A Survey on Evolutionary-aided Design in Robotics

Shanker G. R. Prabhu*, Richard Seals, Peter Kyberd and Jodie Wetherall

Department of Engineering Science, Faculty of Engineering & Science, University of Greenwich, Central Avenue, Chatham Maritime, Kent ME4 4TB, UK.

SUMMARY
The evolutionary-aided design process is a method to find solutions to design and optimisation problems. Evolutionary Algorithms (EAs) are applied to search for optimal solutions from a solution space that evolves over several generations. EAs have found applications in many areas of robotics. This paper covers the efforts to determine body morphology of robots through evolution and body morphology with the controller of robots or similar creatures through co-evolution. The works are reviewed from the perspective of how different algorithms are applied and includes a brief explanation of how they are implemented.

KEYWORDS: Evolutionary robotics; Evolutionary-aided design; Morphology evolution; Co-evolution;

1. Introduction

Evolutionary Computation (EC) stems from Darwin's theory of evolution [1] and Mendel's experiments in hybridising plants [2]. The concept utilises various mechanisms of evolutionary theory to stochastically evolve a population of solutions. Evolutionary Algorithms (EAs) are a subset of EC introduced in the 1970s by Holland [3] as Genetic Algorithms (GA). GAs use a basic structure that changes over time by minor variations made at each time step. At the same time, other researchers developed conceptually similar algorithms like Evolution Strategies (ES) or Genetic Programming (GP). It was almost two decades before they were successfully applied to robotics [4]. Evolutionary Robotics (ER) is concerned with the generation of autonomous robots using the principles of evolutionary computing [5]. EC strategies are applied to autonomous robot development in evolving morphologies and controllers separately or co-evolving both simultaneously [5].

Even though ER is over 20 years old, rarely has it been used to generate a physical demonstration of evolved robots. Nichele estimated that more than 95% of the literature focussed on the evolution of robot controllers. In the rest, less than 1% reported works that physically tested the generated morphology and controller and the remaining 4% only simulated the co-evolution process [6]. Although EAs are known to converge to solutions that are often unintuitive and unconventional (when compared to the results of traditional design methods), it must be noted that applying EAs mean incorporating many variables and parameters. Consequently, different combinations lead to different results which rapidly drift away from optimal solutions [7].

In ER, the applications of EAs can be spread across various domains from traditional mobile and manipulator robotics to newer areas such as; modular robotics, swarm robotics, biorobotics, developmental robotics and soft robotics [8]. This paper reports the evolution

* Corresponding author. E-mail: s.prabhu@gre.ac.uk

of the morphology alone along with the co-evolution of morphology and controller in the traditional areas of robotics.

There have been several surveys covering various aspects of ER. Starting with a discussion of such articles, we review publications that discuss the evolution of physical creatures with or without their control system design with an emphasis on algorithmic variations and application areas along with a discussion on evolved buildable robots.

In 2001, Taylor *et al.* [9] reported on the software packages utilised during the evolution of controller and morphology of virtual creatures. Later, Floreano *et al.* [10] briefly examined evolving morphologies and Lipson [11] explained how ER was used to perform open-ended design automation, while covering a few of the important works involved in robot body and controller design in addition to robot body only design. ER's effect in the area of biologically inspired robots is described in [12]. Evolutionary principles are used in robotics for automatic parameter tuning, designing, online learning and automatic synthesis are explained in [4]. The review article [8] briefly covers different application areas of ER along with the various challenges faced.

Recent publications; Winfield *et al.* [13] and Nolfi *et al.* [14] discuss the challenges faced by the community by analysing the key publications which report robot morphology development and robot body and controller co-evolution. The applications of EC principles in the field of robotics is broadly covered by Eiben *et al.* [15]. A recent review paper by Gupta *et al.* [16] touches on the main applications of EC in robotics for controller only evolution, morphology only evolution and their co-evolution.

Nelson *et al.* [17] surveyed fitness functions used in evolutionary robotics. Papers [18, 19] contain a study on methods used for controller only design. Kicinger *et al.* [20] compiled a comprehensive list of publications that reported the use of EA for fixed structural design. There are more recent surveys on swarm [21], modular [22, 23] and soft [24] robotics. Other articles published which discuss ER are [25-32].

