and also displaying a higher degree of transferability to different levels of task complexity.**

***Index Terms*— Machine Learning, Evolutionary Algorithms, Neuroevolution, Autonomous Vehicles, Autonomous Intersection Management, Intersection Collision Avoidance, Intelligent Transportation Systems**

I. INTRODUCTION

A new era of sophisticated, mass-market transportation technology is on the horizon. Vehicle manufacturers and technology firms are already prototyping autonomous vehicles on public roads, and analysts predict that driverless cars will standardise the vehicle industry in a matter of decades. For the most part, the problem of open-road autonomous driving where traffic and obstructions are sparse has been solved; testimony to this are the results of the 2005 DARPA Grand Challenge in which *Stanley*, a modified Volkswagen Touareg, successfully traversed a windy and narrow mountain path in the Mojave Desert [16]. Of course, there remains the entirely separate issue of driving in urban settings with dense traffic, pedestrians, control mechanisms, and countless unexpected obstructions. The 2007 DARPA Urban Challenge, which took place at the Georgia Air Force Base, was a step towards the tackling of this issue, but featured a track arguably more resemblant of suburban rather than urban settings [17].

Traffic intersections are particularly onerous control mechanisms. Unlike open-road driving where the main algorithmic challenge is basic lane-following behaviour, intersections are far more volatile and demanding. Between 25% and 45% of all vehicle collisions take place at intersections - a wildly disproportionate statistic taking into account the negligible portion of the roadway they comprise.

Traditional intersection control mechanisms are psychologically associated with the message they are supposed to transmit in order to make intersection traversal as resistant to human error as possible. But, autonomous vehicles can judge velocities and distances with unprecedented accuracy, monitor multiple angles and blind-spots simultaneously, and react to emergency situations quicker than the most experienced of human drivers. Looking towards a driverless future, Dresner and Stone [1] propose a multiagent intersection control paradigm called *Autonomous Intersection Management* (AIM), designed to maximise the capabilities of autonomous vehicles rather than to minimise the consequences of human error. The model requires the installation of *arbiter agents* called *Intersection Managers* (IMs) at intersections. *Driver Agents* wirelessly request *space-time* in the intersection from the IM, which responds to requests based on the *intersection control policy* put in place. AIM guarantees that a collision will never take place in an intersection under the idealistic assumptions that there is always a stable communication link between driver agents and IMs, that vehicles never malfunction and respect their reservation statuses, that obstructions such as pedestrians are never present, that vehicles never skid, and that natural or non-natural disasters never compromise the performance of the system. Relaxing any of these assumptions invalidates all safety and efficiency guarantees. Even with a complex intersection control policy and, as suggested by Dresner and Stone [1], IM sensors to detect unusual occurrences in the intersection, the system can do little but send out messages to deal with obstructions such as pedestrians, and this still requires a stable communication link as well as policy-abiding driver agents.

Neuroevolution of Augmenting Topologies (NEAT) has been successfully applied to various complex control tasks, including ones in the domain of autonomous vehicle control [5, 9, 11, 15, 19, 20, 28]. Neuroevolution is attractive in the context of controller design for autonomous intersection management: it addresses many safety concerns that arise in a centralised system, providing a platform for decentralised autonomous intersection traversal without making unrealistic assumptions. Secondly, its highly general requirements eliminate the need for a complex hand-designed model that accounts for every possible scenario that could take place during an intersection traversal control task. The 2 main requirements are vehicle sensors for environmental interaction, and a metric for evaluating and reflecting controller task performance as a numerical value.

A. Research Objectives

The goals of this study were to establish the efficacy of using *neuroevolution* to produce collision-circumvention behaviour for a decentralised autonomous vehicle controller in the context of a simulated intersection traversal control task, and to establish whether such a controller outperforms a centralised system employing a heuristic decision-making policy. Furthermore, we aimed to establish the transferability of the two coordination methods to varying levels of complexity in the task environment, such as random obstructions (pedestrians) and a larger state space (additional lanes connecting to the intersection). Crucially, an aim of this

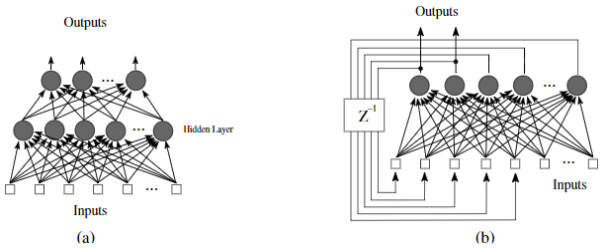


Fig. 1. **Network architecture [4].** (a) In a feedforward network, input data enters the network and activation is propagated forward to produce the output. (b) In a recurrent network, feedback connections provide the network with information from prior outputs.

study was not to evolve *driving* behaviour for autonomous vehicles, but rather to evolve collision-circumvention behaviour.

Problem statement: Is a decentralised vehicle controller evolved using neuroevolution for collision-circumvention behaviour more appropriate for autonomous vehicle coordination at intersections than a centralised reservation system employing a heuristic decision-making policy?

II. Foundations

This section provides the relevant background material for the machine learning facet of this study. Machine learning is a branch of Artificial Intelligence which deals with the design of algorithms that learn from data in order to make predictions and perform tasks without being explicitly programmed.

1) Artificial Neural Networks (ANNs): ANNs are universal and flexible function approximators that take inspiration from the biological neural networks of animal nervous systems. An ANN comprises of a set of interconnected *neurons* (computational nodes) which receive and process inputs to produce outputs. During the training of an ANN, different *weight* and *bias* values (which constrain how input data are related to output data) are explored until the output of the network sufficiently minimises some *cost function* which describes how 'far off' the network's output is from the desired result.

The artificial neuron receives input values from other artificial neurons, which are passed through a *weighted summation function* and an *activation function* before returning an output (an *activation*) that can be received by other neurons as an input. The *activation function* of an artificial neuron is an abstraction of the biological neuron's rate of action potential firing through the cell, and constrains how the neuron's inputs relate to its output. In its simplest form, an activation function is binary (ie: the *Heaviside Function)* - that is, the neuron is either firing an output or not. Due to the non-linearity of real world problems, nonlinear activation functions such as *Sigmoid* are preferred, providing a bounded range of smooth outputs that closely reflect changes to weight values in the network.

An ANN can be *feedforward*, where information flows exclusively from input to output, or *recurrent*, where feedback connections provide the network with information from previous activations. Figure 1 depicts these two architectures. ANNs are powerful modelling tools employed in a wide range of fields.

