Multi-Robot Systems, Complexity and Multi-Objective Neuroevolution in Evolutionary Robotics: A Review

Alexander Furman
University of Cape Town
Cape Town, South Africa
alex.cleo@gmail.com

ABSTRACT

Evolutionary Robotics employs principles of natural evolution for automatic robot design. The field has opened new doors to previously unapproachable design problems in engineering as well as theoretical questions in the natural and cognitive sciences. State-of-the-art methods combine neuroevolution techniques such as Neuroevolution of Augmenting Topologies with multi-objective evolutionary optimisation algorithms such as Non-dominated Sorting Genetic Algorithm II and Strength Pareto Evolutionary Algorithm II. Current research on the development and use of these methods corroborates the advantages of seeking optimal trade-offs in solving ER design problems, which are often multi-objective optimisation problems. One particularly applicable domain for the use of these methods is the automatic design of control policies and morphological parameterisations for multi-robot systems. In multi-robot systems, robot teams coordinate to complete tasks via complex emergent behaviours which are not easily specified by hand. Furthermore, these tasks often require multiple conflicting criteria to be met (such as low energy expenditure as well as high task performance). This literature review overviews the literature on evolutionary robotics and multi-objective neuroevolution, identifying a literary gap - namely the largely unexplored use of multi-objective neuroevolution for multi-robot system design, as well as lack of quantitative support for various current multi-objective neuroevolution approaches.

1 INTRODUCTION

Evolutionary Robotics (ER) is the development of autonomous robots via artificial evolution [24], for a variety of objectives [22]. As a method of inquiry for research in natural and cognitive science, ER allows researchers to evolve and study artificial agents *in silico*[1], where generations of observable evolution can take place in just hours or minutes. This has opened the door to previously unapproachable topics, notably in evolutionary theory, that lack experimental control in the real world. Such topics, to which ER has already contributed, include the evolution of cooperation [57, 85], communication [27, 89], predator-prey behaviour [59, 62], morphological complexity [5, 6, 12] as well as embodied cognition [17]. In the engineering domain, ER facilitates the automatic design and optimisation of robots for non-linear control tasks where the hand-specification of models and controllers is unfeasible. For many simple control tasks, hand-specified models and controllers are sufficient, such as a linear control law for the common thermostat [31]. Other control problems are characterised by irregularity and large solution space, where the control strategy is often not only impractical to model mathematically, but is not known to the designer at all. In such cases, the methods of ER facilitate the automatic *discovery* of the optimal control strategy [31]. For instance, in multi-robot systems [26] requiring robot teams to coordinate together to complete tasks, ER has been used to produce complex cooperative interactions among team members for collective construction tasks [61, 86] as well as for deriving 'microscopic' individual rules for the emergence of macroscopic collective behaviour in swarm robotics [35, 71]. Neuro-evolution [31, 90] techniques are often the methodical basis for approaching control problems in ER [22], but are typically designed to optimise a *single objective*. However, many optimisation problems, especially in ER, require the optimal trade-off among *multiple objectives*. In this regard, a designer might seek the optimal configuration for a robot team which not only maximises task performance but also minimises energy consumption. Such trade-offs are also studied in the field of evolutionary biology [55, 69]. Why do some organisms have larger brains but smaller bodies? Why do we observe varying degrees of morphological complexity in organisms across different environments and terrains? Explorations of such questions via ER must capture the fact that evolution is largely characterised by trade-offs. As such, *multi-objective neuro-evolution* is an area of interest in the contemporary literature [1].

This paper reviews the literature on ER. The covered works include those which explore the relationships between controller, morphology and other variables, principally in the context of, but not limited to, multi-robot system design. We also review recent implementations of multi-objective neuro-evolution that have been presented - specifically those based on the NEAT algorithm [79].

2 MULTI-OBJECTIVE NEURO-EVOLUTION

This section provides an overview of Neuro-evolution and Evolutionary Multi-Objective Optimisation [92], followed by a review of multi-objective neuro-evolution approaches that incorporate the NEAT [79] method of neuro-evolution.

2.1 Neuro-evolution

Neuro-evolution combines the relative strengths of *Artifical Neural Networks* and *Evolutionary Algorithms* to provide a robust parallel search of the space of candidate network solutions for a given problem. As such, it is an effective controller design technique for non-linear problem spaces where other common approaches fail [31]. This section details the components and generic procedure of neuro-evolution, followed by a review of preeminent neuro-evolution algorithms.

[1]In simulation.

2.1.1 *Artificial Neural Networks.*

Artificial Neural Networks (ANNs) are universal function approximators that take inspiration from biological nervous systems [31]. Structurally, an ANN is a set of connected *neurons* or *computational nodes* which process inputs to produce outputs. Typically, an ANN is *trained* such that *weight values* associated with node edges (which influence the relationship between input and output data) are adjusted until the network outputs sufficiently minimise some *cost function* which describes how close the network's output is to the optimal result.

Each artificial neuron receives input values returned by other artificial neurons, which it passes through a *weighted summation function* and an *activation function* before returning an output (an activation) that can be received by other neurons as input. The *activation function* within each node of an ANN 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 the simplest case, an activation function is binary (that is, the *Heaviside Function*), such that a neuron is either firing an output or not (1 or 0). In practice, more complex activation functions such as *Sigmoid* are preferred, providing a range of smooth outputs that closely reflect changes to weight values in the network. An ANN can be *feedforward* such that data flows exclusively from input to output, or *recurrent* where feedback connections provide the network with information from previous activations.

ANNs have been shown to function well as controllers for various tasks in engineering, especially in robotics [22].

2.1.2 *Traditional Training Methods.*

ANN controllers are commonly trained via *Supervised* and *Reinforcement* based techniques [31]. Supervised Learning methods, such as gradient descent backpropagation [2], require the designer to compile labeled training data (correct input-output pairs), so that the margin of error between the labelled data and the network's outputs during training can be propagated backwards through the network and the weights can be adjusted accordingly for more accurate output. For complex problems where the control policy is not known, the designer often cannot compile labeled training data. In such cases, *Reinforcement Learning* [81], which does not require labelled training data, might be preferred. In this class of training, the ANN controller learns via empirical interaction with the environment to maximise *positive reinforcement feedback* for performing goal-oriented behaviours. However, while this class of methods has proven successful for tasks such as robot arm grasping [45] and single quadrotor obstacle avoidance [91], it tends to fail in the multi-robot domain due to large state space and the *credit assignment problem* [52]. *Unsupervised Learning* is a third class of methods which relaxes the requirements of training data and well-observable environments, relying solely on correlations among the input data. *Neuro-evolution* algorithms conventionally fall into this category [31].

