The Relationship between Complexity and Functionality in Evolutionary Robotics

Literature Review

Danielle Rose Nagar
University of Cape Town
Cape Town, South Africa
ngrdan001@myuct.ac.za

ABSTRACT

Evolutionary Robotics offers a promising medium in the automated design of adaptable robots - where hand-designed models are unfeasible. The potential of this field is encapsulated in its ability to produce complex behaviors for teams of co-operative robots; *Multi-robot Systems*. In implementing these systems, it is necessary to find viable trade-offs between functionality and complexity. An evaluation of Multi-robot Systems with multiple objectives will allow for a more nuanced understanding of these different trade-offs. This review surveys the literature on the mechanisms used to evolve Multi-robot Systems with a specific focus on NEAT, HyperNEAT and multi-objective Evolutionary Algorithms, as implementation methods. These techniques are presented along with various applications of Evolutionary Robotics.

1 INTRODUCTION

For many problems in the field of robotics, cooperation among multiple robots is more suitable than the use of a single expensive robot. *Multi-robot Systems (MRS)* [27] - sets of coordinated robots operating in common environments - have been applied to problems such as completion of tasks in dynamic hazardous spaces [34] and mapping of unexplored environments [57]. However, specifying the complex and cooperative behaviour policies used in these systems via traditional analytical methods is generally unfeasible [27, 41]. As such, the field of *Evolutionary Robotics (ER)* [21] presents a paradigm for the automated design of robust and adaptable MRS.

In general, ER is the application of evolutionary-inspired methods to the automatic design of autonomous robots [29]. These methods have been found to facilitate the evolution of both robot controllers and morphologies (both separately [30, 49, 52, 72] and simultaneously [6, 61, 62]). Consequently, it is highly applicable to the field of MRS which often requires the evolution of both control policies for cooperative behavior as well as the evolution of morphological (sensory) configurations [74]. Indeed, embodied cognition [13] posits that intelligent behavior emerges not just from the brain (controller), but from the interplay between the brain, body (morphology) and environment [53]. This idea is corroborated by contemporary work in ER [17].

A common approach to the design of controllers in ER is through *neuroevolution* (NE) [25], which has proven to be useful for controller design tasks across various disciplines [14, 33]. MRS that employs neuro-evolution techniques include collective construction tasks [74], robot soccer [20] and video games[60]. However, neuro-evolution approaches typically do not take into account the inherent issue of *trade-offs* in these examples. In many real-world tasks, every solution contains various trade-offs between functionality and complexity. For example, the most sophisticated solution could elegantly overcome some problem; however, this will usually not be feasible or cost-effective.

A way to interpret the trade-offs and dynamics of these different facets is through multi-objective optimization techniques [22, 23]. Such techniques allow the modeling of different optimal solutions that encompass various trade-offs in relation to multiple objectives [71].

This literature review provides an overview of the significant NE mechanism used in ER, namely NEAT [68] and HyperNEAT [67]. Subsequently it explores the aggregation of NE and multi-objective optimization techniques. Finally, the larger taxonomy of ER is reviewed with a focus on the evolution of controllers and morphologies separately and simultaneously.

2 NEURO-EVOLUTION

Evolutionary Computing (EC) covers a broad range of meta heuristic techniques that are influenced by Darwinian principles of natural selection [23]. EC uses a stochastic population based approach, where a population of solutions compete for survival and reproduction based on their ability to satisfy environmental requirements [7]. These requirements, collectively known as the fitness function, indicate how well a solution is performing at the task. The process begins with an initial population (generation) and iteratively selects the better performing candidates for reproduction and selection into the next generation [24].

The potential of an evolutionary approach is evident by examining the sophistication and diversity of organisms in nature [21]. Furthermore, EC has a wide range of applications [24]. One successful application is the designing of a University lecture and event schedule which is known to be a highly constrained and difficult problem [51]. Procedures within EC are known as Evolutionary Algorithms (EA) and there exists multiple variants, including *Genetic Algorithms* [42], *Genetic Programing* [10] and *Evolution Strategies* [7].

Neuro-evolution [25] is an approach that unifies two biologically-inspired techniques, EA and Artificial Neural Networks [58]. The next section will address these two techniques and how the combination of their respective advantages makes NE especially effective for controller design in MRS [21].

2.1 Mechanisms of Neuro-evolution

2.1.1 Evolutionary Algorithms. Evolutionary Algorithms (EA) utilize a Darwinist inspired approach to selection and exploration

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Figure 1: Evolutionary Algorithm pseudo-code [23]

of a search space [7]. Exploration is achieved through reproduction with two variation operators, *recombination* and *mutation*. These simulate the production of novel and varied solutions [59]. Recombination is a binary mating operator applied to better performing candidates (parents' genotypes), where the exchange of information generates one or more new candidates (children). Mutation slightly modifies one candidate which results in some new candidate. Some selection methodology is used to choose which candidates form the next generation. Examples of this include *elitism* [8], which chooses the better performing candidates from both parent and children populations, or selecting all children. The algorithm terminates once either a desired solution has been found or some specified threshold has been met [23, 24]. Figure 1 provides a brief overview of algorithmic structure of an EA.

Within an EA's problem context, candidate solutions are referred to as phenotypes [23], a core component of an EA is encoding these phenotypes into simpler structures known as genotypes. Genotypes can then be decoded into their respective phenotypes for evaluation with the fitness function. The means by which a phenotype is represented as a genotype is the defining characteristic between differing EAs [7]. For example, Genetic Algorithms use a bit-string representation [42]. Variation operations can be implemented simply through bit-flipping and bit-exchanging algorithms. Consequently, they are one of the more popular EAs. Other representations include Genetic Programing [10] and Evolutionary Strategies [7] which use tree encodings and vector encodings, respectively. The representation chosen is dependent on the domain of the problem.

The literature shows that EAs are particularly efficient at finding the global optimization in large, noisy and discontinuous search spaces [23, 51]. Moreover, they also have the unique capability of evolving novel solutions without human bias [21]. A particularly powerful phenotype is presented in Artificial Neural Networks. [75].