2) Traditional Control Task Learning Approaches: ANNs are commonly trained to apply known control techniques via *Supervised Learning* methods such as gradient-descent back-propagation. A drawback of such techniques is the requirement of labeled training data (correct input/output pairs). Back-propagation, for instance, compares the output of a network to the desired output from the training data, and the error (how 'far off' the output was from the correct output in the training data) is propagated backwards through the network such that its weights can iteratively be adjusted to produce a more accurate output. Within the domains of certain non-linear control tasks, the designer cannot compile training data as the control strategy is not known. Rather, it is the responsibility of the designer to *discover* a control strategy. Autonomous intersection management is a relevant example of such a domain: pre-programming a correct response to every likely situation that could take place in a busy intersection is barely feasible.

Reinforcement Learning discards the requirement of labeled training data, allowing designers to solve problems where control strategies are not well established. In this class of approaches, an *agent* learns to perform a task by interacting with the environment to maximise the *reinforcement signals* (which are correlated with the correct behaviour) it receives. Unfortunately, for real world problems where the state of the environment is not fully observable to the agent, this class of methods is often inappropriate [4]. This drawback certainly applies to the control task of intersection traversal, as (assuming technology in the present and foreseeable future) the average driver agent has considerably limited knowledge of a chaotic intersection's state space.

Unsupervised Learning presents a third class of methods to designers. In this approach, no information regarding the 'correctness' of an output is provided to the controller; the learning process relies solely on correlations among the input data.

3) Neuroevolution: Neuroevolution is an unsupervised learning technique which uses *Genetic Algorithms* (GAs) to evolve populations of ANNs in order to find networks most suitable for control tasks. GAs are a subset of the field of *Evolutionary Computation*, a family of global optimization algorithms inspired by biological evolution and natural selection. A general neuroevolution method encodes each ANN (called a *phenotype*) in a population as a binary or string-represented *genotype* (also called a *chromosome*). The nature of the genotype-phenotype mapping depends on the encoding scheme used. *Direct Encoding* creates a one-to-one mapping of each connection and node between the genotype and phenotype, while *Indirect Encoding* maps only the most important parameters such as number of hidden layers, leaving the other parameters to the training process [12]. The general process of using neuroevolution to evolve ANNs, depicted in Figure 2, is as follows:

1) Each genotype in the population, initialised as a minimal candidate solution, is encoded into its corresponding phenotype (ANN).
2) Each ANN in the current generation attempts to complete trials of the control task, accumulating a 'fitness

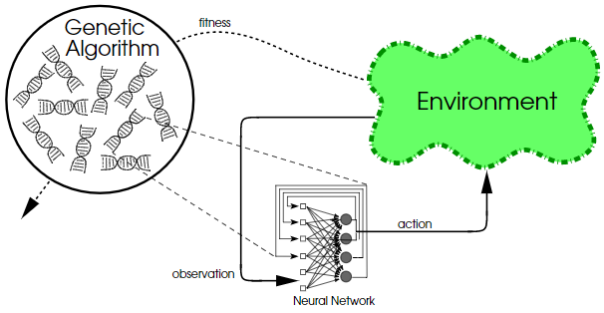


Fig. 2. **General process of neuroevolution [4].** Each genotype in the population encodes an ANN which is evaluated on the control task. The agent interacts with the environment via ANN inputs (observations) and outputs (actions). Propagation to the next generation depends on the genotype's fitness (a function of task performance).

score' based on its performance.

3) After the performance of every individual in the population has been evaluated, genetic operators are used to propagate high-fitness individuals to the next generation, replacing low-fitness individuals.
4) Steps 2 and 3 are repeated until a sufficiently fit individual is found in the search space [12].

For control tasks with unattainable training data, unknown control strategies, and sparsely observable environments, neuroevolution provides a viable alternative to supervised and reinforcement learning, requiring only a measure of an ANN controller's performance at a task.

III. Literature Review

While the specific use of neuroevolution for decentralised autonomous intersection traversal and collision avoidance seems to be unexplored, we highlight multiple studies that illustrate the applicability of neuroevolution to complex control tasks in the domain of autonomous driving, and also provide an account of decentralised methods that have been explored in the context of intersection traversal and collision avoidance thus far. For an account of *centralised* methods that have been investigated, we refer the interested reader to the *Related Work* section of [1].

Togelius [15, 18] has explored applications of neuroevolution to the domains of simulated car racing and vehicles in video games. Findings have shown that proficient racing controllers can be evolved for different types of tracks, and that evolution of sensory parameters can lead to higher fitness if evolution is allowed to build off of a general controller. Findings also suggest that neuroevolution is applicable to the design of vehicle controllers that take into account obstructions such as track edges and obstacles. Furthermore, it has been shown that pie-slice sensors are particularly useful for the detection of close-range and discrete obstacles, while rangefinder sensors are more applicable for the detection of contiguous obstacles such as track edges. We use this finding in our decentralised controller design. Togelius and Lucas [24] developed the *Simplerace* simulator to provide a platform for testing different controller development methodologies, including ones based on evolution. The simulator supports skidding and car-to-car collisions to introduce complexity to control tasks.

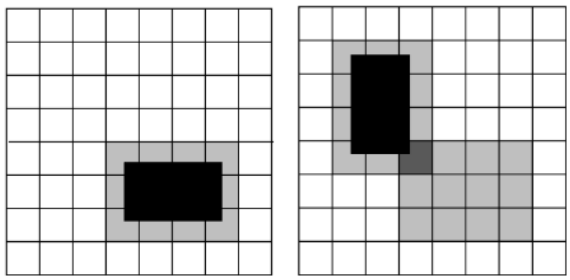
Gowrisankar [19] used NEAT to evolve simulated car-racing controllers in the Simplerace domain. While it was found that advanced driving behaviour can be evolved in this manner, a modular approach was required to evolve high-level decision making processes such as overtaking. The driving task was decomposed into *general navigation* and *waypoint selection* and evolved independently.

Wang and Schreiber [11] used NEAT to evolve a driver agent controller capable of successfully merging onto a simulated highway given any frequency of traffic, given that the traffic is constant and sufficiently sparse to accommodate the agent. Inputs to the controller were *vehicle speed, position, heading, speed of cars in traffic lane,* and *distance from front and back of agent to nearest vehicles.* Outputs were *acceleration* and *rotation*. After 100 generations with 100 individuals evaluated over 4 task trials per generation, NEAT discovered a controller with optimal performance in each simulated scenario. While this study was limited in its requirement of constant traffic, it illustrates the applicability of neuroevolution to unique driver control tasks.