While several publications explain the intricacies of the area, to the best of our knowledge, none of them cover all reported attempts to deal with the applications of EAs in robot or similar creature's morphology design and optimisation with or without simultaneous controller design. Therefore, this paper covers this subset of ER, which would help the understanding of the state of the art along with a brief background of the applied theory.

2. Algorithms for morphology evolution

EAs have a near standard structure and therefore the paper first elaborates the basic principle involved and subsequently explains how different researchers have used EAs in their works.

2.1. The basic structure of evolutionary algorithms:

As the name suggests, the algorithms are inspired from biological evolution. Consequently, each of the steps involved can be closely mapped with the different stages in biological evolution as nature searches for better living beings. The entire topic is explained in detail in [33].

The structure of a typical EA is as follows:

1. Initialise a population of candidate solutions.
2. Evaluate each candidate.
3. Select parent/s.
4. Recombine parents.
5. Mutate offspring.
6. Replace generation.

The steps 3 to 6 are repeated until the results are sufficiently close to the desired objective or after the computation time is exceeded.

2.1.1. Phenotype. Before discussing the above procedure, it is necessary to explain how the candidate solutions are represented. *Phenotype* refers to the physical or observable entity of the candidate solution, or robot, and the *phenotype space* is where the observable properties of a robot are found.

2.1.2. Genotype. In biology, the genome carries the various features of the organism through a set of genes and genotype is the equivalent term in ER. It encodes a description of the phenotype (or robot), and a genotype-phenotype mapping transfers the information between genotype space to phenotype space. Evaluation of each individual is performed on the phenotype while all other steps are conducted in the genotype space.

The genotype holds morphological and control parameters or control algorithms according to the application. Various morphological parameters convey details pertaining to body size, weight, wheel diameters, spatial information about position or coordinates or angle, or any other relevant information. The control side may include various neural network parameters or neural network wiring or other parameters unique to the specific controller type under use. In many of the applications the genome has a fixed length, with exceptions in [34, 35].

Genotype representations in the literature use binary [36-38], integer [39], real number [40-42] and string [43] encoding with many applying either or a combination of the different encoding methods. The basic genome types could also be placed in a matrix [44], vector [34, 45-47], graph [48-53] and tree [54-62]. Further, in other representations, [34, 63] applied the concept of a *hox* gene for encoding body morphology and [39, 64] applied Compositional Pattern-Producing Networks (CPPN) based encoding . The CPPN encoding involves accepting arguments as inputs and generating a resultant graph which explains the connections between various functions. L-system (Lindenmayer) representation in [65] is a type of grammar representation where developmental design rules or building commands are evolved to generate symmetrical structures.

2.1.3. Genotype-phenotype mapping. Each candidate's fitness is estimated through the performance of the corresponding phenotype using genotype to phenotype conversion. There are several mappings which can be separated into parametric and open-ended representations. Parametric representations appeared in fixed topology designs [36, 38, 47, 51, 60, 61, 66-69] where various parameters of the topology were evolved. Open-ended representations can be of generative and non-generative types. The generative encoding allows reuse of its individual elements during phenotype design. Each element is used only once in a non-generative encoding. Generative encoding is further divided into implicit and explicit, and non-generative encoding is either direct [40, 54, 59, 70] or indirect [64] types. See [71] for further details.

2.1.4. Population. The initial population acts as a seed for solutions which are later evolved into solution(s) with the desired performance. In the area under discussion, the population varied from 15 to 1000 candidate solutions when fitness evaluation was performed in computer simulation. But in a unique case, from 1 to 3 elements, actual robots were built for evaluation [72]. While most populations were initiated with random values, Samuelsen *et al.* [34] chose 200 identical individuals and the GOLEM project [73] had 200 to 1000 null individuals to start the evolutionary process. Single population evolved solutions except in [74, 75]. In [74], for faster convergence, when the average fitness of the first population reached a fixed value (or after 50 generations), the best individual was transferred to a second population. Once twenty such transfers were completed, the solutions were further optimised with the second population.