2.1.2 Artificial Neural Networks. Artificial Neural Networks (ANN) [58] are processing models inspired by neural activity in the brain. This activity occurs from neuro-chemical communication between neurons and through synapses in the brain . An ANN's structure consists of layers of multiple processing units (neurons) interconnected via directed links (synapses). A synapse consists of a numeric weight which determines the strength of the connection. The input layer is the first layer in an ANN and receives it's input from the environment. Similarly, the output layer is the final layer of an ANN and produces the output. The output actuates the controllers' behavior. All intermediate layers are known as hidden layers. In most cases, each neurons output is computed by an activation function of the weighted sum of its input [25, 43].

ANNs can approximate non-linear target functions, are proficient in dealing with large search spaces, and can be built using recurrent connections such that previous input patterns can effect current output [75]. Consequently, they provide a powerful and robust manner of implementing controllers [28]

Traditionally, an ANN is trained by adjusting the synapse's weights in order to approximate some function. This process can be performed through a supervised learning approach, such as back-propagation which attempts to minimize the error function between the actual output and desired output of a network [43]. However, back-propagation is not ideal in applications where the error function is non-differentiable. So, another approach is by means of unsupervised learning algorithms such as EAs[75].

Neuro-evolution is the unsupervised learning approach of training ANNs using EAs [25]. In NE, an optimal solution is derived by evolving a population of ANNs (phenotypes). ANN genotype encodings traditionally consist of the concatenation of its synapses' weight values contained in either a binary representation or real-valued vector[75]. Every genotype within a population is decoded into their respective ANNs which then executes some task. A fitness value is then determined for each ANN. The fittest members of the population are then chosen and variation operations (recombination and mutation) are applied which result in the new child population.

There are two broad types of NE, direct encodings and indirect encodings [40]. Direct encoding is defined by a one-to-one mapping of phenotype and genotype. The explicit relationship between representation makes direct encoding far simpler and more natural. In contrast, an indirect encoding utilizes a higher and more compact level of abstraction. The genotype defines some function to construct the phenotype. Indirect encoding is more representative of genetics [75].

Other cogent optimization methodologies that are worth comparing to NE include traditional Reinforcement Learning(RL) [69], Particle Swarm Optimization(PSO) [54] and ANN back-propagation training techniques [58].

RL refers to a wide range of machine learning algorithms that improve some solution based on behaviorist psychology such as *operant conditioning* [43]. A solution policy explores an environment and maps states to actions based on numerical reward signals that serve as feedback [69]. However, within the domain of MRS its search space increases drastically with respect to the number of agents. NE is also shown to be more robust and exhibits better exploration behavior [59].

PSO is another such algorithm that iteratively improves a population of candidate solutions (particles) that move around a virtual search-space [54]. Particles are modeled after the collective behavior exhibited by entities in nature, such as termites or ants. Where a particle's migration is influenced not only by its local best known position but also the best known position of other particles [36]. In comparison to NE, PSO performs efficiently but requires more candidate solutions and consumes more time [76].

ANNs can be trained the supervised learning approach of back-propagation [43, 58]. However, supervised learning is not suitable for complex control systems where the desired behavior is unknown.

Essentially, NE is one of the most effective methodology for producing complex controllers. This is due to the general applicability of NE, its operation in continuous search spaces, and possible applications to a broad range of network architectures, including recurrent networks [28, 40]. Consequently, NE is a promising methodology for solving design problems for MRS [41]. The following sections discuss the more notable NE algorithms.

2.2 Conventional Neuro-evolution

Conventional Neuro-evolution(CNE), is the simplest approach to evolving an ANN [65]. In this approach the topology and activation function of an ANN are defined *a priori*. A direct encoded schema, such as binary representation, is then used to evolve the synapse weights of an ANN [75].

However, this approach suffers from several issues, one of which is premature convergence [40]. Another issue is the *Competing Conventions Problem* [65]. This occurs when similar solutions can be expressed through different genotypes. For example, two ANNs that differ only by inverted hidden nodes. The recombination of these solutions often results in duplicated structure and thus poor performing children [25]. Moreover, the synapse weights of ANN are not the defining characteristic of how well an ANN performs. Rather, it has been shown that the topology of an ANN plays a role too [68]. An promising alternative to conventional neuro-evolution is to evolve topology as well.

2.3 Topology and Weight Evolving Artificial Neural Networks

Topology and weight evolving artificial neural networks (TWEANNS) [65] provide an automated method of searching for an ideal topology and weight. This saves the designer time and has also been shown to be more effective than any human designed topologies. However, TWEANNS still suffer from the *Competing Conventions Problem* and two additional difficulties, namely protecting innovation and initial population specification [65].

Innovative ANNs arise from the addition of a new connection or node to the structure of an ANN. These initially perform poorly as they still require training to optimize [65]. Thus, it is necessary to protect innovative ANNs and allow them this optimization time before they can fairly compete on a global scale.

The third issue of TWEANNS is the creation of an initial population of candidate solutions. A random approach is not ideal as it can lead to infeasible solutions. For example, a lack of synapses between layers. Moreover, there is the potential of the initial solutions being too complex and more minimalistic solutions being lost [65].

There have been multiple attempts to subvert these issues. One such example is Symbiotic Adaptive Neuro-evolution (SANE) [44], which evolves neurons and topology separately. An ANN is formed by assigning neurons to a topology. This assignment process is based on the adequacy of a neuron's previous interactions with a specific topology. The assembled ANN is then evaluated. Topologies are assigned these evaluations and neurons are assigned the N-best evaluations they participated in, where N is pre-defined. This approach was demonstrated to evolve satisfactory and diverse networks swiftly [44]. However, SANE is unable to evolve recurrent networks [28]. This is a significant drawback as many interesting and complex tasks, especially within MRS, require some form of recurrent memory.

The most well known and effective TWEANN algorithms that attempt to solve these problems are Neuro-evolution of Augmenting Topologies (NEAT) [68] and HyperNEAT [67]. The difference between the two techniques being that NEAT uses direct encodings; whereas HyperNEAT uses indirect encodings.