Stanley et al. [9] used NEAT to evolve a reliable crash warning system for autonomous vehicles in the *RARS* Simulator, allowing driver agents to navigate challenging simulated test roads without crashing. Simulated rangefinder sensors were provided to driver agents, allowing each vehicle to project rays to the track edges and giving the controller a sense of the vehicle's position with respect to the track's curvature. After 400 generations of evolution with 100 individuals each, the controller learned behaviours such as driving in a straight line at the curviest section of a windy track. While such counter-intuitive behaviour would not come naturally to a human driver, this technique was discovered solely via evolution.

The related works mentioned thus far show that neuroevolution is capable of evolving robust behaviours that are essential for intersection traversal, such as lane, speed and route selection, as well as obstacle detection.

A number of studies have investigated approaches to decentralised intersection traversal and collision avoidance based on vehicle-to-vehicle communication technology. Hafner et al. [27] explored and validated the efficiency of decentralised algorithms for two-vehicle cooperative collision avoidance at intersections, while Farahmand [26] investigated an approach which used Extended Kalman Filtering and an intervehicle communication method, producing promising results in reducing average vehicle delay at intersections. Hall et al. [28] have explored a teamwork model, based on the platoon approach, for decentralised collaborative navigation. In this model, groups of vehicles are led by a single vehicle which acts as the 'platoon leader', making decisions about how the platoon should coordinate. We are more interested, however, in decentralised methods that are not dependent on intercommunication. Rasche and Naumann [25], for example, investigated an approach that fulfills these criteria using a decentralised semaphore-based model, but we argue that the use of semaphores naturally fails to exploit the potential of autonomous vehicle technology. It should also be noted that each of these studies make little or no mention of measures to account for obstacles such as pedestrians or other imperfect conditions, focusing near-exclusively on vehicle-vehicle colli-



(a) Successful (b) Rejected

Fig. 3. **Depiction of FCFS Reservation Policy [1].** Black rectangles represent vehicles, whilst shaded tiles are reserved. In 2(a), a vehicle is granted space-time as its simulated path does not cross any reserved paths. In 2(b), a vehicle's request is rejected as its simulated path crosses the reserved path of another vehicle (at the grey tile).

sions. Lastly, we point out that the discussed literature deals primarily with traditional intersection control mechanisms as opposed to models more suited for autonomous vehicles. This study hopes to address each of these points and bridge the literary gap between neuroevolution and decentralised autonomous intersection traversal (particularly collision avoidance).

IV. Methods

In this section, we present the two methods of producing intersection coordination behaviour that were compared in this study.

A. AIM with FCFS Intersection Control Policy

Dresner and Stone [1] propose a heuristic AIM *intersection control policy* for IMs to make decisions called *First Come, First Served (FCFS).* The process of the policy, depicted in Figure 3, is as follows:
The intersection is divided into an $n \times n$ grid of reservation tiles, where n is the granularity of the policy.

1) *Upon receiving the reservation parameters from an approaching driver agent, the pol-*
   icy runs an internal simulation of the trajectory of the vehicle across the intersection using these parameters.
2) *At each time step of the internal simulation, the policy determines which reservation tiles will be occupied by the vehicle*
3) *If at any time during the simulation the requesting vehicle occupies a reservation tile that is already reserved by another vehicle, the policy rejects the request. Otherwise, the policy accepts the reservation and reserves the appropriate tiles for the times they will be required [1].*

Since FCFS has been used to demonstrate the benefits of AIM [1], and since it seems no other relevant policies have been created, it was used as the AIM *intersection control policy* for this study.

1) The Pedestrian Heuristic: To investigate how AIM adapts to imperfect conditions, we added a heuristic to AIM's *Driver Agent* model that allowed vehicles to detect, stop and wait for pedestrians walking through the intersection, as well as for obstructing vehicles.

B. Neuroevolution of Augmenting Topologies (NEAT)

NEAT[1] was the neuroevolution method used to produce the ANN controllers. NEAT evolves both connection weights and topology, and is based on three crucial innovations: *complexification*, *speciation*, and *historical marking*.

Complexification allows NEAT to begin with a homogenous population of candidate solutions (ANNs) which increase in complexity throughout the evolutionary process. Each initial candidate solution is unique only in terms of its connection weights. Thus, NEAT starts minimally and gradually discovers the architecture of the desired solution over multiple generations.

Speciation allows individual solutions to compete within their own niches rather than against the entire population, by dividing the solutions that comprise the population into species. This allows for the protection of young topological innovations, providing time for structural optimisations before competing on the same playing field as other niches. It would be undesirable, for instance, to compare a solution that has been developing for 100 generations to a novel innovation which has emerged in the most recent generation. An immediate comparison would likely favour the older solution, failing to take into account the potential of the novel solution should it be given time to develop.

Historical marking is largely the backbone behind complexification and speciation. This provides a means of tracking genes throughout the evolutionary process to ensure that the mating operator is only applied to compatible genomes (ANN solutions). The innovation number of each genome is a historical marker that identifies the ancestral history of that genome. Innovation numbers are unaffected by genetic operators, allowing genomes to mate without the requirement of topological compatibility analyses [21].

The genetic operators used by NEAT are *Selection*, *Crossover* and *Mutation*. The Selection operator is used to select fit parent solutions for propagation to the next generation, the Crossover (or *Recombination*) operator combines portions of fit parent solutions to produce child solutions, and the Mutation operator applies subtle structural changes to solutions to encourage genetic diversity [8].

V. Simulation Environment

Version 4 of the open-source, Java-based *AIM Simulator* [2] was used as the simulation environment for the experiments. The environment was 250 x 250 metres, with a grid-based intersection in the centre of the area. The size of the intersection depended on the number of lanes connected to it, which was set prior to simulations.

A. Driver Agents

The simulator provides each vehicle with general lane following behaviour that is independent of the behaviour determined by the vehicle's communication with the intersection manager. This lane following behaviour is active at

[1]The NEAT Users Page can be found at http://www.cs.ucf.edu/ kstanley/neat.html

[2]The AIM4 homepage can be found at http://www.cs.utexas.edu/ aim/

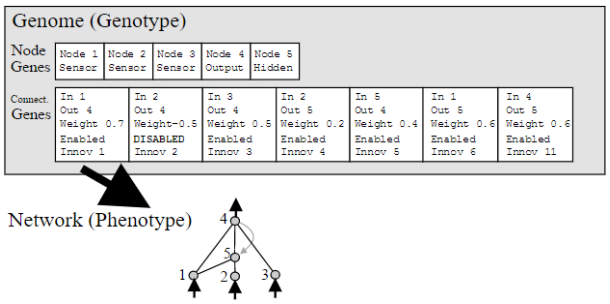


Fig. 4. **A NEAT genotype to phenotype mapping [8].** During evolution, the genotype is converted to the phenotype to be evaluated on the control task. This genome has 3 input nodes and 1 output node, with one of the 7 connections being recurrent. The connection between nodes 2 and 4 is disabled in the genotype and thus does not appear in the phenotype.

all times, and was used throughout the experiments to steer vehicles until encountering an obstruction, at which point the NEAT controller activates to circumvent impending collision. The general lane following behaviour features a number of heuristics to keep vehicles centred in their correct lanes. If the NEAT controller careens a vehicle to the other side of the intersection to dodge an obstruction, the general lane following behaviour will navigate the vehicle back to the lane from which it deviated via the shortest route possible, using the *A* search algorithm*. The simulator accounts for the technological constraints of vehicles in the real world (including in the foreseeable future), requiring driver agents to abide by maximum and minimum acceleration/deceleration and velocity limits which are set prior to simulations. Vehicles are spawned probabilistically to simulate real world traffic, and the wait period before a new vehicle is spawned (traffic level) can be set.