2.1.5. Fitness function. Each candidate solution in the population is tested using a fitness function. The phenotypes are active entities which are affected by the environment they are in. Each phenotype is placed in a simulated environment and the fitness function is designed for use in the phenotype space. Fitness functions were task specific through individual or combinations of penalty functions, cost functions, reward functions or based on the extent of objective attainment. The different kinds of fitness functions designed are listed in Table I.

Table I. Different types of fitness functions.

Fitness function based on	Citations
Movement (e.g. distance, area, speed)	[67, 70, 72, 76, 77]*, [34, 52, 78]* [36, 40, 79]*, [43, 51, 57, 59, 73, 80]*, [81]*, [74]*, [47]*, [82, 83]*, [37, 58]*, [69]*, [35]*, [53]*, [63]*, [84]*, [60, 61]*, [85]*, [86]*
Energy or cost	[40, 79]*, [46]*, [84]*, [38]
Action (e.g. collision, touch, fall, damage inflicted or suffered, obstacle avoidance)	[70]*, [67]*, [51]*, [60]*
Mechanical features (e.g. centre of gravity, torque generated, load, weight, size, dexterity, mobility, tension)	[64, 81]*, [74]*, [47]*, [83]*, [37]*, [35]*, [69]*, [46]*, [84]*, [85, 87]*
Design or feature (e.g. novelty, diversity)	[78]*, [55]
Goal specific (e.g. time to contact with prey, lifting, grasping, time on line, food eaten, competition winner, climbing ability)	[41, 45, 50, 53]*, [63]*, [46]*, [44, 86]*
Human feedback	[70]*, [66]

*Combination of more than one category.

2.1.6. Parent selection. The probability of creating a better offspring depends strongly on parent selection and hence, selection plays a major role in determining the time taken by EAs to converge to a satisfactory solution. Normally, parents with higher fitnesses are allotted a higher priority to be selected for reproduction. However, to avoid solutions being trapped in local optimum, parents with lower fitnesses are also selected, through a selection probability which is set at a low value [33]. In most cases, two parents are mated, except when applying asexual reproduction in [64]. There are several ways to select the parents:

Fitness proportional selection (FPS): In FPS, the individual's selection probability depends on its absolute fitness relative to the remaining population. Windowing or sigma scaling are commonly used to avoid premature convergence or to avoid solutions getting stuck in local minima due to the consistent selection of best candidates or to control selection pressure when fitness values are close to each other [33].

Ranking selection: FPS suffers from the lack of constant selection pressure. To avoid this, in *ranking selection*, the probability is allocated according to rank and not the fitness value itself with higher ranked individual being allocated higher selection probability. For selecting probabilities, commonly used techniques are *roulette wheel* and *stochastic universal sampling*. *Roulette wheel selection* conceptually involves randomly selecting parents by repeatedly spinning a hypothetical roulette wheel. Each slot on the wheel represents a selection probability (allocated from a *cumulative probablity distribution function*), and a parent is selected after every spin of the wheel. Among its disadvantages, a uniform selection of parents from the probability distribution is not possible as the wheel is spun more than once. To overcome this, *stochastic universal sampling* is applied where all parents are selected simultaneously from the distribution [33]. This is accomplished by using the concept similar to a *wheel of fortune,* where instead of one selecting pointer directing to a single slot, multiple uniformly spaced pointers (equal to number of parents) select multiple slots (with their size proportional to corresponding probability) in a single spin.

Tournament selection: The parent selection methods explained so far require the probability distribution of entire the population in every generation to apply the necessary algorithm. By contrast, tournament selection works by dividing the population into groups and selecting the best for each candidate's relative fitness in that group. The prominent parent selection methods applied in the area are listed in Table II.

2.1.7. Variation operations (Mutation and Recombination). Variation operators are applied to the selected parents to create new offspring. Usually, both recombination and mutation after mating are implemented in that order. When a child is generated from a

single parent, the process is called mutation, and when multiple parents are used, the process is referred to as recombination. Mutation involves stochastic modification of parent's genotype. It must be noted that genotype should be altered at random and should be unbiased. Therefore, slightly targeted tweaking of the genotype is not considered as mutation [33].