2.3.1 Neuro-evolution of Augmenting Topologies. NEAT [68] is a direct encoded genetic algorithm that is based on three fundamental principles:

(1) Historical markings which offers a solution to the competing conventions problem,
(2) Speciation to protect innovative solutions, and
(3) Complexification in order to favor minimalistic solutions.

NEAT adapts a genetic encoding schema that represents ANNs with a list of *connection genes*. Each one of these genes refers to the connection of two *node genes* and stores other pertinent information such as the weight and whether a connection is enabled. A connection gene also specifies an *innovation number* which is a historical marker that represents the gene's origin. NEAT assigns these markings by keeping track of a counter called a *global innovation number*. When a novel gene appears in the system (through mutation), the *global innovation number* is incremented and assigned to the gene. The genetic encoding representation paired with the process of historical marking allows for similar genes to be lined up during the recombination process. This inherently solves the *competing conventions problem* as recombination methods can be applied sensibly.

However, the addition of novel genes to a solution causes an initial decrease in overall fitness. These solutions require additional time to optimize before being able to fairly compete with other optimized solutions. NEAT solves this problem by allowing organisms to compete with similar organisms rather than the entire population. Thus, speciation is used to categorize solutions based on their topology. The comparing of topologies can be done trivially through the use of historical markings. In every generation, the process of speciation occurs. The process first randomly selects a group of solutions from the previous generation. These solutions represent the species and are referred to as representative solutions. All solutions are then iteratively placed in a species based on their topological similarity with the representative solution. Competitions are then performed within a species which accordingly protects innovation. Innovation is further protected through the use of explicit fitness sharing where species share a single aggregated fitness value. This ensures that species remain reasonably small and thus cannot take over the population.

The final principle element of NEAT is complexification. It is based on the concept that structures should start out as minimalist as possible and grow in complexity as they evolve. Consequently, the initial population of solutions are uniform and contain no hidden layers. This ensures that all complex solutions are justified and

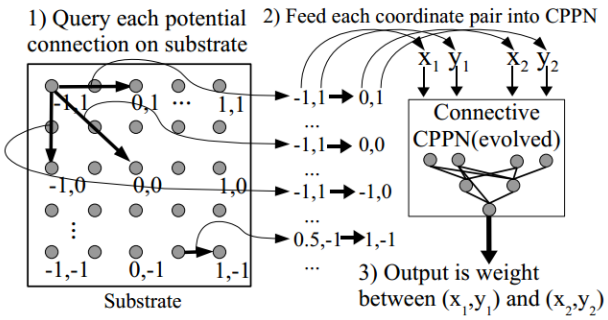

Figure 2: Creating two-dimensional Connectivity Patterns in HyperNEAT

gives NEAT a performance advantage over other approaches, such as SANE [28, 44].

NEAT has been successfully used in many ER and MRS applications such as, evolving autonomous cars [72] and simulating video games [60].

2.3.2 HyperNEAT. HyperNEAT [67] is an extension of NEAT which allows the evolution of large ANNs by exploiting geometrical patterns in the ANN controller and task environment. HyperNEAT utilizes an indirect encoding schema for TWEANNS. The genotype representation of an ANN is a connective Compositional Pattern Producing Network (CPPN). CPPNs are compact 'abstractions of development' that are able to generate complex and repetitive patterns [66]. They have a similar structure to that of ANNs but differ in their ability to utilize multiple activation functions compared to ANNs' utilization of just one.

A CPPN is able to compactly represent extremely large structures by mapping co-ordinates from a space of few dimensions to a space of many. This ability allows a CPPN to produce complex ANNs. Prior to this production, a substrate (grid of nodes on a Cartesian plane), needs to be specified. A CPPN then accepts the Cartesian co-ordinates of any two nodes in the substrate and returns a weight. A connection is made between these two nodes only if the weight produced is above a certain threshold [67]. Thus, a two-dimensional connectivity pattern is represented by a CPPN's spatial pattern in a four-dimensional hypercube. This is presented in figure 2.

There are two main advantages for using a CPPN as an indirect encoding [67]. The first advantage is that complex and large ANNs can be represented in a more compact form. This is particularly important for the modeling of natural phenomena such as the brain where there are approximately 100 trillion connections. Secondly, CPPNs possess the unique ability to exploit the physical underlying structure inherit in the problem. This is also evident in biology with the structure of a physical body being exploited in the construction of its brain. For example, the symmetry and regularity of an organism's eyes is reflected in its neural organization.

HyperNEAT's foundations lie in CPPN-NEAT [66] where the principles of the NEAT algorithm are easily extended to evolve CPPNs. In HyperNEAT, the substrate configuration is designed *a priori* based on the problem. A population of CPPNs are then iteratively queried to produce connections within a substrate which results in an ANN. Each ANN is evaluated based on some fitness function. The next generation of CPPNs are then produced based on the NEAT algorithm [67]. It is important to clarify that HyperNEAT can be extended to more than two dimensions.

Evolvable-substrate HyperNEAT [55] is a recently developed extension of HyperNEAT. It is particularly promising as it deduces the optimal placement of nodes on a substrate. Thus, it further widens the varieties of discoverable structures.

HyperNEAT has been applied successfully as a technique to evolve controllers, both for MRS and for single-agent robots [56, 73, 74]. Moreover, The principle elements of CPPNs, symmetry, regularity and compactness, offer the potential of being able to model morphologies [66]. In such algorithms CPPN-NEAT is used and the values returned by the CPPN define the placement of morphological segments on the robots body[6, 16].

3 MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

Within the research area around optimization problems, tasks may require multiple competing objectives. Since these objectives may be conflicting, a question arises of how to consolidate them.