VI. Experiment Setup

A. Extensions to Simulator

1) Decentralised Intersections: The simulator does not by default support the running of simulations without communication between vehicles and intersection managers - a necessary requirement for a decentralised coordination approach. Thus, the first modification was to allow for the dismantling of intersection managers in a given simulation.

2) Pedestrians: A pedestrian class was written, allowing for the spawning of pedestrians that walk back and forth between random points in the intersection throughout a simulation. The speed, size, number of pedestrians and waypoints walked between can be specified prior to simulations.

3) Collision Detection: Collision detection was implemented such that a collision is said to have taken place in a simulation if (a) two vehicles are intersecting at a given timestep, (b) a vehicle veers out of the intersection without taking an exit lane, or (c) a vehicle is intersecting a pedestrian at a given timestep, but only if the current velocity of the vehicle is greater than 0.3 mph, allowing pedestrians to 'walk over' stationary cars that are waiting for obstructions to move away without registering as a collision. Crashed vehicles are removed from the simulation.

TABLE I
NEAT Network controller Inputs and Outputs

Input	Normalised range
Front sensor distance to obstruction	[0,1]
Left sensor distance to obstruction	[0,1]
Right sensor distance to obstruction	[0,1]
Angle to obstruction	[-1,1]
Distance to inside border of intersection	[0,1]
Output	**Normalised range**
Steering	[-1,1]
Virtual gas pedal	[-1,1]

4) Vehicle Sensors: Sensors provide vehicles with a pie slice field of view that can be adjusted prior to simulations. If multiple obstructions appear in a vehicle's field of view, the sensor can return the distance from the starting point of the FOV to the nearest obstruction, as well as the angle to that obstruction.

5) The Pedestrian Heuristic: To enable AIM vehicles to stop and wait for pedestrians (and obstructing vehicles), each was provided a 30 degree front sensor. An important implementation detail was to preserve AIM's usual performance in scenarios devoid of unexpected obstructions, without triggering the heuristic. As such, the sensors provided view fields that were sufficiently wide for pedestrian or vehicle detection under imperfect conditions, but too narrow to detect other vehicles under perfect circumstances. Figure 5(b) depicts the AIM sensory configuration.

B. NEAT Implementation

1) Sensors: NEAT driver agents had 3 sensors, with each providing a pie-slice field of view spanning 90 degrees. The front sensor protrudes from the centre-front point of the vehicle, while the left and right sensors protrude from the centre-left and centre-right points of the vehicle, respectively. The sensory configuration for NEAT vehicles is provided in Figure 5(a).

2) Controller Design: An ANN controller maps 5 sensory inputs to 2 motor outputs. The inputs from the front sensor are *Angle to nearest obstruction*, *Distance to nearest obstruction*, and *Distance to border of intersection*. *Angle to nearest obstruction* is the angle from the starting point of the front sensor field of view to the nearest point on the nearest detected obstruction, and is normalised within the range [-1,1]. *Distance to nearest obstruction* is the distance from the starting point of the front sensor field of view to the nearest point on the nearest detected obstruction, and is normalised within the range [0,1], where 0 is the vehicle touching the obstruction and 1 is the obstruction on the arc of the field of view. *Distance to edge of intersection* is the distance from the starting point of the front sensor field of view to the nearest point on the inside border of the intersection, and served to prevent vehicles from straying illegally out of the intersection. This input is normalised within the range [0,1], where 0 is the vehicle touching the border and 1 is the vehicle detecting the border at the arc of the field of view. The last 2 inputs are *Distance to nearest obstruction in left sensor field of view* and *Distance to nearest obstruction in right sensor field of view*, following the same design scheme

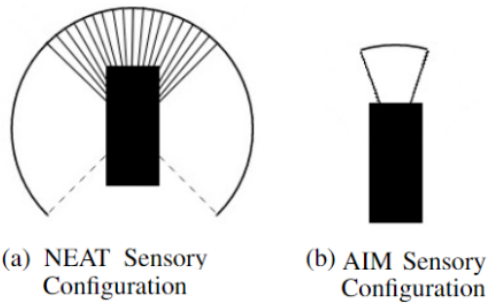


Fig. 5. **Sensory configurations.** The black rectangles are vehicles, and each circular sector corresponds to the field of view of a sensor. (a) NEAT vehicles have 90-degree left, right, and front sensors. Each of these sensors returns the distance to the nearest obstruction, with the front sensor also returning angle to obstruction and distance to inside border of intersection. (b) AIM vehicles have 30-degree front sensors for detecting the presence of obstructions.

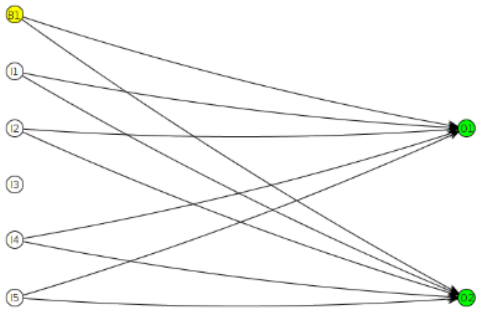


Fig. 6. **Feedforward neural network controller encoded by the best performing genome.** The 5 white nodes are environmental inputs to the network, the 2 green nodes are actuator outputs, and the yellow node is the bias. The third input, *distance to inside border of intersection*, was not needed for this network to perform well, and is thus not connected to the output nodes. Each of the other 4 input nodes are connected to both output nodes.

as the front sensor distance input, but measuring from the respective starting points of the side sensor view fields.

The outputs were *Steering* and *Virtual gas pedal*. The steering output was implemented by partitioning the front sensor field of view into sectors of 5-degrees each (refer to Figure 5(a)). Any point on the arc of the field of view at which a new five-degree segment begins was a point that the NEAT controller could angle the steering wheel towards. Each of these points was added to a list of possible steering outputs, and the indexes of the points in the list were normalised within the range [-1,1] such that -1 and +1 correspond to full left and full right, respectively. The gas pedal output allowed the ANN controller to accelerate and decelerate, and is normalised within the range [-1,1] where -1 is full deceleration, 0 is no gas, and +1 is full acceleration.