Table II. Methods of parent selection

Parent selection methods	Citations
Random	[52, 53, 67, 74]*
Tournament selection (binary/deterministic/triple/multiple population)	[34, 36, 50, 56, 60, 63, 68]*, [41, 88]*, [69]*, [58]*, [38]
Fitness proportionate (FPS/rank/cost/x best/neighbouring pair)	[39, 49, 74]*, [41, 55, 82, 88]*
Non-dominated sorting	[34]*, [81]*
Crowed distance sorting	[81]*
Diversity measure	[34]*
Stochastic universal sampling	[35]
Roulette wheel selection	[37, 46, 69]*, [58]*, [45]

*Combination of more than one category.

In another method, recombination or crossover operation is applied to the two parents to generate offspring. In this stochastic method, random parts of the mating parents are joined to inherit offspring with features from both parents. Moving away from the purely Darwinian model, theoretical studies and demonstrations have shown that using more than two parents in recombination produces more fit offspring. Despite this the method is seldom used [33]. There are several variations to the recombination and mutation operations when applied to solve problems in ER, and some of the significant variations are listed in Tables III and IV, respectively.

Table III. Mutation methods.

Mutation operations	Citations
Fixed non-adaptive	[67]
Gaussian	[41, 56, 57]
Gaussian Random	[44]
Standard	[36, 40, 45, 46, 51, 55, 61, 63, 64, 69, 79-81, 84, 87]
Through deletion, duplication, modification	[34, 53, 59, 73, 89], [48]
Variable probabilities	[82]
Concatenation	[58]
Binary/float/integer	[85, 86]
Vector differential mutation	[46]

Table IV. Types of crossover.

Crossover operations	Citations
Random	[39, 46, 89], [59], [61], [48, 85]
Single point	[36, 68], [83]
Two point	[38, 60, 76]
N point	[57]
Grafting, Copying	[50]
Classic	[40, 45, 51, 55, 69, 79, 81, 84, 87], [52, 86]
BL-$\propto$	[80]
Segregation	[86]

2.1.8. Survivor selection. The population size is fixed by finite resources. So the proportions of which of the offspring are added to the population and which individuals are replaced need to be decided. Age-based and fitness-based replacement are the two main types of

selection strategies. Fitness-based selection involves several methods, and the most prominent ones are discussed below.

A fixed number of least fit members are replaced from the population and rest are moved to the next generation in the *replace the worst* strategy. It results in a fast increase of average fitness, but this may also lead to premature convergence. As a result, it is only commonly applied to large populations. *Elitism* is implemented in addition to age-based and stochastic replacement methods to ensure that the fittest member of the population is retained and not replaced by other methods. In cases where fittest is among the population and not among the children, the latter is discarded. The round-robin tournament is another method in which, competitions are conducted between an offspring and a randomly selected opponent from a merged parent-child population. Each child competes for a fixed number of times, and once all of them finish competing, the offspring with highest wins replace the weakest of the population [33].

In the area under survey, 10% of the population were deleted by Lee [61], and that space was added with randomly generated candidates while [54] chose to retain the best children. In the single objective EA discussed in [67], the offspring immediately replaced parent if found to be fitter, and when multi-objective EA was used, offspring was compared with every individual in the population before replacement. Variable replacement strategy was used in [56] and *elitism* was excercised in [35, 38, 41, 46, 51, 52, 64, 81, 90]. Truncation strategy appeared in [34], and *elitism* with tournament selection was applied by Clark *et al.* [68] and Moore *et al.* [36]. Researchers [50] replaced the loser of the population with the child and in [74], individuals with costs higher than average of the population were replaced. Further, [44] retained random non-dominated solutions by replacing dominated solutions and parents were always replaced in [37]. The entire population was filled by children at every generation and the ten worst performers were replaced with randomly generated individuals every 50 generations in [82]. 20% of the highest scoring individuals in the population after competing against opposing population were retained, and rest were filled with offspring in [58]. Miniature round-robin tournaments were applied for selection and genotype validation, with garbage collection in [53]. Sims [48] and Shim *et al.* [69] retained 20% of the elite population, and rest were filled with offspring.