2.1.3 *Controller Neuro-evolution.*

Neuro-evolution is the use of *Evolutionary Algorithms (EAs)* to construct ANNs, and is commonly used to *discover* optimal ANNs for complex control tasks characterised by inaccessible training data and large state space [**?**]. EAs are one generic technique within the family of evolution-inspired global optimisation methods that comprise the field of Evolutionary Computation [24]. Over some number of *generations*, an EA iteratively modifies and combines (via mutation and crossover operators) portions of each of a population of *candidate solutions* (individuals) to some problem or task, until a solution of high enough quality - measured by a *fitness function* that the designer specifies - is discovered. Figure 1 illustrates the generic scheme of an EA, both in principle and in the context of ER.

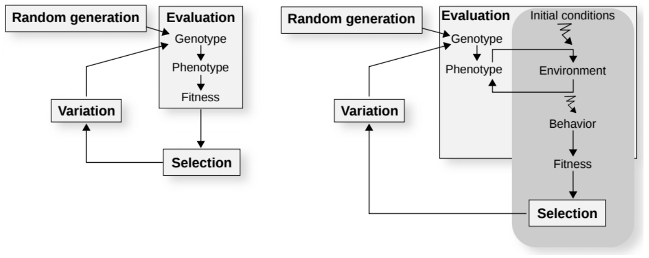

Figure 1: The generic scheme of an Evolutionary Algorithm both in principle (left) and in the context of Evolutionary Robotics (right) [23].

More specifically, the EA applies mutation and crossover to the *genotype* - the underlying encoding - of each candidate solution. The solution encoded by a given genotype is known as a *phenotype* and, in general, this genotype-phenotype mapping is referred to as *representation*.

Different instances of EAs are namely classified according to their representation schemes. One such instance, a *Genetic Algorithm (GA)*, uses strings (traditionally of binary digits) to represent genotypes [24].

The typical neuro-evolution approach exploits the fact that ANNs, especially in the context of controller design, can naturally conform to the genotype-phenotype mapping of a generic GA. In a *direct encoding* scheme, the genotype (i.e. binary string) exactly specifies the phenotype (i.e. the ANN's topology). For example, a *connectivity matrix*, which can be succinctly represented by a binary string, can encode the topology of a network [90]. On the other hand, in an *indirect encoding* scheme, not every detail of the phenotype is specified in the genotype, but can be derived from it. Typically, the genotype specifies *rules* for constructing a phenotype [33]. Rewriting systems such as *Lindenmayer-Systems* [48], where *production* or *replacement* rules are repeatedly applied to some *initial string* to produce complex strings and objects, are often the basis of indirect encodings. While indirect encoding allows for more compact representation than direct encoding, it can be disadvantageous. Indeed, indirect encodings implicitly restrict the phenotype search space to the class of network topologies that their rules expand to, and the general quality of this search space might be suboptimal [79].

Via the iterative process of a GA, a population of genotypes encoding candidate ANNs can be *evolved* [90] to produce an optimal 'solution' controller for some problem, and this process is referred to as *neuro-evolution* [31]. Notably, some neuro-evolution approaches encode just the *weights* to be evolved, leaving topology fixed. *Conventional Neuro-evolution* is one such example, where the encoding scheme is simply a concatenation of the numerical weight values in the network [73]. Other approaches specify both weights and topology in the genotype, allowing both to be evolved [79].

2.1.4 *Neuro-evolution of Augmenting Topologies (NEAT).*

NEAT [79] is a direct-encoding neuro-evolution method which evolves both connection weights and topology, and has been used to evolve ANN controllers in various domains [60]. NEAT is based on three principal ideas: *historical marking*, *speciation* and *complexification.*

Historical Marking (or *innovation numbers*) is an ordered numbering system for new structural innovations during evolution. Whenever a parent genotype is mutated, a record is kept of that mutation. This allows for structural comparison of individual networks in a given population without the need for computationally expensive graph traversal. If two individuals both have a given innovation number in their records, then they both share the topological characteristic corresponding to that innovation number (for example, a connection between *node two* and *node four*). Hence, individual networks can be compared structurally by simply iterating through the records and noting which innovation numbers they have in common. This innovation made NEAT the first approach capable of feasibly evolving topology in addition to weights. Additionally, it provides a solution to the *Competing Conventions Problems* [79], which holds that applying crossover to genotypes which encode the same phenotype results in child networks of poor quality. Cheap topological comparison of networks via historical marking records allows such crossover to be avoided.

Speciation (or *niching*) groups individual ANNs with similar topologies into their own species using a *compatibility function* (which uses historical markers for comparing structure). Each species represents a unique behaviour (or *innovation*) that might be a step towards the optimal solution (controller). Conventionally, crossover between ANNs takes place only within species (i.e. there is no inter-species breeding). A key benefit of speciation is the *protection of new innovations.* Since mutation tends to decrease the fitness of a candidate solution in the short run, the solution should be given time to refine itself over a number of generations, rather than immediately comparing it to older solutions in the broader population with higher fitness but less potential in the long run. This benefit is enabled by the use of *explicit fitness sharing*, which sets each genotype's fitness to the representative fitness of the species it belongs to.

Complexification is the idea that ANN solutions should start off with minimal topologies and grow incrementally, *complexifying* over the evolutionary process and forming a desirable solution. This idea contrasts with the choice of *random starting topologies* which was more common in prior neuro-evolution approaches to NEAT [79]. Thus, each candidate ANN in NEAT starts off minimally with just the specified input nodes, output nodes, and no hidden layers.

2.1.5 *HyperNEAT.*

HyperNEAT [78] is an extension of NEAT. Its key innovation is its ability to take *geometric properties* of the task problem into account during the search for the optimal ANN controller. For example, a controller for a circular robot with sensors on its perimeter could conceivably benefit from information about the physical placement of its sensors, such as how close they are to one another. It might, for instance, yield optimal performance to always activate the two nearest sensors to the current sensor that is detecting something. Such *geometric* information about the actual agent for which a controller is being produced cannot be directly exploited by the conventional NEAT algorithm, which might only realise this information near the final generations of evolution, if at all.