A crucial design element of the NEAT controller is that it activates only when obstructions are detected by vehicle sensors, otherwise the vehicle is controlled by the simulator's heuristic driving behaviour. If a vehicle is 10 metres outside of an intersection and detects an obstruction inside of the intersection, the controller will activate, but will only have access to the *Virtual gas pedal* actuator. This measure was put in place to prevent the NEAT controller from steering the vehicle into the wrong lane to dodge the obstruction before even entering the intersection. If at least the front half of a vehicle is inside an intersection and an obstruction is detected, that vehicle's ANN controller will have access to both the *Steering* and *Virtual gas pedal* actuators.

3) Evolving Controllers: The NEAT parameters were chosen via systematic experimental search. The best performing controller was evolved with a population of 150 genomes (ANNs) over 75 generations. The full list of parameters is provided in Appendix I. Per generation, each genome's fitness was established by its average fitness score over 10 intersection traversal task trials. In each task trial, 30 vehicles which were assigned random spawn and destination roads were given 50 timesteps to traverse the intersection. For a vehicle to traverse the intersection successfully, it had to pass a checkpoint placed 10 metres before the entrance to the intersection, drive through without crashing, and finally pass a checkpoint placed 10 metres after the intersection exit that corresponds to its randomly assigned destination road. A vehicle may exit onto any lane, as long as that lane lies on the destination road it was assigned to. The fitness function was defined as follows:

$$F = (S * 0.05) - (V * 0.05) - (P * 0.05) - (I * 0.025)$$

where F is the fitness score describing a network's performance, S is the number of successful traversals, V is the number of *vehicle-vehicle* collisions, P is the number of *vehicle-pedestrian* collisions, and I is the number of vehicles that crash as a result of straying out of the intersection without taking an exit lane. As NEAT lacks support for negative fitness scores, each fitness score F was passed through the exponential function with base 2 to ensure NEAT only received positive fitness values.

Encog-NEAT, available as part of the Java-based *Encog Machine Learning Framework*[3], was used to evolve the ANN controllers for the experiments.

VII. Results and Discussion

The experimental comparison evaluated intersection coordination task performance in 12 unique scenarios, each featuring a different number of lanes connecting to the intersection and a different number of pedestrians. The number of vehicles per scenario was chosen such that each scenario featured the same proportion of traffic relative to the number of lanes (as intersection size is a function of number of lanes per road). This was necessary as, for instance, it would take longer for 40 vehicles to traverse a 1-lane intersection than 40 vehicles to traverse a 3-lane intersection due to the increased intersection size. Task performance was based on average intersection throughput and average number of timesteps taken for each vehicle to traverse the intersection in a given trial. Throughput was defined as the ratio of successfully traversed vehicles to the number of vehicles that were spawned in the given task trial, and a successful traversal followed the same definition as that which was used during controller evolution (*ie:* both

[3]The Encog project homepage can be found at http://www.heatonresearch.com/encog/

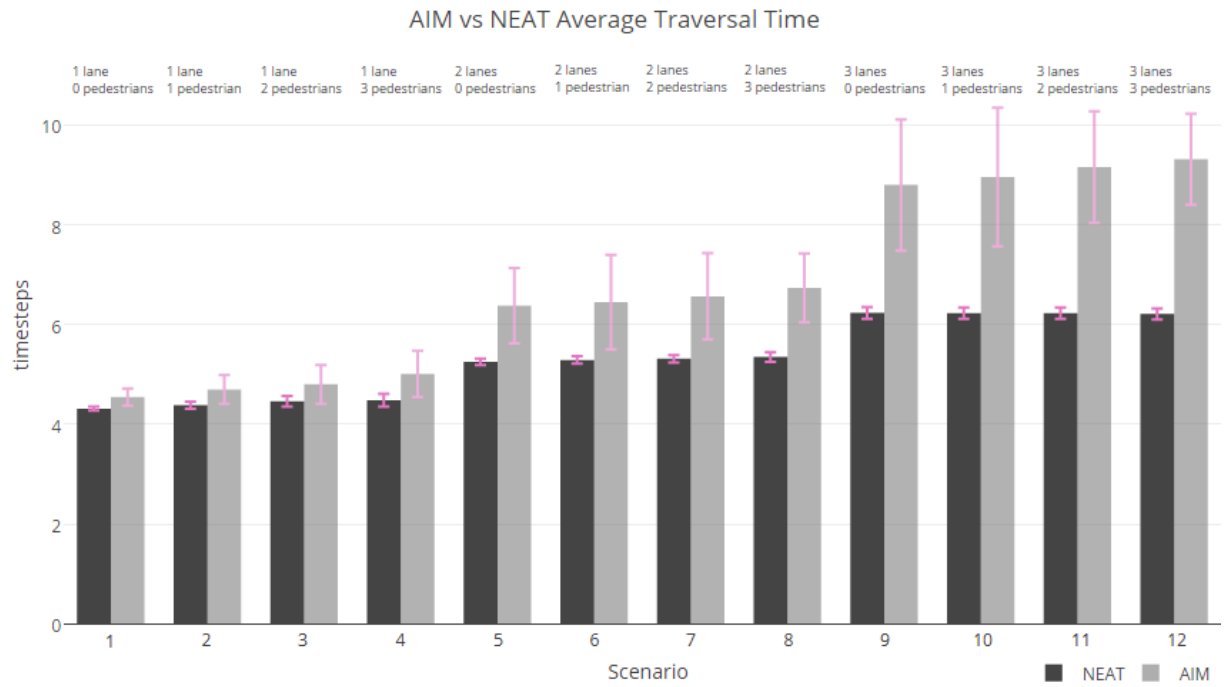


Fig. 7. **AIM vs NEAT Average Vehicle Traversal Time.** Each bar shows the average number of timesteps taken for either a NEAT or AIM coordinated vehicle to successfully traverse the intersection in a given scenario, averaged over 100 90-timestep trial runs. Error bars show Standard Deviations of performance over the 100 trials of each scenario.

checkpoints passed without crashing inside the intersection). Figures 7 and 8 provide average traversal time and average throughput, respectively, for AIM and NEAT in each of the 12 scenarios. The methods were evaluated on each of the 12 task scenarios 100 times, where each task trial lasted 90 simulated time steps. A controller's average performance over the 100 trials of a given scenario was used to determine its average throughput and traversal time statistics for that scenario. Figures 7 and 8 also include *Standard Deviation* (SD) error bars for both AIM and NEAT's 100 trial results for each scenario.