2.2. Different EAs applied

EAs have been realised using variations of the basic algorithm. While standard EAs were applied in [49, 67, 69, 72, 73, 76], an alternative approach is Multiple Objective Evolutionary Algorithms (MOEA) [40, 44, 67]. MOEA works with two fitness functions (eg. size fitness and performance fitness) that are calculated in parallel to find the dominant solutions (solutions with at least one fitness in one parent is better than the corresponding fitness in the other) which replace the non-dominant ones.

Genetic Algorithm (GA), Simple GA (SGA) or canonical GA uses a binary representation for its genotypes, FPS mechanism for parent selection, low mutation rates, and one point crossover is used as the standard recombination mechanism. The entire population is refreshed every generation as everyone is selected for crossover and is replaced by mutated children. However, depending on the probabilities set on variation operators, there can be copies of parents in the new population. SGAs possess flaws, to achieve faster convergence, they have been modified to include *elitism* and *tournament selection* [33]. SGAs were used for morphology development in [50, 56, 63, 66, 74, 83, 84] and they were amended with *elitism*, or other selection methods [35, 38, 43, 45, 46, 48, 51, 68, 87, 88]. A GA with novelty search and an objective based GA found application in [70] where individuals that performed differently were rewarded if they possessed novelty. A two level GA made of SGAs are in [37, 85, 90].

Another variation to the standard GA, is called the *Last Elite Opponent* (LEO) algorithm developed by Cliff *et al.* [75] which was inspired from Sims's [91] algorithm. It used two populations of solutions. The fitness of each individual in the first population was calculated by competing it with the champion of the other population. The best 20% were then retained and remaining were filled with children of these parents. Parents were selected via *roulette-wheel selection*, and the same operation was applied to the second population [58].

Smith *et al.* combined GA, extremal optimisation and pareto optimisation for evolution [74]. A GA generated two populations and best of the first population was moved into second population every 50 generations or after attaining a specified fitness. Later when the

population size reached 20, an extremal optimisation technique was applied. During this operation, the overall cost of each genotype was improved by modifying a specific gene which had the maximum adverse effect on the phenotype. A pareto optimisation technique then differentiated the best among competing designs. In this step, the cost of each gene in the genotype was compared with every other corresponding gene in other genotypes. The value of genes with greater costs was increased, and best genotype in the population was finally selected.

Non-dominated Sorting Genetic Algorithm II (NSGA-II) is a popular variation of GA developed by Deb *et al.* [92]. In NSGA-II, the parents are chosen from a ranked list of candidate solutions. After mutation and recombination operations, the child and parent populations are combined and ranked before applying elitist selection criteria for building the new population. The algorithm was implemented by Samuelsen *et al.* [34] and Rubrecht *et al.* [81].

In Differential Evolution (DE) [46], individual solutions are ranked through a cost function. During mutation, a vector differentiation method is applied by adding a third solution vector from the weighted difference of two others. Normally, parameter vector dimensions are equal to the number of design variables and population size is same as the number of parameter vectors.

A Multi-Chromosome Evolutionary Algorithm (MEA) proposed by Chocron [85] concentrated some of the robot's features on a single chromosome of floating point numbers. The variation operations were performed on each of them and not globally on the genotype. The algorithm was inspired from evolutionary strategy principles developed by Back [93]. The paper also tested an Adaptive Multi-Chromosome Evolutionary Algorithm (AMEA) with variation operator parameters modified as per an adaptive selection pressure function. A hard selection pressure was applied when fitness of solutions exhibited higher spread or greater standard deviation.

While standard EAs search for solutions, the more recent, Genetic Programming (GP) works by searching for a method or steps or an algorithm to build a solution. Chromosomes are represented as parse trees, and either recombination or mutation is applied instead of both [33]. GP can be found during morphology design in [55, 57, 79].