In contrast to NEAT, HyperNEAT uses an indirect encoding scheme based on *Compositional Pattern Producing Networks (CPPNs)* [77]. A CPPN [77] is a function of *n* Cartesian dimensions which outputs a pattern in *n*-dimensional space. CPPNs differ from ANNs internally in that they are connected graphs of *multiple different* activation functions, while ANNs use a single specified activation function. Through this *composition* of functions, complex regular patterns can be produced by querying a CPPN for each point in the Cartesian space *independently*, where each output specifies what should exist at that point in *n*-dimensional space. For a given node pair in the substrate, a connection between that pair in *n*-dimensional space only exists if the weight returned by the CPPN for that pair satisfies a certain threshold. This process is illustrated in Figure 2.

For HyperNEAT to exploit geometric information, the designer first hand-specifies a *substrate* network whose topology resembles the geometry of the task (for example, nodes placed and connected to form a circle, representing a circular robot agent and its sensor placement). The substrate is then mapped onto a Cartesian plane, such that each node has a coordinate, and passing the coordinates for each potential connection to the CPPN yields a weight value for the edge between those nodes. Thus, HyperNEAT is able to choose which connections in the substrate exist in the *n*-dimensional space, as well as set their weights, such that a candidate ANN can be drawn from the space as a function of substrate geometry. In all other respects, HyperNEAT closely resembles the usual NEAT algorithm.

Note that since HyperNEAT uses the NEAT algorithm to evolve CPPNs, it is an extension of the more general *CPPN-NEAT* approach [77] which can be implemented in a number of different ways.

2.2 Evolutionary Multi-objective Optimisation

A multi-objective problem (MOP) consists of multiple (often conflicting) *objectives* that must be optimised simultaneously [20]. A day-to-day example is minimising cost while maximising comfort when purchasing a new car. Solving a MOP typically yields a set of solutions, where each solution represents one possible 'compromise' or trade-off among the objectives. The set of trade-off solutions is generated according to the notion of *Pareto Optimality*, which states that any solution to a MOP is Pareto Optimal if none of the objective functions can be better optimised without degrading another of the objective functions in value. To address the fact that different solutions will perform better for different subsets of the objectives, the concept of *dominance* is used for performance comparison of solutions. A solution x^* *dominates* another solution x if the following conditions hold [23]:

(1) the solution x^* is not worse than x with respect to all objectives;
(2) the solution x^* is strictly better than x with respect to at least one objective.

The set of trade-off solutions exists on the *Pareto Frontier*, and a *Pareto Improvement* is a change to some solution which improves at least one of the objective functions in value without degrading any other objective function [20].

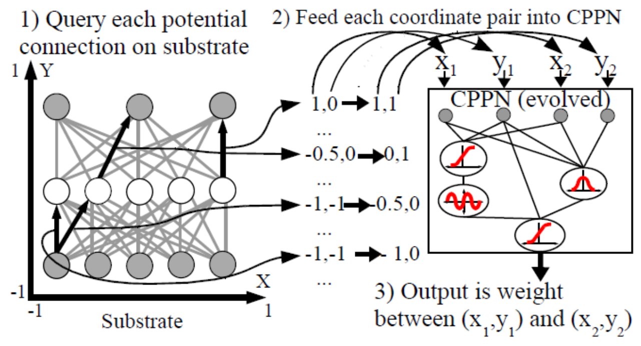
1) Query each potential connection on substrate
2) Feed each coordinate pair into CPPN
1,0 → 1,1
-0.5,0 → 0,1
-1,-1 → -0.5,0
-1,-1 → 1,0
CPPN (evolved)
x_1 y_1 x_2 y_2
3) Output is weight between (x_1,y_1) and (x_2,y_2)
Substrate
Y
X
1
-1
-1
1

Figure 2: *Querying a CPPN with a substrate to produce a connectivity pattern in space.* Each potential connection (node pair) in the *substrate* topology on the left is passed to the CPPN, which calculates a weight for that pair. If the weight computed for a potential connection satisfies a certain threshold, then that connection will be 'painted' in *n*-dimensional space (two dimensional in this instance). Note the presence of different activation functions in the different CPPN nodes, as well as the fact that not all potential connections in the substrate (left) are chosen by the CPPN to exist in the space (right) [67].

Mathematical programming is commonly the basis of methods for solving MOPs [32, 54], but has certain limitations depending on the context of the problem, such as the requirement of differentiable objective functions. EAs provide an alternative basis for solving MOPs, and are desirable in this context as they allow multiple solutions in the Pareto-optimal set to be discovered in a single evolutionary run. The most appropriate trade-off solution in the set can then be established via human decision *after optimisation*. EAs also tend to better handle continuity and concavity on the Pareto Front. Thus, in contrast to the historical trend of mono-objective EA design, the design of *Multi-objective Evolutionary Optimisation Algorithms (EMOAs)* is a contemporary area of interest in the literature [20].

While in mono-objective EAs the objective and fitness functions tend to be identical, multi-objective optimisation requires the fitness function to take multiple objectives into account. Generally, an EMOA uses either *Aggregation-based*, *Criterion-based*, or *Pareto-based* fitness assignment [93]. The remainder of this section covers two Pareto-based state-of-the-art EMOAs: *NSGA-II* [21] and *SPEA-II* [94].

2.2.1 SPEA-II. Zitzler et al. proposed SPEA-II [94] as an improved version of the original *Strength Pareto Evolutionary Algorithm (SPEA)* [95]. As SPEA-II is based on SPEA, we briefly cover SPEA here:

SPEA begins with an initial population of solutions and an empty archive (*external non-dominated set*). At each generation, all non-dominated solutions in the population are copied to the archive. If the archive grows beyond a certain threshold, it is pruned via a clustering technique which preserves the non-dominated front. Each solution in both the population and archive is then assigned a fitness value. In the archive, the fitness value assigned to a solution is known as its *strength value* and represents the number of solutions in the population that it dominates. In the population, the fitness value assigned to a solution is computed by summing the strength values of all solutions in the archive that dominate it. When fitness assignment is completed, the mating phase begins. Binary Tournament Selection [70] is used to select individuals from the union of the population and archive, where each solution in the archive has a higher chance of selection than any solution in the population. Crossover and mutation is applied to solutions in the population, and the resulting population replaces the old population, marking the end of the current iteration [93].