A two-tailed t-test was run for each of the scenarios for both traversal time and vehicle throughput, in order to examine whether there was a difference between NEAT and AIM performance in each of the scenarios. Results yielded sufficiently strong evidence to suggest that, for average performance in each task scenario, there was a significant difference between AIM and NEAT for both *average traversal time* and *average throughput* at the 1% significance level ($p < 0.01$ in all cases).

We first analyse average traversal times, shown in Figure 7, of the two methods. An initial observation is that average traversal time of both methods increases as the level of complexity increases (where *complexity* is defined by number of lanes and number of pedestrians). This is illustrated clearly in Figure 7, where we see an increase in timesteps as we observe each additional scenario.

Further observation reveals that the variables *number of pedestrians* and *number of lanes* independently affect average traversal time. Again referring to Figure 7, we concentrate on specific scenarios in order to keep either one of the variables constant, allowing us to observe the effect of increasing the other. Keeping number of pedestrians constant at 0 by focusing on scenarios 1, 5 and 9, we see that adding lanes increases the average traversal time for both methods. A similar trend can be observed by keeping number of pedestrians constant at 1 (Scenarios 2, 6 and 10), 2 or 3. This finding is unsurprising, given that intersection size is a function of number of lanes, and that larger environments naturally take longer to traverse. Similarly, if number of lanes is kept constant at 1 (Scenarios 1, 2, 3 and 4), we see that adding pedestrians increases average traversal time for both methods. This same trend appears when keeping number of lanes constant at 2 or 3.

Comparing the performance of the individual methods in each scenario yields additional insights regarding average traversal time. Figure 7 shows that NEAT provides faster average intersection traversal than AIM in every scenario - even in those that satisfy AIM's perfect-world assumptions by leaving out pedestrians. Furthermore, given a constant number of lanes, it can be observed that adding pedestrians causes notable growth in AIM's average traversal time, but influences NEAT's negligibly. These findings are plausible: The NEAT controller utilised evolved behaviours that favoured steering and, to a lesser extent, deceleration to avoid collisions rather than depleting time by stopping altogether. AIM, on the other hand, requires vehicles awaiting reservations to stop and wait outside the intersection, causing congestion and depleting time. Additionally, the pedestrian heuristic, by design, causes even more waiting time. One might argue that the AIM pedestrian heuristic we implemented is a confounding variable which is the actual reason for AIM taking longer than

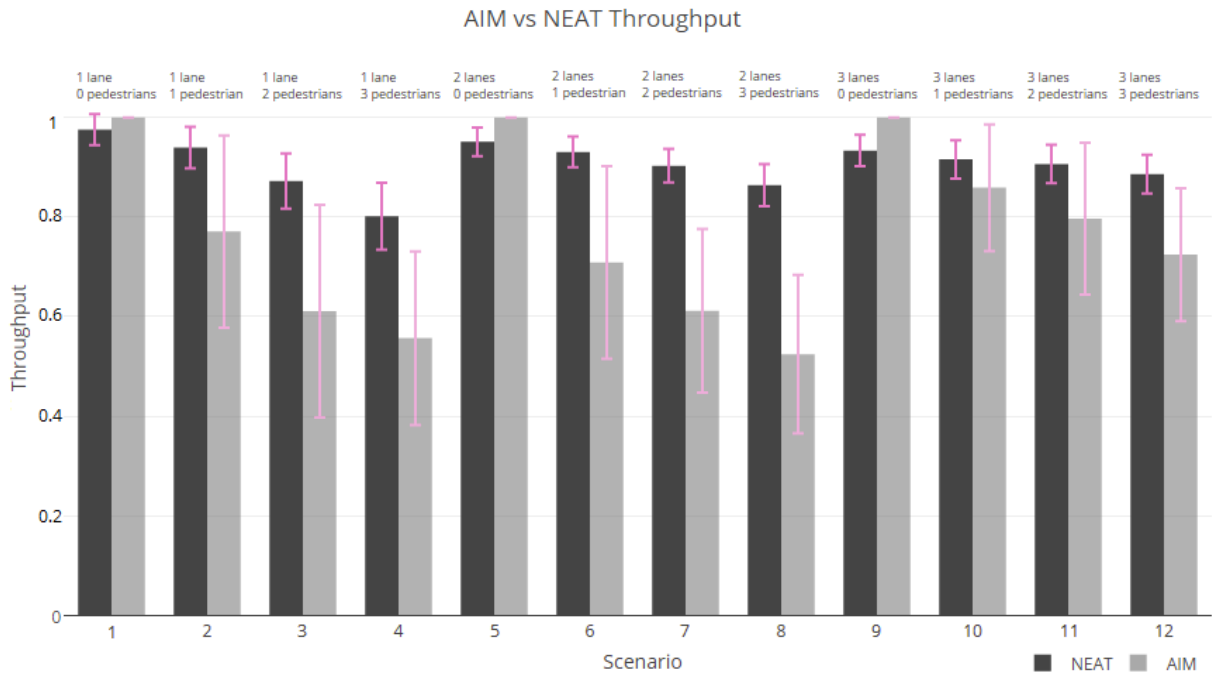


Fig. 8. **AIM vs NEAT Average Intersection Throughput.** Each bar shows the percentage intersection throughput of vehicles in a given scenario where either NEAT or AIM was the controller, averaged over 100 90-timestep trial runs. Error bars show Standard Deviations of performance over the 100 trials of each scenario.

NEAT, rather than the reservation system. Such an argument might suggest that the heuristic's mechanism, which AIM's designers may have implemented differently, increases vehicle waiting time to the extent that AIM is slower than NEAT in every scenario, deceivingly making it look as though the reservation system is to blame. This is, however, not true for the following reason: Even if the perfect AIM heuristic had been implemented and we assume that the reservation system is not to blame for AIM's comparatively lower performance, it would be unreasonable to suggest that AIM's average traversal time given some number of pedestrians could be any lower than its average traversal time under perfect conditions (*ie:* keeping number of pedestrians constant at 0). We then notice in Figure 7 that in each scenario satisfying AIM's perfect-world assumptions (Scenarios 1, 5 and 9), NEAT's average traversal time is lower than AIM's. It is thus clear that, while the mechanism of our heuristic could possibly exaggerate AIM's comparatively slower performance, NEAT will still always require fewer timesteps regardless of additional pedestrians or the heuristic in place.

If we keep number of pedestrians constant at any value with lanes variable, we see in Figure 7 that AIM's average traversal time increases at a significantly faster rate than NEAT's as additional lanes are added to the intersection, illustrating NEAT's superior degree of transferability to larger environments in terms of task speed.