NeuroEvolution of Augmenting Topologies (NEAT) developed by Stanely *et al.* [94] found its application in [64] for developing Ribosomal robots. The algorithm starts on a simple CPPN encoded network and builds it complexity by the addition of nodes to the genotype. The genotype consists of node genes and corresponding connection genes. A structural mutation is carried out through the addition of a connection or a node. The connection weights are also allowed to mutate freely, and thus producing complex systems as the generations progress.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a variation of Evolution Strategy (ES). In a standard ES, a population with μ candidate solutions generate λ offspring. A random number is added to a randomly selected parent where the number is chosen from a Gaussian distribution with mean as zero and standard deviation or mutation step size called σ. In $(\mu+\lambda)$ ES, the offspring replaces parent if found to be fitter. On the other hand in (μ,λ) ES, the child always replaces the parent [33]. CMA-ES applied in [47, 77] is a leading form of ES utilised especially for high-dimensional and non-linear optimisation problems. A covariance matrix represents the pairwise joint variability of parameters in the Gaussian distribution which is updated by CMA. The technique self-adapts parameters in two stages. In the first step, the mean of the distribution and covariance matrix is incrementally updated to increase the probability of successful solutions from previous generations. In the second step, evolution paths (two paths from time evolution of mean) which hold useful information of nearby steps are used. One helps to avoid premature convergence while other supports the increase in the rate of variance. For a detailed explanation, see the tutorial by Hansen [95].

Java Evolutionary Algorithm Framework (JEAF) developed by Caamaño *et al.* [96] was applied in [54] to incorporate multiple algorithms like DE, GA and CMA-ES. The framework was built to allow the use of multiple algorithms without worrying about the background coding of each of the algorithms.

2.3. Simulators and the reality gap

During fitness evaluation, the individually designed fitness function is applied on the behaviour of candidates normally in a simulated environment. The task specific simulation

environments for mobile creatures where flat surfaces, except in a few cases when they were curved [47], uneven or stepped [97]. None of the reported works attempted to use a dynamically changing environment. The only partial dynamic aspect of a simulation was an automatic removal of consumed food [50]. Other than in [55, 79], a 3-D arena was always necessary for simulation. In a unique approach to building simulation platforms, [63] and [98] modelled them by converting the behaviour of a robot in reality to a virtual environment. The only online driven evaluation process was reported in [72]. The software packages developed were individually built from scratch or on easily available packages such as *MATLAB* or *PhysX* or *Open Dynamics Engine (ODE)* (Table V). Another notable common feature was the sending of each individual robot to another CAD module in the software package for evaluation. To speed up the performance of the simulator, multiple design checks were also performed before testing. For instance in [56], each robot was checked for the number and type of links, actuators and sensors before simulation. Likewise, Chocron [86] applied three hierarchical levels namely; mathematical elimination, quick simulation and full simulation.

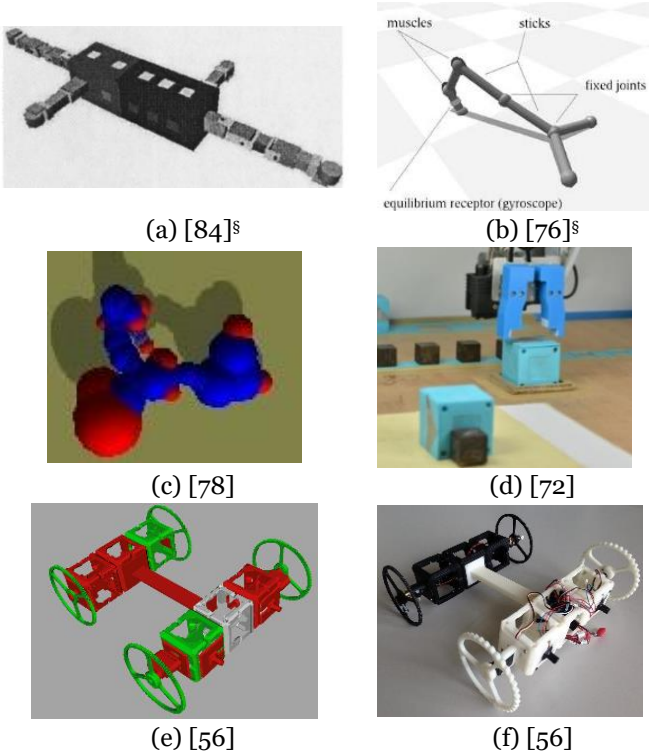
Quality of the simulation setup plays an important role in deciding if the evolved individual can perform the same task in reality as in a simulation. This difference is referred to as the *reality gap* in ER [99] and it is a widely-researched topic. Even though reality gap is a well-known concern in ER, only limited papers reported here shed light in the problems faced during physically building robots. Moore *et al.* [36] experienced difficulties during the physical validation of robots owing to poor modelling of mechanical elements such as servo motor joints and an inability to simulate physical conditions. The difference in on-board controller timing and simulator timing resulted in a drastic difference of robot speeds (55 cm per minute and 14 cm per minute, respectively) in virtual and real systems [59]. The inconsistency of 3D printing added towards a notable reality gap in [68]. Further, *numerical explosion* due to accuracy of simulator environment resulted in evolving solutions with unreasonable fitness [52].