SPEA-II improves on SPEA as follows [94]:

(1) The fitness scheme takes into account, for each solution, both the number of solutions that dominate it and that it is dominated by. Without this improvement, solutions in the population that are dominated by the same archive members will have the same fitness values, and it is possible for every solution in the population to have the same fitness if there is only one solution in the archive.
(2) Incorporation of nearest neighbour density estimation allows for increased efficiency and diversity during the search.
(3) A truncation technique for guaranteeing the preservation of boundary solutions.

2.2.2 NSGA-II. Deb et al. proposed the *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* [21] as the forerunner to NSGA [76], to which it is significantly different. NSGA-II addresses the following weaknesses of NSGA: *computationally complex non-dominated sorting*, *lack of elitism* and *requirement of specified sharing parameter*.

For each solution in the population, NSGA-II determines the solutions that it dominates as well as the solutions it is dominated by (that is, it computes each solution's *non-domination rank*). It then estimates the *crowding distance* for each solution in the population. A solution's crowding distance is the density of solutions surrounding that solution in the population. During selection, both non-domination rank and crowding distance are taken into account. During the application of genetic operators, an *elite* mechanism combines the best parent solutions with the best child solutions.

2.3 Multi-objective Neuro-evolution

Neuro-evolution techniques can be designed or modified to employ EMOAs for the evolution of ANNs. This section reviews recent work in this area, focusing on implementations using the NEAT neuro-evolution algorithm. A notable challenge in this domain is the preservation of core NEAT innovations, such as speciation, when EMOAs (commonly NSGA-II and SPEA-II) are designed to control the main evolution loop and perform selection themselves.

A NEAT-like algorithm was first combined with an EMOA in the implementation of *MM-NEAT* by Schrum and Miikkulainen for evolving complex non-player-character behaviour in a video game [74]. A modified version of NSGA-II was employed and conventional features of NEAT, including crossover, speciation and fitness sharing, were excluded in the implementation. Rather, for crossover a similar method to the $(\mu + \lambda)$ strategy [7] was employed, and a custom elitist mechanism was used for selection.

Other approaches have focused on directly extending NEAT to support an EMOA while attempting to preserve its core innovations [1, 6, 16, 41, 80, 83].

NEAT-PS [83] incorporates the Pareto Strength approach used in SPEA-II [94] into NEAT, computing a single scalar fitness value for each candidate solution and passing that value to NEAT. While the implementation was found to perform well on a task requiring a Pareto front of autonomous passenger driving preferences to be produced, it should be noted that by transforming the multi-objective performance vector into a scalar score, NEAT-PS cannot guarantee monotonic evolution of the different objectives [1].

Auerbach and Bongard [6] extend CPPN-NEAT [77] to support multiple objectives by replacing the standard speciation mechanism for selection with NSGA-II. To account for the removal of speciation, the implementation included a genotypic diversity objective based on the compatibility function used for NEAT speciation. Like speciation in NEAT, the genotypic diversity objective allowed solutions in different areas of the solution space to evolve without directly competing with all other solutions, and was found to perform well. Szerlip and Stanley [80] and Lehman et al. [41] similarly combine NEAT and NSGA-II for multi-objective support using diversity objectives. However, these works focused on diversity in the behaviour space[2] rather than the genotype space.

Another more recent behaviour-focused approach which combines elements of NEAT and NSGA-II, and does not use genotypic-diversity based speciation nor crossover, is the *Map-Elites* algorithm [58].

More recent efforts have focused on the preservation of NEAT's conventional speciation mechanism [1, 16].

Cheney et al. [16] implement a multi-objective version of NEAT which is reportedly the first such implementation to preserve speciation. Instead of removing or replacing speciation and adding a genetic diversity objective, this implementation performs a Pareto ranking of the solutions within each NEAT species, followed by traditional Tournament Selection, as in SPEA-II [94], for selection based on each ranking. However, the authors neglect to provide a thorough treatment of, nor any quantitative support for, the algorithm.

Abramovich and Moshaiov present another approach, *NEAT-MODS*[3], to multi-objective NEAT which includes speciation [1]. The approach is largely analogous to NEAT with minor adjustments to accommodate for multi-objective optimisation. Namely, NEAT and NEAT-MODS differ at the *selection* phase. NEAT-MODS performs selection in the following two phases: *(1) Selecting Species for Next Generation* and *(2) Selecting Individuals from Selected Species*. The first phase uses crowd distance and non-domination rank based sorting (as in NSGA-II) as part of the species selection process. In the second phase, each selected species is sorted internally (again using crowd distance and rank based sorting). Individual selection from each species is then performed using a *serial species progression* scheme. This scheme iteratively selects the first individual in each sorted species, after all of which it selects the second individual in each species, and so on (note that the algorithm accounts for species having different lengths). As such, the scheme produces a diverse set of trade-off solutions without sacrificing the benefits of speciation. Intra-species reproduction takes place as in conventional NEAT, except that the superiority of the better parent is defined by an index in the sorted species that they both belong to. NEAT-MODS was tested on a single-robot (*Khepera* [56]) sensor-based obstacle avoidance task, and was found to produce a wider-accumulated pareto-front than NEAT-PS [83]. We note, however, that NEAD-MODS and NEAT-PS have not been directly compared in a mutual task domain.

[2]See *Novelty Search* [42].

[3]'MODS' stands for 'Multi-Objective Diversified Species'

3 EVOLUTIONARY ROBOTICS

The reviewed works in ER are categorised according to three fundamental design patterns in ER research: *controller evolution*, *morphology evolution*, and *the simultaneous evolution of controller and morphology*. That is, the first section covers research in which controller design is left to artificial evolution while morphology is hand-designed and fixed, the second section covers research which focuses on the evolution or automatic design of morphology or substrate, and the final section treats the concurrent evolution of controller and morphology.

3.1 Evolution of Control

Following from seminal work in 1992 by Beer and Gallagher in which genetic algorithms were successfully used to evolve artificial agent controllers for simple locomotion tasks [10, 19, 46], evolution of controllers for robots with fixed morphologies was a pervasive theme in the literature for much of the 1990s. For instance, a 1992 paper by Franceschini et al. [28] used a genetic algorithm to evolve the connection weights for a fixed-morphology six-legged insect like robot for a simple locomotion task, and a 1994 paper by Floreano et al. [27] used a genetic algorithm to evolve a physical Khepera-like [56] robot for a navigation and obstacle avoidance task. The applicability of genetic algorithms to multi-robot system design was also a growing area of interest at this time. In 1997, for instance, Jeong and Lee [38] used a genetic algorithm to evolve cooperative behaviour in a multi-robot system for a soccer playing task. Moreover, various works have used neuro-evolution to evolve collective behaviour in multi-robot systems. Nitschke et al. [63], for instance, use CONE (collective neuro-evolution) to evolve collective behaviour for a multi-rover task requiring sensory detection of 'points of interest' in a virtual environment.