We now analyse average vehicle throughput, shown in Figure 8, of the two methods. Immediately, we observe that performance of the two methods is not, in general, an obvious function of scenario complexity, as in the average traversal time analysis. If, for either method, we keep number of lanes constant at any value and pedestrians variable, we see that adding lanes reduces throughput for both methods. Furthermore, the rate at which NEAT's average throughput decreases as a result of this is significantly lower than that of AIM, indicating that, in terms of throughput, NEAT copes with additional pedestrians more robustly than AIM. In contrast, we find that keeping number of pedestrians constant with lanes variable has little effect on average throughput for both methods, indicating high transferability of both methods to larger task environments in terms of throughput. With this in mind, there seems to be a higher degree of stability in the rate at which NEAT's average throughput changes with respect to number of lanes, whereas AIM's performance is noticeably more inconsistent. We believe this inconsistency is linked to a high degree of variation in AIM's results, which arose as a consequence of the method's poor transferability to changing scenarios.

Observing the SD error bars for AIM's average traversal time and average throughput results in Figures 7 and 8 respectively, we notice a significantly higher degree of variation than in the NEAT results. Error bars in Figure 7 suggest that variation in AIM's average traversal time increased with added complexity (changing scenarios), while there was a remarkably higher degree of consistency in NEAT's results. Keeping either number of lanes or number of pedestrians constant produces this same trend: for instance, keeping number of pedestrians at 0 by only looking at scenarios 1, 5 and 9 shows that AIM's SD increases from *SD=0.168*, to *SD=0.389*, to *SD=0.690*, while NEAT's SD merely changes from *SD=0.0414*, to *SD=0.0626*,

to *SD=0.119* for these scenarios. A similar observation can be made by keeping number of lanes constant, again showing significantly higher AIM SDs than that of NEAT.

Focusing on average throughput results in Figure 8, we once again observe high AIM SDs and comparatively low NEAT SDs. But, AIM's average throughput SDs do not seem to follow any sort of pattern, unlike for the case of AIM's average traversal time SDs. The trend of variation in AIM's average throughput seems to be binary: under perfect conditions (*ie:* keeping number of pedestrians constant), there is no variation (*SD=0)* as AIM is guaranteed 100% throughput in each trial, but with any number of pedestrians, SD spikes to some value between *SD=10* and *SD=20*. These results support the claim that the centralised AIM paradigm generalises poorly to imperfect conditions.

Once again, one might argue that the AIM pedestrian heuristic played a confounding role in these results. Indeed, it is more difficult in the case of throughput to show that NEAT will always outperform AIM regardless of the method in place for dealing with pedestrians. Until an alternative method is designed and shown to be superior to our heuristic, we contend that the pedestrian heuristic was appropriate for the sake of candid experimental comparison. We also argue that our pedestrian heuristic highlights a number of the AIM paradigm's inherent flaws, notably its severe lack of modular design. Before adding the pedestrian heuristic to AIM's Driver Agent model, it was originally planned to fully integrate pedestrian support into the actual system. IMs would have been aware of vehicles applying the pedestrian heuristic, knowing that said vehicles would arrive at their destinations late and, as such, should not have their reserved tiles re-issued until further notice. It became clear, however, that even if this rationale was sound, implementation was impossible as the system's current infrastructure does not provide for a mechanism that allows IMs to know about or respond to unusual activity in the intersection. Once a vehicle has been granted space-time and is inside the intersection, the IM is powerless over it. Even if the system was not limited in this regard, adding pedestrian support would have still required the FCFS policy to be rethought from scratch. Designing a new intersection control policy was beyond the scope of this study, and should certainly not be required each time the system needs to be extended in some small but crucial way. It is also important to note that in every 'perfect world' scenario where a method of dealing with pedestrians was not even required, the NEAT controller consistently performed faster than AIM, and came considerably close to AIM's average throughput (NEAT's Scenario 1 average throughput, for example, was 98%). This certainly suggests that NEAT is capable of producing robust behaviours for intersection traversal, even with a minimal controller configuration such as what was used in this study.

A. Evolved Behaviours

Since NEAT was only responsible for evolving collision circumvention behaviour to act on top of the simulator's default driving behaviour, it was undesirable to find controllers with many exotic behaviours. Such controllers were found to conflict with the simulator's general driving behaviour: vehicles would stray and maneuver aggressively, constantly requiring the general driving behaviour's path-finding mechanism to guide the vehicle back on track. One such controller displayed a behaviour where vehicles would attempt to drive around obstructions in a spiral pattern to avoid collisions, while another displayed 'reversing behaviour' such that vehicles would reverse and go forward continuously until a detected obstruction was out of view. Controllers such as these tended to emerge during large training sessions of 200 or more generations with population sizes of 150 and above. More subtle but robust behaviours were found by reducing population size and number of generations, and increasing task trials per genome as well as NEAT's speciation compatibility threshold. These parameters encouraged fewer niches but more fine-tuned behaviours that suited the control task. An interesting observation was the fact that the best discovered controller did not require the *Distance to inside border of intersection* input, yet never seemed to attempt to circumvent collisions by straying out of the intersection. It was also interesting to observe how well the controller, evolved in a 2-lane intersection with just 1 pedestrian, generalised to increasingly complex scenarios that were absent from training. This was not too surprising, however, as the general structure of the environment was similar across scenarios, with differences being the number of lanes (and thus intersection size) and number of pedestrians.

VIII. CONCLUSIONS AND FUTURE WORK

This study assessed the viability of using NEAT to evolve decentralised collision-circumvention behaviour for autonomous vehicles in intersections over the centralised AIM paradigm - extended to heuristically account for pedestrians - with FCFS as the *intersection control policy*. NEAT evolved generalised and fine-tuned behaviours for the intersection traversal task, producing a single controller which almost exclusively outperformed AIM throughout the comparative experiments. AIM, in contrast, generalised poorly to any complexity, even with the addition of the heuristic to deal with unexpected pedestrians. We conclude that neuroevolution is conceivably applicable to the domain of autonomous intersection coordination, especially as a decentralised alternative to the AIM paradigm. However, both AIM and the decentralised model we have discussed will require significantly more research before transference from the simulated domain to the real world becomes a relevant discussion.

We propose the following ideas for future research:

- Alternative sensory configurations for NEAT driver agents, such as to include back sensors, the use of ray-casting rather than pie-slice sensors, and to find optimal sensory parameters via evolution rather than by hand.
- Using NEAT to evolve both driving behaviour and collision-circumvention behaviour, rather then leaving the former to a heuristic algorithm.
- Explore different Evolutionary Algorithms and variants of NEAT for the intersection control task, such as *HyperNEAT*.