Multi-objective Evolutionary Algorithms (MOEA) [12] are an increasingly popular solution to this problem. More robust MOEAs produces a set of *Pareto-optimal solutions* that embody optimal trade-offs between several objectives [1, 23]. A candidate solution is Pareto-optimal if there exists no other solution that could perform better at a single objective, without decreasing the performance of other objectives [22]. Pareto-optimal solutions are identified through the concept of domination. A solution is said to dominate another solution if it is superior in relation to one objective and performs no worse in all others. The solutions that are not dominated by any other solutions are the known as the Pareto front and are considered the best solutions. EAs are a favorable method for Pareto-optimal problems as they work with a population of solutions and can thus return a set of best solutions. Moreover, EAs are efficient at handling continuity and concavity of the Pareto-optimal solutions. Within the field of robotics, MOEAs have become increasingly popular as they allow designers to understand the trade-offs between various solutions and hand pick their preferred trade-off [71].

It is also possible to produce a single Pareto-optimal solution through aggregating multiple objectives into a single objective fitness function [22]. This is conventionally achieved through using a weighted sum of all objectives. However, this approach has various drawbacks. Firstly, a trial-error approach is required to find the optimal weight. Further, in some cases, certain optimal solutions cannot be found [45].

There exists multiple algorithms that attempt to provide MOEAs. While there are many implementations, two prominent ones can be identified. The rest of this section reviews these MOEAs and goes on to discuss multi-objective NEAT and HyperNEAT.

3.1 The Non-dominated Sorting Genetic Algorithm

The Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb and was one of the first prominent

MOEAs [64]. In each generation of this algorithm, all non-dominated candidates are clustered into a single category and are subsequently removed from the population. This process is repeated until all solutions have been categorized. Variation operators are then assigned based on the order of the categories determined. However, this algorithm is not very efficient as it requires high computational complexity to categorize candidates [19].

In 2001, Deb et al., proposed NSGA-II [19], a more efficient variation of NSGA. The algorithm is based on a principle of sorting as well as a density estimation. A solution is ordered based on the number of solutions that dominate it and the number of solutions it dominates. The density estimation takes into account how spread out all the solutions are. Selection and variation are based on novel and pareto-optimal solutions. NSGA-II is one of the most efficient MOEAs and in recent years has become the benchmark for other MOEAs [18].

3.2 The Strength Pareto Evolutionary Algorithm

The Strength Pareto Evolutionary Algorithm (SPEA) [78] is another popular technique within MOEA algorithms. SPEA is based on elitism and embodies this by implementing an archiving technique. An archive contains all non-dominated solutions that have existed in previous generations. For each generation, a solution is evaluated based on the dominance relationship with the archived population, this value is known as the *Pareto Strength*. This approach effectively keeps track of all optimal solutions while ensuring their variance. However, if the archive becomes too large too quickly, the search may slow down due to a reduction in selection pressures [18].

SPEA2 [77] was thus presented as an extension of SPEA to subvert this problem. SPEA2 utilizes a mechanism that reduces the size of the archive to be below a certain threshold. This mechanism also incorporates more efficient density estimation techniques.

3.3 Multi-objective NEAT and HyperNEAT

NEAT and HyperNEAT are two of the most prominent NEAs. However, they were developed for single optimization problems. With an increased demand for MOEAs, integrating these has become an active focus of contemporary research. A core issue in this integration is maintaining all the principle features of NEAT. This section provides a brief time-line of some of the more prominent multi-objective NEAT and HyperNEAT implementations.

A seminal work in this domain was NEAT-PS [72] produced by Willigen et al. in 2013. NEAT-PS attempted to incorporate the Pareto Strength approach from SPEA-II into NEAT. This was achieved through augmenting the fitness function of NEAT to use an adaptation of the Pareto-strength value approach in SPEA2. Subsequently, the fitness function was plugged into the original NEAT algorithm. This work successfully used NEAT-PS to find the optimal trade-offs between comfort (fewer lane changes) and speed (more lane changes) in vehicle controllers [72]. However, since NEAT does not follow an elitist strategy, NEAT-PS does not either. Consequently, Pareto-optimal solutions could be lost.

In 2014, Bongard and Auerbach implemented a multi-objective HyperNEAT implementation in their research surrounding morphological complexity in robots [6]. The algorithm replaced the speciation and selection principles of NEAT with a NSGA-II implementation. Subsequently, an additional fitness function based on genotypic diversity was included in all experiments. This additional function was meant to protect the innovation that speciation brings in HyperNEAT. There are two fundamental concerns with this algorithm. The first is that the algorithm does not address its removal of speciation. This could result in decreasing the reliability of the algorithm. Secondly, there may be issues that arise from the genotypic diversity function. In particular, solutions may advance through generations for their novelty as opposed to their ability to perform well at the task.

Another multi-objective HyperNEAT algorithm was produced by Cheney et al. in 2015 [16]. This paper looked at the training of soft robots (explained in 4.2) to attain two conflicting objectives; namely, escaping a partially enclosed box and, increasing in size. The algorithm used is said to be a novel implementation that maintains NEAT's speciation principle. However, the algorithm is not the focus of the paper and thus very little detail is shared on this implementation. Furthermore, comparisons to other methods are not supplied.

In contrast, a recent paper by Abramovich et al. presents a generic multi-objective NEAT implementation and compares its performance to NEAT-PS [2]. The proposed algorithm is named NEAT Multi-objective Diversified Species (NEAT-MODS). NEAT-MODS considerably augments the selection mechanism while still maintaining the fundamental principles of NEAT. The selection mechanism considers both parents and offspring and sorts them using the NSGA-II non-dominating sorting algorithm. The best solutions are then chosen to be the representative candidates for speciation as in NEAT. The rest of the population is then placed in their appropriate species based on their non-dominance relation. Within each species, the solutions are again sorted based on the non-dominating sorting algorithm. The next generation is then selected based on the best available individuals in a serial progression of the species. When comparing this algorithm to NEAT-PS, it was found to be substantially more effective [2]. A reason for this could be the elitism aspect of NEAT-MODS and its selection process that favors novelty and diversity.

From reviewing the literature, it can be seen that there is still a substantial amount of research that needs to be done in the domain of multi-objective NEAT and the general domain of multi-objective neuro-evolution. Moreover, these methods have rarely been applied to MRS applications.