Recently, Watson and Nitschke [87] used HyperNEAT to investigate the relationship between the complexity of fixed robot sensory configurations and evolved controllers in yielding optimal task performance for a multi-robot collective construction task. While the robot team was homogenous in terms of both controller and morphology during each of the six experiments, ANN controllers were evolved using HyperNEAT and a set of sensory configurations (morphologies) which varied in complexity (namely number and range of sensors) were manually specified by the researchers. In each experiment, a controller was evolved for one of the sensory configurations. It was found that minimal sensory configurations were optimal for evolving collective team behaviours that maximised task performance; namely, increasing morphological complexity did not increase task performance. While Watson and Nitschke [88] recently extended this work by evolving both controller and sensory configuration[4], they have not yet explored the potential benefits of team heterogeneity, which has been found to produce better performance [61].

Outside the domain of multi-robot systems, van Willigen et al. [83] implemented *NEAT-PS* to evolve autonomous vehicle controllers while selecting for both *speed* and *comfort* [84]. This would allow vehicle drivers to toggle their driving preferences in real-time, depending on whether their current preference is *speed* (ie: many lane changes, frequent changes in velocity) and *comfort* (fewer lane changes, constant velocity). Results indicated that NEAT-PS is suitable for evolving a set of controllers that perform well on different areas of the Pareto Front, providing a range of controllers for different 'driving preference settings'.

Risi and Stanley [68] evolved controllers for simulated *quadrupeds* using HyperNEAT. Specifically, Risi and Stanley focus on the *transferability* of evolved controllers to different morphologies. In the experiment, a set of hand-designed morphologies (namely differing by leg length) are passed as substrate inputs to HyperNEAT, such that the evolved CPPN functions as a quadruped controller which performs well on a diverse set of never-seen morphologies. Results showed that HyperNEAT is indeed capable of producing effective controllers for never-seen morphologies. An interesting extension of this work would be to evolve both morphology and controller simultaneously instead of evolving controllers with hand-designed morphologies, in which case multi-objective optimisation might have some benefit (such as to select for both morphological simplicity and controller task performance). While no such extension of this particular experiment has been conducted, similar experiments are reviewed in Section 3.3.

3.2 Evolution of Morphology

Seminal work by Karl Sims in 1994[5] demonstrated the use of artificial evolution for both controller and morphology [75], in lieu of the fixed-morphology trend that had prevailed hitherto [34]. However, the caliber of parallelised computing resources used by Sims at the time was largely inaccessible to researchers in the field, and so it was only several years later that comparable results were attained [3]. Nevertheless, the work of Sims inspired new interest in the evolution of morphology alone and, more generally, was seminal in establishing the field of Artificial Life [3].

Studies have for the most part dealt with the evolution of *rigid structures*. Early work by Funes and Pollack [29], for instance, demonstrated the evolution of solid structures, such as crane arms and bridges, made out of LEGO bricks. Parker et al. [65] conducted similar research on the evolution of LEGO block-based solid structures, and both studies contributed to later work on the evolution of LEGO robot body morphologies with a fixed controller for a wheeled locomotion task [66]. Similar work for a robot motion task was conducted by Lichtensteiger and Eggenberger [47], where evolution parameterised[6] the positions of 16 light sensors on the body of a fixed-controller robot. Specifically, the motion task required the robot to avoid some obstacles but purposely collide with others.

[4]We cover this extended work in Section 3.3

[5]We elaborate on the work of Sims in section 3.3.

[6]We place more focused attention on parametric evolution in Section 3.3, namely in comparison to *topological* evolution.

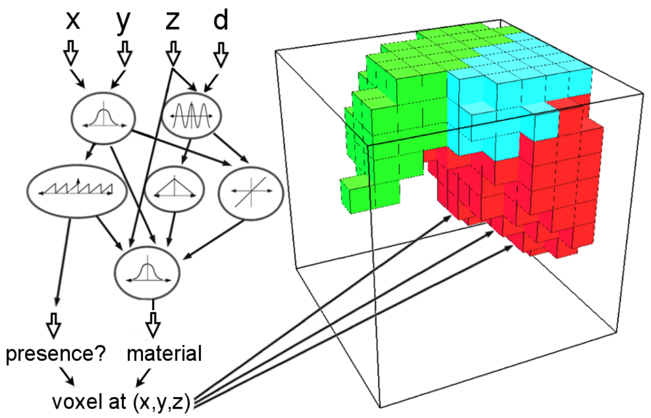

Figure 3: *Constructing morphology with a CPPN.* The CPPN (left) is queried for each *potential* voxel in the hypercube (right), such that the output of the CPPN for some potential voxel defines the presence of that voxel in the hypercube. If the CPPN output for some voxel falls above a certain threshold, that voxel is excluded from the hypercube. In this example, the different colours represent different *material types* for voxels in the hypercube, as determined by the output of the CPPN. Note how the CPPN's exclusion of potential voxels (that fell above the threshold) from the hypercube allows included voxels to accumulate and form morphological structure [18].

Recent studies have employed instances of CPPN-NEAT [77] for the evolution of morphology. While HyperNEAT, one such instance of CPPN-NEAT, is principally concerned with the evolution of *controllers* that exploit morphological geometric properties, CPPN-NEAT can also be implemented to construct organisms and structures as a function of *voxels* (arrays of three-dimensional coordinates) that are passed to it. In general, a bounded area (fully comprised of potential voxels) can be defined, such that passing a potential voxel to the CPPN produces an output which defines the presence of that voxel in *hypercube* space. This process is illustrated in Figure 3.

Auerbach and Bongard [4], for instance, use CPPN-NEAT to evolve rigid robot morphologies that capture relationships between physical structure and controller. This was conducted as groundwork for later research in which both controller and morphology are evolved [5].

A developing trend in the contemporary literature is work on the evolution of *soft* rather than *rigid* structures. Hiller and Lipson [36] conduct the first instance of such research, evolving amorphous[7] robots, using CPPN-NEAT, that are able to locomote via the expansion and contraction of the materials that morphologically comprise them. This research was a step forward for the use of ER in modeling and exploring the continuous properties of biological systems, possibly paving the way for novel methods of inquiry into classical evolutionary theory and other topics. Additionally, while controller evolution has traditionally been the focal point of emerging behaviours in ER, this research demonstrates that interesting

[7]Soft

behaviours can emerge from the interaction between morphology and environment.