[3]The interested reader is referred to https://github.com/rudolfbono/NAIM for all sourcecode required to reproduce or extend the experiments, as well as additional material.

- Testing performance in other types of intersections, instead of just the grid-based model.
- AIM pedestrian support which is fully integrated with the reservation system.

IX. Acknowledgements

I thank my supervisor, Dr Geoff Nitschke, for his invaluable guidance throughout the development of this study. I also extend my gratitude to Aashiq Parker for his helpful insights.

Appendix I
Simulation and NEAT Parameters for controller evolution

Simulation Parameters	
Environment size	250 x 250
Traffic Level	0.7
Number of vehicles per task (simulation) trial	30
Time steps per task (simulation) trial	50
Max vehicle acceleration	40
Max vehicle deceleration	-35
Max vehicle velocity	60
Min vehicle velocity	-17
Vehicle length	4
Vehicle width	1.75
Vehicle sensors	3
Height of each sensor view field	4
Width of each sensor view field	4
NEAT total field of view span	270 degrees
AIM total field of view span	30 degrees
Number of sensors	3
Pedestrian height	1
Pedestrian width	1
NEAT Parameters	
Population size	150
Number of generations	75
Task trials per genome	10
Initial connection density	0.75
Compatibility threshold	0.65
Generations allowed without improvement	15
Target number of species	40
Default survival rate	0.2
Number of activation cycles	4
Weight range	[-5,5]
Activation function	Hyperbolic Tangent
Sensory input nodes	5
Sensory output nodes	2

References

[1] Dresner, K. and Stone, P., 2008. A multiagent approach to autonomous intersection management. Journal of artificial intelligence research, 31, pp.591-656.

[2] Eiben, A.E. and Smith, J.E., 2003. Introduction to evolutionary computing(Vol. 53). Heidelberg: springer.

[3] Gershenson, C., 2003. Artificial neural networks for beginners. arXiv preprint cs/0308031.

[4] Gomez, F.J. and Miikkulainen, R., 2003. Robust non-linear control through neuroevolution. Computer Science Department, University of Texas at Austin.

[5] Narayan, Aparajit, Elio Tuci, and Frdric Labrosse. "Simulated Road Following Using Neuroevolution." Artificial Life and Intelligent Agents Symposium. Springer International Publishing, 2014.

[6] Forbes, J.R.N., 2002. Reinforcement learning for autonomous vehicles(Doctoral dissertation, UNIVERSITY of CALIFORNIA at BERKELEY).

[7] Stanley, K.O. and Miikkulainen, R., 2004. Competitive coevolution through evolutionary complexification. J. Artif. Intell. Res.(JAIR), 21, pp.63-100.

[8] Stanley, K.O. and Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), pp.99-127.

[9] Stanley, K., Kohl, N., Sherony, R. and Miikkulainen, R., 2005, June. Neuroevolution of an automobile crash warning system. In Proceedings of the 7th annual conference on Genetic and evolutionary computation (pp. 1977 1984). ACM.

[10] Beyer, H. (2002). Evolutionary Computation. MIT Press Journals.

[11] Wang, Y. and Schreiber, B., 2015. Creating a Traffic Merging Behavior Using NeuroEvolution of Augmenting Topologies.

[12] Yao, X., 1999. Evolving artificial neural networks. Proceedings of the IEEE,87(9), pp.1423-1447.

[13] Gauci, J. and Stanley, K., 2007, July. Generating large-scale neural networks through discovering geometric regularities. In Proceedings of the 9th annual conference on Genetic and evolutionary computation (pp. 997-1004). ACM

[14] Whiteson, S., Stone, P., Stanley, K.O., Miikkulainen, R. and Kohl, N., 2005, June. Automatic feature selection in neuroevolution. In Proceedings of the 7th annual conference on Genetic and evolutionary computation (pp. 1225-1232). ACM.

[15] Risi, S. and Togelius, J., 2014. Neuroevolution in games: State of the art and open Challenges.

[16] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G. and Lau, K., 2006. Stanley: The robot that won the DARPA Grand Challenge. Journal of field Robotics, 23(9), pp.661-692.

[17] Buehler, M., Iagnemma, K. and Singh, S., 2008. Editorial:[for the Special issue on the 2007 DARPA Urban Challenge, Part I]. Journal of Field Robotics, 25(8), pp.423-424.

[18] Togelius, J. and Lucas, S.M., 2006, July. Evolving robust and specialized car racing skills. In 2006 IEEE International Conference on Evolutionary Computation (pp. 1187-1194). IEEE.

[19] Aravind Gowrisankar, B.E., 2008. Evolving Controllers for Simulated Car Racing Using Neuroevolution.

[20] Moriarty, D.E. and Langley, P., 1998. Distributed learning of lane-selection strategies for traffic management. Daimler-Benz Res. Technol. Center, Palo Alto, CA, Tech. Rep, pp.98-2.

[21] Gauci, J. and Stanley, K.O., 2008, July. A Case Study on the Critical Role of Geometric Regularity in Machine Learning. In AAAI (pp. 628-633).

[22] Pugh, J.K., Soros, L.B. and Stanley, K.O., 2016. Quality diversity: A new frontier for evolutionary computation. Frontiers in Robotics and AI, 3, p.40.

[23] Abdulhai, B. and Kattan, L., 2003. Reinforcement learning: Introduction to theory and potential for transport applications. Canadian Journal of Civil Engineering, 30(6), pp.981-991.

[24] Togelius, J., Lucas, S., Thang, H.D., Garibaldi, J.M., Nakashima, T., Tan, C.H., Elhanany, I., Berant, S., Hingston, P., MacCallum, R.M. and Haferlach, T., 2008. The 2007 ieee cec simulated car racing competition. Genetic Programming and Evolvable Machines, 9(4), pp.295-329.

[25] Naumann, R., Rasche, R., Tacken, J. and Tahedi, C., 1997, November. Validation and simulation of a decentralized intersection collision avoidance algorithm. In Intelligent Transportation System, 1997. ITSC'97., IEEE Conference on (pp. 818-823). IEEE.

[26] Farahmand, A.S., 2008. Cooperative decentralized intersection collision avoidance using extended kalman filtering (Doctoral dissertation, Virginia Polytechnic Institute and State University).

[27] Hafner, M.R., Cunningham, D., Caminiti, L. and Del Vecchio, D., 2013. Cooperative collision avoidance at intersections: Algorithms and experiments. IEEE Transactions on Intelligent Transportation Systems, 14(3), pp.1162-1175.

[28] Hall, S., Laumonier, J. and Chaib-Draa, B., 2004, October. A decentralized approach to collaborative driving coordination. In Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on (pp. 453-458). IEEE.