4 EVOLUTIONARY ROBOTICS

Evolutionary Robotics (ER) [29] is a biologically influenced technique which employs evolutionary algorithms for design of physical (morphological) and control structures of robots. ER is a prominent field within robotics as it subverts the constraints of traditional human intuitive design. It presents optimal designs through intelligently exploring large and complex solution spaces. ER is therefore better equipped at designing non-linear, robust and adaptable morphological and control structures as well as handling uncertainty of behavioral requirements [28]. In many cases, ER methods are able to find solutions that would be unintuitive to humans [3].

A key component of ER research is how the environment, morphology and controller inter-operate to influence the design of the architecture while considering trade-offs [21]. This section presents a collection of research within this domain and classifies them based on the three major ER subfields- evolutionary controllers, evolutionary morphology and the co-evolution of both controllers and morphology.

4.1 Evolutionary Controllers

During the 1990's, evolutionary computation techniques dominated much of the research around controller design for a fixed morphology [25, 29]. One of the more renowned works is that of Beer and Gallagher which successfully makes use of neuro-evolution to mimic the adaptiveness and versatility of locomotion in insects [11]. Other works in this era focused on the evolution of controllers for Khepera robots with fixed sensory configurations [35, 47, 48, 63].

However, there were few works that attempted to addresses the role of the environment in relation to the complexity of the controller itself. One such work, was in 1998 where Odagiri et al. [49] evolved a Khepera robot with static sensors in four increasingly complex maze environments. The results reflected a direct correlation between environmental complexity and control structure complexity. This pivotal work and others in this era paved the way for controller based ER. However, there was room for improving on NE techniques with a specific focus on more complex environments.

In the 2000's the advent of NEAT and HyperNEAT subverted many previous issues around NE and they became the most prominent and effective techniques within this domain [67, 68]. The potential of these methodologies is shown in Stanley and Risi's [56] publication on their successful creation of a controller that can be adapted to unique and novel derivatives of a morphology. HyperNEAT is used in this experiment which accepts hand-designed morphologies and outputs a controller.

Another relevant HyperNEAT study is Watson and Nitschke's evaluation of the degree of sensory configuration complexity needed to successfully evolve a controller for homogeneous MRS collective construction [73]. The morphologies were designed *a priori* and HyperNEAT was used to create controllers. An interesting outcome of this research, was that task performance did not sufficiently increase with more complex sensory configuration. An extension of this research could evaluate if this holds true in heterogeneous systems.

More modern approaches attempt to combine this domain with a multi-objective approach. A seminal work is that of Willigen's [72] which evolved autonomous vehicle controllers for both speed and comfort using *NEAT-PS*, an extension of NEAT with multi-objective support. We treat NEAT-PS in greater depth in section 3.3. The aim of Willigen's work was to allow autonomous vehicle passengers to prioritize different 'driving experience' parameters based on individual needs. However, NEAT-PS outputs a single solution which intelligently weights objectives into a single fitness function.

4.2 Evolutionary Morphology

One notable challenge within the ER field is the design and selection of a specific morphology for a given problem domain [29]. EAs have been demonstrated as successful in this aspect as they are able to obtain solutions that are in some cases unintuitive to humans. A distinguished example is the use of a genetic algorithm to automatically find novel antenna designs. The designs produced were on average 55% more effective than would otherwise have been developed [31]. Furthermore, the throughput of designs exceeded that of traditional engineering techniques.

Most of the early papers in this domain defined morphology as the connection of modular static structures with rotational joints to form complete functional systems [3, 61, 62]. LEGO blocks were a particularly popular apparatus as they were able to transfer easily from simulation to real world. Research employing LEGO blocks was able to successfully evolve structures such as bridges, chairs and towers for multiple different objectives including stability and height [26, 52].

However, a key criticism in this area is the limitation of having a predefined set of components to choose from. Consequently, in 2001, Hornby and Pollack proposed L-Systems as a more flexible approach to generating morphologies [32]. L-Systems are a formal grammar rewriting tool, developed by Biologist and Botanist Lindenmayer to model plant and cellular development [38]. In Hornby and Pollack's work, they exploited L-Systems as genetic encodings for an EA and successfully evolved simulated creatures for locomotion. These creatures displayed far more complex and regular morphologies than previous works [32].

The work within rigid morphologies was also revolutionized by the development of HyperNEAT. The adoption of CPPN-NEAT to evolve morphologies is a growing trend [4–6, 66]. One such example was produced by Auerbach and Bongard, which successfully evolved robots for locomotion using HyperNEAT [4]. The CPPNs produced were efficiently converted into three-dimensional morphological structures. Their subsequent work evolved both morphology and controller using HyperNEAT [5].

In 2010, Hiller and Lipson shifted the focus of research in the evolution of morphology from rigid to soft robotics [30]. The defining property of soft robots is their ability to contract and expand their amorphous physical form. This has various real word advantages such as their increased robustness, ability to conform to uneven surfaces and efficient stress redistribution [16]. However, they follow a more probabilistic model of control in comparison to their rigid body deterministic counterparts. Hillel and Lipson employed three different types of morphological materials which yielded different qualities of volumetric expansion and contraction. The evolved robots made use of these materials for locomotion in order to scoot, bounce and flop [30]. A notable finding in this research is that behavior is not only affected by controller and morphology but also by material representation. Cheney et al., followed on from this research by noticing how these findings coincided with biology where parts of cognition are physically embedded in animals [17]. They created soft electro-physiological robots inspired by the electrical properties of cardiac tissue. Morphological materials were formed by different types of electrical conductors. Electric signals were then propagated throughout the physical bodies and governed the behavior (control) of the robots. CPPN-NEAT [66] was used to produce the evolved morphologies which themselves exhibited complex and interesting forms of locomotion. This paper [17] was

one of the first to challenge the traditional disjoint between morphology and control in ER. However, further research could be conducted using a multi-objective optimization algorithm rather than a shared fitness function.