In a 2014 paper, Cheney et al. [17] use CPPN-NEAT to evolve soft *electrophysiological* robots where behaviour is actuated principally by electrical signals charging through the tissue cells of the soft morphology, rather than by a controller. Interesting and complex behaviours were found to emerge directly from the morphologies of the robots, such that *control* was interlaced with physical structure. We note one potentially crucial limitation of this work: Namely, the researchers specified a fitness function for CPPN-NEAT which incentivised both *minimising amount of conductive tissue* as well as *maximising distance traveled* for a simple locomotion task. Thus, *two objectives* were being optimised, yet a *single-objective* implementation of CPPN-NEAT was employed. Henceforth, a valuable extension of this work could employ a multi-objective implementation of CPPN-NEAT for finding the optimal trade-off between the objectives.

Cheney et al. conduct a similar study of morphology-actuated behaviour in soft robots in a 2015 paper which evolves morphology for a task requiring evolved robots to escape from partially-enclosed boxes via tight apertures [16]. Results indicated high suitability of the evolved soft robots to the aperture escape task. The researchers implemented a novel multi-objective implementation of CPPN-NEAT for the experiment, selecting for both *maximum number of voxels outside the enclosed box* (ie: squeezing as much of the morphology outside of the box as possible) and *maximum morphology size* (ie: evolving creatures that were as large as possible).

3.3 Control and Morphology

This section reviews research on the evolution of both controller and morphology. Namely, we differentiate between *parametric* and *topological* evolution [3]. Parametric evolution is principally concerned with the *configuration* of a morphology model, such as the placement of sensors on an otherwise fixed morphology. On the other hand, topological evolution evolves the *complete* morphology. While the seminal work of Sims [75] on the *topological* evolution of both organism brain (controller) and body (morphology) was only accurately reproduced within several years of being presented, research on *parametric evolution* was sooner in motion [3].

3.3.1 *Parametric Evolution of Control and Morphology.*

Much of the work in this domain has dealt with the evolution of sensory parameters, such as the number, placement and sight range of robot sensors.[8]

In a 1996 paper by Balakrishnan and Honavar [8], the sensor placements and sight ranges for a simulated Khepera-like [56] robot with a fixed number of sensors (n=8) were evolved. Both the robot and a set of 'boxes' were placed in a two-dimensional grid environment, and the robot was tasked with pushing the boxes to the edges of the environment. In addition to the sensory parameters, the connection weights for an ANN controller with a mostly fixed topology was evolved for the box-clearing task. During evolution of sensor positions, a *sensor switch* was implemented such that the mutation operator - with extremely low probability - could disable sensors. A particularly interesting result was the observation that, even without a cost for having extra sensors - the highest performing individuals always disabled some of the sensors. This indicated a potential strength of having fewer sensors over the complete available set, which could improve the cost-effectiveness and performance of real life implementations (that is, physical robots).

Similar work was conducted by Mark et al. [51] in 1998, also incorporating simulated Khepera-like robots. Morphologically, the robots were circular with simple vision sensors and two wheels for movement. Evolution was used both for the production of network controllers as well as for the parameterisation of sensory view angles and number of sensors. The evolved robots were evaluated on two separate tasks. The first task placed a set of robots in an environment containing obstacles and 'lamps'. Each robot had an initial 'energy level' which was spent by moving and interacting with other robots and obstacles, and gained by dwelling in the light of a lamp. Additional energy was spent if two robots collided, and a robot 'died' if its energy level fell below a threshold. The second task required a single robot to navigate obstacles in order to reach a target. Despite computational limitations rendering mostly inconclusive results, other researchers have since conducted similar work. For instance, in an experiment which evolved Khepera sensory parameters and the weights for a fixed topology ANN controller, Buason et al. [15] simulated a predator/prey situation to investigate the impact of variable sensory parameters on evolution. Notably, a trade-off between speed and view angle was discovered, such that the *prey* robots preferred speed over vision.

Recently, Watson and Nitschke [88] investigated the evolution of both sensory parameters and control for a collective construction task, extending their work in [87] which evolved control but not morphology. HyperNEAT was used for evolution, and results demonstrated that evolution is capable of producing optimal sensory paramaterisations as well as cooperative behaviour for the specified collective construction task. We note that the only morphological characteristic which was evolved was *number of sensors* rather than placement or range of sight. Additionally, the authors have yet to explore the use of heterogenous (robots with different configurations) rather than homogenous (robots with universal configuration) teams in this ongoing work.

3.3.2 *Topological Evolution of Control and Morphology.*

Sims [75] presented a system for evolving virtual creatures capable of behaviours such as swimming jumping and walking. Creatures were evaluated together on a 'box grabbing competition', and each was morphologically composed of solid cuboid body segments connected by different joint types (such as 'twist' and 'rigid'). Neural controllers, which were co-evolved with morphology using a genetic algorithm (incorporating a graph-based genotype representation), actuated the joints by sending specific output values to simulated muscles (effectors). Additionally, three types of sensors were implemented: joint-angle sensors for proprioception, contact sensors for collision detection, and photosensors. Additionally, unlike for the case of traditional ANNs, each node in the evolved networks could be activated by a range of functions, as in the case of CPPNs [77].

The first successful reproduction of Sims' work on standard hardware was by Taylor and Massey [82] in 2000, where the controllers

[8]However, there has also been work on other morphological parameters such as body size, wheel radii and joint ranges [25, 50].

and morphologies of virtual creatures were evolved for swimming and locomotion tasks. Another reproduction of the work was presented in 2004 [64], in which spheres were used instead of cuboids and the performance task was for virtual creatures to 'fight' one another such that they touched the root nodes of their opponents before their own root nodes were touched.

Krah [39] implemented a system based on Sims' graph-encoding which used an extension of NEAT to evolve creatures for tasks such as swimming, jumping, walking and phototaxis. Lehman and Stanley [44] extended this work, using multi-objective novelty search [43] to evolve a diverse range of creatures capable of locomotion, rather than converging to a single creature (optimum).

In addition to graph-based encodings based on the work of Sims[9], other encodings have been explored for the topological evolution of controller and morphology. One such example is the work of Hornby and Pollack [37], which employed the first use of L-Systems for the co-evolution of controller and morphology. Other encoding approaches which have been used include direct encoding [13, 49] and developmental encoding [11, 14].