To counter the reality gap and to accommodate the uncertainty of physical systems, a common method was to induce random noise to the measurements [56, 59]. For testing controller robustness, Bongard [44] performed damage testing via disabling sensors on the robot. Lee *et al.* [60] proposed using a training set with multiple starting positions and incorporating cumulative fitness to ensure robustness.

3. Algorithms in controller

During co-evolution, the body and controller are evolved simultaneously with EAs as the generations progress. The controller type selected for such evolutions are either artificial intelligence based (mainly Artificial Neural Networks (ANNs)) or traditional control techniques based.

In an ANN based control scheme, artificial neurons are internally wired to connect sensors (receptors), actuators (effectors) or other neurons. Output signals are generated based on the input value, corresponding internal weights, biases and other operations that undergo in the neural network. Various arithmetic operations or oscillating signals act at different neural nodes to manipulate the input signal. As the genotype representation carries information about the control system info of each part, it gets carried to offspring and gets modified during the variation operations. Examples of such or similar systems are included in [50-53, 56, 58, 73, 76, 88, 89, 91, 97]. Endo *et al.* [79] built a tree type of neural network with a maximum depth of five. For decision making, six different types of neural networks namely; AA, DD, AD, ND, NA, DA where A was for Analog, D was for Digital were incorporated. In an AA type network, input and output were analogue, in DD, both input and output were digital and in the other two, output was either analogue or digital but with no input signal. In a different approach, Shim *et al.* fed the evolved body parameters of a flying robot to a neural network to create a controller [69].



(a) [84]§ (b) [76]§

(c) [78] (d) [72]

(e) [56] (f) [56]

Fig.1 Examples of the evolved designs.

§ Reprinted courtesy of Springer.

Table V. Evaluation platforms.

Platform	Citations
Webots	[40, 67]
MARS	[47]
Ella	[57]
ODE	[36, 52, 53, 59, 64, 68, 69, 77, 82]
CimStation	[45]
Gazebo	[54]
MATLAB	[46, 70, 72]
PhysX	[34, 49]
GOLEM	[73, 89]
ORCOS	[81]
Custom six modules	[35]
MathEngine	[9]
LeGen[a]	[83]
Breve[a]	[97]
RoboGen[a]	[56]
YAKS	[41]
MASS[b]	[50]
EDHMoR	[54]
FEM based	[46, 76]
Custom C++ based	[86]

[a]Based on ODE
[b]Based on PhysX

Mazzapioda *et al.* [82] designed a neural controller with five neurons per joint with one for angular motor position and the rest for interaction with nearby joints. Here, each neuron generated four different signals and a sinusoidal oscillator controlled the motor angle based on the normalised signal values. A neural oscillator with PD (Proportional-Derivative) control and a neural network with a Central Pattern Generator (CPG) for control appeared in [90] and [34], respectively. The latter had six neurons with a total of 14 parameters that were interconnected in a specific pattern in every module. Similarly, the parameters of CPGs were evolved again by Larpin *et al.* [40]. A GP was used in [63] and [57] to evolve a perceptron (single layer neural network) based controller. Another variation of ANN called Continuous Time Recurrent Neural Networks (CTRNNs) could be seen in [44, 59, 78] and Elman's recurrent network in [83]. Lessin *et al.* [49] developed ESP (*encapsulation, syllabus and pandemonium)* principles to generate neural networks.