Another study on soft robots was by Bongard et al., which investigated the performance of these robots with a better suited task to match their unique capabilities [16]. The study used a multi-objective CPPN-NEAT implementation to evolve a soft robot morphology in a semi enclosed box. The two main objectives were to maximize morphological size as well as developing capability to reach outside the box. Results showed that many robots were able to go further than just reaching outside of the box but rather escaping from it. However, further research needs to be done into the way environmental parametric changes affect results, such as box and hole size. Furthermore, as mentioned above in Section 3.3, their NEAT multi-objective algorithm was not tested in comparison to other algorithms.

A key work that deals with the relationship between morphological complexity and environment is that of Auerbach and Bongard in 2014 [6]. The research was formed around the 'arrow of complexity' hypothesis which states that systems within an evolutionary context increase in complexity over time. In order to further explore this idea, the study investigated the role that environment plays in morphological complexity. Consequently, the study used 50 different icy environments that differed in complexity. Two main experiments were carried out in these environments. The first one, evolved morphologies for a single objective, locomotion. The results showed that morphological complexity increased regardless of the environment. The second experiment added a second objective, producing as simple morphologies as possible. In this case, a direct correlation between an increase in morphological complexity and environmental complexity was found. Thus, the research concluded that 'the arrow of complexity' holds when there is no cost on complexity. However, when there is a cost to complexity (as in nature), there is a proportional relationship between environment and complexity. This work could be furthered by evaluating if this conclusion holds true for MRS. It is also important to mention that the controller evolved concurrently with the morphology in this study. However, the experimental design ensured simplicity of the controller to maintain the research's focus on morphology. Thus, subsequent studies could investigate the role of various trade-offs, such as energy expended in these environments.

4.3 Evolution of Control and Morphology

The seminal work produced by Karl Sims in 1994 can be considered the founding work in the field of controller and morphology co-evolution [61, 62]. Sims produced two renowned papers that presented a novel system for evolving the morphology and control of virtual creatures in a three-dimensional environment. The first paper evolved creatures capable of producing numerous behaviors such as walking and jumping [62]. The second evolved creatures to compete against one another based on their ability to capture a cube [61]. A creature's morphological structure was defined by rigid cuboids. Sims adapted an evolutionary algorithm which used a directed graph as the genotype representation for both the morphology and control structures. The use of a directed graph allowed for flexible and efficient methods of its conversion to physical structures. However, there was a lack of previous work around using a directed graph as the genotype and thus Sims created his own genetic variation operators. A core component of SimâĂŹs research involved the creation of a custom designed physical simulator. The work produced by Sims was groundbreaking for the time and many attempts were made to extend the research. However, it took several years before an equally advanced and computationally powerful simulator was implemented. The first successful extension of Sims was by Taylor and Massey in 2000 which implemented creatures comprised of cylinder segments as opposed to cuboids [70]. Many extensions which followed created more effective evolutionary methods and explored different control tasks [15, 37, 50]. One notable implementation is that of Krcah which employed an extension of NEAT [33].

Despite the groundbreaking and fascinating work that Sims' framework (and its extensions) were able to produce, they were not applicable to a real-world context. Mechanical structures that could produce similar creature movement and joint materials do not currently exist [3]. This concept is referred to as the *Reality Gap* [46]. It suggests that the transfer from simulation to reality often leads to poor results and in some cases is infeasible [22].

One technique to overcome the *Reality-Gap* is to only evolve a select number of robot behaviors and physical features that can be applied to the real world [22, 46]. Such an approach is refereed to as Parametric Evolution [3] which evolves certain properties of robots, rather than the topological approach mentioned in previous sections. However, much of this work was still inspired by Sims.

The most prevalent parametric approach is the configuration of a robot's sensory system. An early work in this domain was produced by Balakrishnan and Honavar in 1996 [9]. Wherein, the placement and radius of eight sensors on a robot were evolved to optimize its ability to shift boxes in a two-dimensional environment. The robot controller was implemented using neuro-evolution and allowed for sensors to be turned off. However, the robots were not penalized nor rewarded for turning off their sensors. A relevant outcome of this experiment was that some of the more capable robots turned off certain sensors regardless. An extension of this work could potentially explore the reasons behind this.

Another significant experiment was conducted by Mark et al. in 1998 [39]. In this experiment, both the number and angle of view of optic sensors were evolved for a robot. These robots were placed in a simulated iterative environment and were tasked with keeping their energy level from decreasing to below a specified threshold. Energy levels decreased naturally over time or when robots crashed. Energy levels increased when a robot interacted with a light form. The relevance of this paper lies in its ability to train robots to use reduced amounts of energy. This offers a particularly fascinating insight into how we can understand trade-offs. However, this experiment [39] was not able to be carried out due to inefficient amount of computational resources.

Similar to the disciplines discussed in the previous sections, more recent papers in this domain tend to utilize the capabilities of NEAT and HyperNEAT. In 2015 Watson and Nitschke explored the optimization of the number of sensors for heterogeneous MRS [74]. In this paper two sets of MRS were produced for a collective construction task. The first set was hand-designed while the second set was

produced by HyperNEAT. A compelling outcome of this research was that the controllers produced by HyperNEAT were able to better adapt to morphological change than the hand-designed set. An extension of this research could evaluate if this outcome holds true for morphologically heterogeneous robot teams.

5 CONCLUSIONS

Evolutionary Robotics has been able to produce innovative robots for the completion of complex tasks. It has successfully been able to investigate various fields such as MRS for collective gathering, swarm robotics and evolutionary processes. The advent of advanced NE methods has laid the foundations for ER's vast progress in the last few years. However, there is still much progress to be made in ER. This literature review has identified two main fields that have the potential to be investigated further.

Firstly, multi-objective algorithms for NEAT and HyperNEAT are not yet fully developed. There are various algorithms which take multiple objectives into account. However, there have been few successful attempts to incorporate these techniques into NEAT and HyperNEAT. There have been even fewer which conclusively maintain the principles of all these methods. Moreover, there is little research conducted on the application of these in MRS.

Secondly, few research studies have explored the relationship between morphological and controller complexity and task performance. The field can be further expanded to include other trade-offs such as energy consumption. Furthermore, more progress on some established multi-objective versions of NEAT and HyperNEAT will provide investigators with a better understanding of the trade-offs in various problem domains, for example, complex behaviours in MRS.