In a recent study employing CPPNs for encoding, Auerbach and Bongard [6] conduct an ER-based inquiry of the *arrow of complexity* hypothesis [9], which posits that the complexity of an organism tends to increase with evolutionary time. Specifically, the authors investigate why this positive relationship between morphological complexity and evolutionary time is not always observed (why, for instance, do single-celled organisms still exist at this point in evolutionary time?).

Two sets of experiments were conducted, each of which evaluated evolved organisms on a locomotion task in fifty different environments of varying complexity. For the first set of experiments, the virtual organisms were evolved using a single-objective implementation of CPPN-NEAT, where evolution only selected for locomotive ability. For the second set of experiments, a multi-objective implementation of CPPN-NEAT was used to evolve virtual organisms, where evolution selected for both locomotive ability and *simple morphology*, thereby incurring a cost on morphological complexity.

Results of the experiments showed that virtual organisms become more complex with evolutionary time if there is *no cost* incurred on being morphologically complex, regardless of whether the environment is complex or simple. However, *incurring a cost* on being morphologically complex causes organisms to *remain simple in simple environments* over evolutionary time, but *become more complex in complex environments* over evolutionary time. These results corroborate the conjecture that the arrow of complexity might only occur when a complex morphology is necessary to survive or perform well in the environment. More generally, this work is a seminal example of the applicability of ER to the field of biological evolution.

4 CONCLUSIONS

In general, the field of Evolutionary Robotics (ER) has broadened considerably since its inception in the 1990s. On the engineering side of the field, the general focus has shifted from evolving controllers (brains) for individual robots to the evolution of both robot controllers and morphological (body) parameterisations, namely for use in *multi-robot systems* where the emergence of cooperative team behaviour facilitates task completion. From the perspective of the natural sciences, the principal use of ER has shifted from reproducing the first seminal works in the field of Artificial Life to exploring questions in domains such as evolutionary biology [55] and embodied cognition [30].

Current research on the development and use of state-of-the-art multi-objective evolutionary methods for ER has demonstrated the benefit of finding *optimal trade-offs* for multi-criteria problems rather than converging to a single optimum. On one hand, this is enabling designers to evolve cheaper and better-performing task-completing robots, such as by off-loading complexity of the morphology onto the controller and vice-versa. On the other hand, it is enabling scientists to investigate evolutionary trade-offs and questions of complexity that are observed in nature.

However, we notice gaps in this body of the literature. Firstly, these state-of-the-art methods, namely multi-objective implementations of widely-used neuro-evolution algorithms, are often sparsely documented and tend to lack quantitative support. Secondly, these methods are seldom applied to multi-robot systems in which they are potentially highly applicable.

As such, avenues for future research could include placing greater detail on quantifying existing and novel approaches to multi-objective neuro-evolution, as well as applying these approaches to a wider variety of applicable fields, pertinent examples of which are multi-robot systems [26] and swarm robotic systems [72].

[9]For additional graph-based encoding re-implementations of Sims' work, see [40, 53]

REFERENCES

[1] 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.

[2] Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing* 5, 4-5 (1993), 185–196.

[3] Joshua E Auerbach. 2013. *The Evolution of Complexity in Autonomous Robots*. Technical Report. 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 Josh C Bongard. 2014. Environmental influence on the evolution of morphological complexity in machines. *PLoS computational biology* 10, 1 (2014), e1003399.

[7] Thomas Back, Frank Hoffmeister, and Hans-Paul Schwefel. 1991. A survey of evolution strategies. In *Proceedings of the fourth international conference on genetic algorithms*, Vol. 2. Morgan Kaufmann Publishers San Mateo, CA.

[8] 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.

[9] Mark A Bedau. 1998. Four puzzles about life. *Artificial life* 4, 2 (1998), 125–140.

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

[11] Josh Bongard. 2002. Evolving modular genetic regulatory networks. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, Vol. 2. IEEE, 1872–1877.

[12] Josh Bongard. 2011. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences* 108, 4 (2011), 1234–1239.

[13] Josh C Bongard and Chandana Paul. 2000. Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In *From Animals to Animats: The Sixth International Conference on the Simulation of Adaptive Behaviour*. Citeseer.

[14] Josh C Bongard and Rolf Pfeifer. 2003. Evolving complete agents using artificial ontogeny. In *Morpho-functional Machines: The new species*. Springer, 237–258.

[15] Gunnar Buason, Nicklas Bergfeldt, and Tom 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.

[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] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. 2013. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 167–174.

[19] Dave Cli, Philip Husbands, and Inman Harvey. 1993. Evolving visually guided robots. In *Meyer, JA., HL Roitblatt, and SW Wilson (1993) From Animals to Animats2. Proceedings of the Second International Conference on Simulation of Adaptive Behavior. MIT Press/Bradford Books, Cambridge Ma.* Citeseer, 374–383.

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

[21] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[22] 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.

[23] 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.

[24] A. Eiben and J. Smith. 2003. *Introduction to Evolutionary Computing*. Springer, Berlin, Germany.

[25] Ken Endo, Takashi Maeno, and Hiroaki Kitano. 2002. Co-evolution of morphology and walking pattern of biped humanoid robot using evolutionary computation. Consideration of characteristic of the servomotors. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, Vol. 3. IEEE, 2678–2683.

[26] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. 2004. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34, 5 (2004), 2015–2028.

[27] Dario Floreano, Sara Mitri, Stéphane Magnenat, and Laurent Keller. 2007. Evolutionary conditions for the emergence of communication in robots. *Current biology* 17, 6 (2007), 514–519.

[28] Nicolas Franceschini, Jean-Marc Pichon, and Christian Blanes. 1992. From insect vision to robot vision. *Phil. Trans. R. Soc. Lond. B* 337, 1281 (1992), 283–294.

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

[30] Arezou Ghane and Kate Sweeny. 2013. Embodied health: A guiding perspective for research in health psychology. *Health Psychology Review* 7, sup1 (2013), S159–S184.

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

[32] Salvatore Greco, J Figueira, and M Ehrgott. 2005. Multiple criteria decision analysis. *Springer's International series* (2005).

[33] Frederic Gruau. 1993. Genetic synthesis of modular neural networks. In *Proceedings of the 5th International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., 318–325.

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

[35] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. 2009. Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots* 26, 1 (2009), 21–32.

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

[37] Gregory S Hornby and Jordan B Pollack. 2001. Body-brain co-evolution using L-systems as a generative encoding. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 868–875.

[38] Il-Kwon Jeong and Ju-Jang Lee. 1997. Evolving cooperative mobile robots using a modified genetic algorithm. *Robotics and Autonomous Systems* 21, 2 (1997), 197–205.

[39] Peter Krah. 2008. Towards efficient evolution of morphology and control. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 287–288.

[40] Nicolas Lassabe, Hervé Luga, and Yves Duthen. 2007. A new step for artificial creatures. In *Artificial Life, 2007. ALIFE'07. IEEE Symposium on*. IEEE, 243–250.

[41] Joel Lehman, Sebastian Risi, David DâĂŹAmbrosio, and Kenneth O Stanley. 2013. Encouraging reactivity to create robust machines. *Adaptive Behavior* 21, 6 (2013), 484–500.

[42] Joel Lehman and Kenneth O Stanley. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. 329–336.

[43] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[44] Joel Lehman and Kenneth O Stanley. 2011. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 211–218.

[45] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[46] M Anthony Lewis, Andrew H Fagg, and Alan Solidum. 1992. Genetic programming approach to the construction of a neural network for control of a walking robot. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 2618–2623.

[47] Lukas Lichtensteiger and Peter Eggenberger. 1999. Evolving the morphology of a compound eye on a robot. In *Advanced Mobile Robots, 1999.(Eurobot'99) 1999 Third European Workshop on*. IEEE, 127–134.

[48] 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.

[49] Hod Lipson and Jordan B Pollack. 2000. Automatic design and manufacture of robotic lifeforms. *Nature* 406, 6799 (2000), 974.

[50] Henrik Hautop Lund, John Hallam, and Wei-Po Lee. 1997. Evolving robot morphology. In *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE, 197–202.

[51] 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.

[52] Maja J Matarić. 1997. Reinforcement learning in the multi-robot domain. In *Robot colonies*. Springer, 73–83.

[53] Thomas Miconi and Alastair Channon. 2005. A virtual creatures model for studies in artificial evolution. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, Vol. 1. IEEE, 565–572.

[54] Kaisa Miettinen. 1999. Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science. (1999).

[55] William A Mitchell and Thomas J Valone. 1990. The optimization research program: studying adaptations by their function. *The Quarterly Review of Biology* 65, 1 (1990), 43–52.

[56] Francesco Mondada, Edoardo Franzi, and Paolo Ienne. 1994. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental robotics*

III. Springer, 501–513.

[57] Jean-Marc Montanier and Nicolas Bredeche. 2011. Surviving the tragedy of commons: Emergence of altruism in a population of evolving autonomous agents. In *European Conference on Artificial Life*.

[58] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[59] Geoff Nitschke. 2003. Emergence of Cooperation in a Multiple Predator, Single Prey Game.. In *FLAIRS Conference*. 234–238.

[60] G Nitschke. 2012. Behavioral heterogeneity and collective construction. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 387–394.

[61] Geoff S Nitschke, AE Eiben, and Martijn C Schut. 2012. Evolving team behaviors with specialization. *Genetic Programming and Evolvable Machines* 13, 4 (2012), 493–536.

[62] Geoff S Nitschke and Leo H Langenhoven. 2010. Neuro-evolution for competitive co-evolution of biologically canonical predator and prey behaviors. In *Nature and Biologically Inspired Computing (NaBIC), 2010 Second World Congress on*. IEEE, 546–553.

[63] Geoff S Nitschke, Martijn C Schut, and AE Eiben. 2010. Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task. *Evolutionary Intelligence* 3, 1 (2010), 13–29.

[64] Michael JT OâĂŹKelly 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.

[65] 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.

[66] Gary B Parker, Dejan Duzevik, Andrey S Anev, and Ramona Georgescu. 2007. Morphological evolution of dynamic structures in a 3-dimensional simulated environment. In *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*. IEEE, 534–540.

[67] Sebastian Risi and Kenneth O Stanley. 2011. Enhancing es-hyperneat to evolve more complex regular neural networks. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 1539–1546.

[68] 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.

[69] DA Roff and DJ Fairbairn. 2007. The evolution of trade-offs: where are we? *Journal of evolutionary biology* 20, 2 (2007), 433–447.

[70] Elizabeth M Rudnick, Janak H Patel, Gary S Greenstein, and Thomas M Niermann. 1994. Sequential circuit test generation in a genetic algorithm framework. In *Design Automation, 1994. 31st Conference on*. IEEE, 698–704.

[71] Erol Şahin. 2004. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*. Springer, 10–20.

[72] Erol Şahin. 2004. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*. Springer, 10–20.

[73] J David Schaffer, Darrell Whitley, and Larry J Eshelman. 1992. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Combinations of Genetic Algorithms and Neural Networks, 1992., COGANN-92. International Workshop on*. IEEE, 1–37.

[74] Jacob Schrum and Risto Miikkulainen. 2008. Constructing Complex NPC Behavior via Multi-Objective Neuroevolution. *AIIDE* 8 (2008), 108–113.

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

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

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

[78] 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.

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

[80] Paul A Szerlip and Kenneth O Stanley. 2013. Indirectly Encoded Sodarace for Artificial Life.. In *ECAL*. 218–225.

[81] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[82] 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.

[83] 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.

[84] Willem H van Willigen, Evert Haasdijk, and Leon JHM Kester. 2013. Evolving intelligent vehicle control using multi-objective neat. In *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2013 IEEE Symposium on*. IEEE, 9–15.

[85] Markus Waibel, Dario Floreano, and Laurent Keller. 2011. A quantitative test of Hamilton's rule for the evolution of altruism. *PLoS biology* 9, 5 (2011), e1000615.

[86] Markus Waibel, Laurent Keller, and Dario Floreano. 2009. Genetic team composition and level of selection in the evolution of cooperation. *IEEE Transactions on Evolutionary Computation* 13, 3 (2009), 648–660.

[87] 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.

[88] 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.

[89] Steffen Wischmann, Dario Floreano, and Laurent Keller. 2012. Historical contingency affects signaling strategies and competitive abilities in evolving populations of simulated robots. *Proceedings of the National Academy of Sciences* 109, 3 (2012), 864–868.

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

[91] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. 2016. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 528–535.

[92] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1, 1 (2011), 32–49.

[93] Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. 2004. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation*. Springer, 3–37.

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

[95] Eckart Zitzler and Lothar Thiele. 1998. An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK-report* 43 (1998).