Therefore an opening exists in the literature surrounding these two issues, specifically in the domain of MRS.

REFERENCES

[1] Hussein A Abbass, Ruhul Sarker, and Charles Newton. 2001. PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, Vol. 2. IEEE, 971–978.

[2] Omer Abramovich and Amiram Moshaiov. 2016. Multi-objective topology and weight evolution of neuro-controllers. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 670–677.

[3] Joshua E Auerbach. 2013. *The Evolution of Complexity in Autonomous Robots*. Ph.D. Dissertation. The University of Vermont.

[4] Joshua E Auerbach and Josh C Bongard. 2010. Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 627–634.

[5] Joshua E Auerbach and Josh C Bongard. 2011. Evolving complete robots with CPPN-NEAT: the utility of recurrent connections. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 1475–1482.

[6] Joshua E Auerbach and Joshua C Bongard. 2012. On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 521–528.

[7] Thomas Back. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

[8] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. 2000. *Evolutionary computation 1: Basic algorithms and operators*. Vol. 1. CRC press.

[9] Karthik Balakrishnan and Vasant Honavar. 1996. On sensor evolution in robotics. In *Proceedings of the 1st annual conference on genetic programming*. MIT Press, 455–460.

[10] Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. 1998. *Genetic programming: an introduction*. Vol. 1. Morgan Kaufmann San Francisco.

[11] Randall D Beer and John C Gallagher. 1992. Evolving dynamical neural networks for adaptive behavior. *Adaptive behavior* 1, 1 (1992), 91–122.

[12] MCDA Juergen Branke. 2002. Opinion Makers Section. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[13] G. Buason, N. Bergfeldt, and T. Ziemke. 2005. Brains, bodies, and beyond: Competitive co-evolution of robot controllers, morphologies and environments. *Genetic Programming and Evolvable Machines* 6(1) (2005), 25–51.

[14] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2009. On-line neuroevolution applied to the open racing car simulator. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2622–2629.

[15] Nicolas Chaumont, Richard Egli, and Christoph Adami. 2007. Evolving virtual creatures and catapults. *Artificial life* 13, 2 (2007), 139–157.

[16] Nick Cheney, Josh Bongard, and Hod Lipson. 2015. Evolving soft robots in tight spaces. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*. ACM, 935–942.

[17] Nicholas Cheney, Jeff Clune, and Hod Lipson. 2014. Evolved electrophysiological soft robots. In *ALIFE*, Vol. 14. 222–229.

[18] CA Coello Coello. 2006. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine* 1, 1 (2006), 28–36.

[19] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International Conference on Parallel Problem Solving From Nature*. Springer, 849–858.

[20] Sabre Didi and Geoff Nitschke. 2016. Neuro-Evolution for Multi-Agent Policy Transfer in RoboCup Keep-Away. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1281–1282.

[21] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E Gusz Eiben. 2015. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI* 2 (2015), 4.

[22] Stephane Doncieux and Jean-Baptiste Mouret. 2014. Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence* 7, 2 (2014), 71–93.

[23] Agoston E Eiben and Marc Schoenauer. 2002. Evolutionary computing. *Inform. Process. Lett.* 82, 1 (2002), 1–6.

[24] Agoston E Eiben, James E Smith, et al. 2003. *Introduction to evolutionary computing*. Vol. 53. Springer.

[25] Dario Floreano, Peter Durr, and Claudio Mattiussi. 2008. Neuroevolution: from architectures to learning. *Evolutionary Intelligence* 1, 1 (2008), 47–62.

[26] Pablo Funes and Jordan Pollack. 1999. Computer evolution of buildable objects. *Evolutionary design by computers* 1 (1999), 387–403.

[27] Avinash Gautam and Sudeept Mohan. 2012. A review of research in multi-robot systems. In *Industrial and Information Systems (ICIIS), 2012 7th IEEE International Conference on*. IEEE, 1–5.

[28] Faustino John Gomez. 2003. *Robust non-linear control through neuroevolution*. Ph.D. Dissertation.

[29] Sameer Gupta and Ekta Singla. 2015. Evolutionary robotics in two decades: A review. *Sadhana* 40, 4 (2015), 1169–1184.

[30] Jonathan D Hiller and Hod Lipson. 2010. Evolving Amorphous Robots.. In *ALIFE*. Citeseer, 717–724.

[31] Gregory Hornby, Al Globus, Derek Linden, and Jason Lohn. 2006. Automated antenna design with evolutionary algorithms. In *Space 2006*. 7242.

[32] Gregory S Hornby and Jordan B Pollack. 2001. Evolving L-systems to generate virtual creatures. *Computers & Graphics* 25, 6 (2001), 1041–1048.

[33] Peter Krcah. 2007. Evolving virtual creatures revisited.. In *GECCO*, Vol. 7. 341–341.

[34] Vignesh Kumar and Ferat Sahin. 2003. Cognitive maps in swarm robots for the mine detection application. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, Vol. 4. IEEE, 3364–3369.

[35] F. Lambercy and J. Tharin. 2013. *Khepera III User Manual: Version 3.5*. K-Team Corporation, Lausanne, Switzerland.

[36] Leo H Langenhoven and Geoff S Nitschke. 2010. Neuro-evolution versus particle swarm optimization for competitive co-evolution of pursuit-evasion behaviors. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–8.

[37] Nicolas Lassabe, Herve Luga, and Yves Duthen. 2006. Evolving creatures in virtual ecosystems. In *Advances in Artificial Reality and Tele-Existence*. Springer, 11–20.

[38] Aristid Lindenmayer. 1968. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of theoretical biology* 18, 3 (1968), 280–299.

[39] Alexandra Mark, Daniel Polani, and Thomas Uthmann. 1998. A framework for sensor evolution in a population of braitenberg vehicle-like agents. In *Artificial Life VI*. 428–432.

[40] Risto Miikkulainen. 2017. Hopfield Network. In *Encyclopedia of Machine Learning and Data Mining*. 625. https://doi.org/10.1007/978-1-4899-7687-1_127

[41] Risto Miikkulainen, Eliana Feasley, Leif Johnson, Igor Karpov, Padmini Rajagopalan, Aditya Rawal, and Wesley Tansey. 2012. *Multiagent Learning through Neuroevolution*. Springer Berlin Heidelberg, Berlin, Heidelberg, 24–46.

[42] Melanie Mitchell. 1998. *An introduction to genetic algorithms*. MIT press.

[43] Tom M Mitchell et al. 1997. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill* 45, 37 (1997), 870–877.

[44] David E Moriarty and Risto Miikkulainen. 1997. Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation* 5, 4 (1997), 373–399.

[45] Jean-Baptiste Mouret and Stéphane Doncieux. 2008. Incremental evolution of animatsâĂŹ behaviors as a multi-objective optimization. In *International Conference on Simulation of Adaptive Behavior*. Springer, 210–219.

[46] Jean-Baptiste Mouret, Sylvain Koos, and Stéphane Doncieux. 2013. Crossing the reality gap: a short introduction to the transferability approach. *arXiv preprint arXiv:1307.1870* (2013).

[47] Stefano Nolfi, Dario Floreano, Orazio Miglino, and Francesco Mondada. 1994. How to evolve autonomous robots. In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, 190–197.

[48] Stefano Nolfi and Domenico Parisi. 1995. Evolving non-trivial behaviors on real robots: an autonomous robot that picks up objects. In *Congress of the Italian Association for Artificial Intelligence*. Springer, 243–254.

[49] Ryoichi Odagiri, Wei Yu, Tatsuya Asai, Osamu Yamakawat, and Kazuyuki Murase. 1998. Measuring the complexity of the real environment with evolutionary robot: Evolution of a real mobile robot Khepera to have a minimal structure. In *Evolutionary Computation Proceedings, 1998*. IEEE, 348–353.

[50] Michael JT O'Kelly and Kaijen Hsiao. 2004. Evolving simulated mutually perceptive creatures for combat. In *Artificial Life IX: Proc. Ninth Intl. Conf. on the Simulation and Synthesis of Life*. 113–118.

[51] Ben Paechter, RC Rankin, Andrew Cumming, and Terence C Fogarty. 1998. Timetabling the classes of an entire university with an evolutionary algorithm. In *International Conference on Parallel Problem Solving from Nature*. Springer, 865–874.

[52] Gary B Parker, Andrey S Anev, and Dejan Duzevik. 2003. Evolving towers in a 3-dimensional simulated environment. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, Vol. 2. IEEE, 1137–1144.

[53] Rolf Pfeifer and Josh Bongard. 2006. *How the body shapes the way we think: a new view of intelligence*. MIT press.

[54] Riccardo Poli, James Kennedy, and Tim Blackwell. 2007. Particle swarm optimization. *Swarm intelligence* 1, 1 (2007), 33–57.

[55] Sebastian Risi and Kenneth O Stanley. 2012. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial life* 18, 4 (2012), 331–363.

[56] Sebastian Risi and Kenneth O Stanley. 2013. Confronting the challenge of learning a flexible neural controller for a diversity of morphologies. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 255–262.

[57] Rui Rocha, Jorge Dias, and Adriano Carvalho. 2005. Cooperative multi-robot systems:: A study of vision-based 3-d mapping using information theory. *Robotics and Autonomous Systems* 53, 3-4 (2005), 282–311.

Table 1: My caption

Risk Condition	Probability
Scope creep in order to maintain a realistic view of the project. Moreover, be cautious when agreeing to extra functionality.	medium Consult the project supervisor based o
Development taking longer than initially expected	 medium
Difficulty adapting framework	low
Impairment of hardware resources	low
Long training time	medium
A team member discontinues their involvement in the project	 low

[58] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

[59] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint* (2017).

[60] Jacob Schrum and Risto Miikkulainen. 2014. Evolving multimodal behavior with modular neural networks in Ms. Pac-Man. , 325–332 pages.

[61] Karl Sims. 1994. Evolving 3D morphology and behavior by competition. *Artificial life* 1, 4 (1994), 353–372.

[62] Karl Sims. 1994. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 15–22.

[63] Tom Smith. 1997. Adding vision to Khepera: An autonomous robot footballer. *Master's thesis, School of Cognitive and Computing Sciences, University of Sussex* (1997).

[64] Nidamarthi Srinivas and Kalyanmoy Deb. 1994. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation* 2, 3 (1994), 221–248.

[65] Kenneth Owen Stanley. 2004. *Efficient evolution of neural networks through complexification*. Ph.D. Dissertation.

[66] Kenneth O Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines* 8, 2 (2007), 131–162.

[67] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life* 15, 2 (2009), 185–212.

[68] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.

[69] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[70] Tim Taylor and Colm Massey. 2001. Recent developments in the evolution of morphologies and controllers for physically simulated creatures. *Artificial Life* 7, 1 (2001), 77–87.

[71] Vito Trianni and Manuel López-Ibáñez. 2015. Advantages of task-specific multi-objective optimisation in evolutionary robotics. *PloS one* 10, 8 (2015), e0136406.

[72] Willem van Willigen, Evert Haasdijk, and Leon Kester. 2013. A multi-objective approach to evolving platooning strategies in intelligent transportation systems. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 1397–1404.

[73] James Watson and Geoff Nitschke. 2015. Deriving minimal sensory configurations for evolved cooperative robot teams. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 3065–3071.

[74] James Watson and Geoff Nitschke. 2015. Evolving Robust Robot Team Morphologies for Collective Construction. In *Computational Intelligence, 2015 IEEE Symposium Series on*. IEEE, 1039–1046.

[75] Xin Yao. 1999. Evolving artificial neural networks. *Proc. IEEE* 87, 9 (1999), 1423–1447.

[76] Jin Zhao and Gang Peng. 2011. NEAT versus PSO for evolving autonomous multi-agents coordination on pursuit-evasion problem. In *Informatics in Control, Automation and Robotics*. Springer, 711–717.

[77] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report* 103 (2001).

[78] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3, 4 (1999), 257–271.