Instead of neurons, Mautner *et al.* [43] simulated artificial cells in the neural network. Multiple cell division operations resulted in adding connections to new sensors, and the vector sum at the output controlled the direction and power propagation in the network. A variation of the NEAT algorithm called Hyper-NEAT applied primarily in large scale ANN design was also used in [56] along with a standard ANN based controller. In other methods of controller design, Lee *et al.* [60, 61] based GP for the neural network evolution while GA for morphology evolution. There were instances when blank neural networks evolved into a useful controller [100] and combination of ANN and Finite State Machine (FSM) based controllers were employed [83].

Parker *et al.* [98] designed a reactive controller where a GA selected the rules for the sensor system to act. Simple inverse kinematic control evolved by an EA [47], Constraint Compliant Control (CCC) law which relied on velocity kinematic principles [81] and a simple periodic open loop control [70, 86] are examples for non-ANN based controllers evolved by EAs.

4. Application Areas

The areas discussed can be classified into works that purely deal with the evolution of mechanical design and works where control and morphology are co-evolved with the help of EAs. A comprehensive chronological list of works that report morphology only evolution, and co-evolution of morphology and control scheme can be found in Table VI and Table VII, respectively.

Table VI. List of works with morphology only evolution.

Author/s	Robot application type	Algorithm	Year
Chedmail *et al.* [45]	Manipulator robot design	GA	1996
Chung *et al.* [87]	Manipulator robot design	GA	1996
Chocron *et al.* [37]	Manipulator robot design	GA	1997
Farritor *et al.* [84]	Inspection robot design for constrained areas	GA	2001
Shiakolas *et al.* [46]	Manipulator robot design	GA, DE	2002
Parker *et al.* [35]	LEGO robot for locomotion	GA	2007
Lipson [55]	2D robot mechanism	GP	2008
Smith *et al.* [74]	Legged robot design	GA with Extremal & Pareto Optimisation	2010
Clark *et al.* [68]	Robot fin design	GA	2012
Lim *et al.* [67]	Six-legged robot design	SOEA, MOEA	2015
De Beir *et al.* [66]	Social robot design	GA	2016
Cruz *et al.* [38]	Mechanical design of humanoid	GA	2016

4.1. Morphology only evolution

4.1.1. Articulated robots. Many groups have developed manipulators using EAs starting with Chedmail *et al.* [45]. They designed serial and parallel manipulator robots with fixed end effector trajectory for avoiding obstacles. Similarly, Chung *et al.* [87] fixed degrees of freedom (DOF) of the robot to follow a specified trajectory and evolved its design. The Denavit-Hartenberg (DH) parameters of Selective Compliance Articulated Robot Arm (SCARA) and articulated robots with predefined initial and final position, motion time and joint constraints were evolved in [46]. In a different approach to serial manipulator design, Chocron *et al.* [37] applied a two level GA for a task specific robot, with upper layer for topology evolution and the lower level for finding the inverse kinematic solutions. Two-dimensional kinematic mechanisms which could draw a straight line were developed by Lipson [55] while keeping DOF fixed during evolution. The study also applied compensation operators for deleting redundant links from the evolved designs.

4.1.2. Mobile robots. Inspection robots for space constrained areas (like duct) were designed with a set of modules for power, control, joint and foot by Farritor *et al.* [84] (Fig. 1(a)). Even though this can be characterised under modular robots, it is included here as it has a single central controller as opposed to multiple controllers in a typical modular reconfigurable robot system. *LEGO* robot designs were searched by a GA primarily for maximising distance travelled with wheels in [35]. The GA required over 300 generations to evolve structures that stayed clear from the ground except by the robot's wheels. The fin length and *Young's Modulus* of a robotic fish were evolved in simulation using *ODE* by Clark *et al.* [68].

In legged robots, Smith *et al.* evolved various physical parameters of a legged robot, mainly body and leg dimensions including the number of legs, with the aim of increasing stability and mobility [74]. A six-wheeled articulated robot body parameters were optimised for step climbing operation by Lim *et al.* [67]. The kinematic parameters of a passive humanoid robot were evolved in [38] with a 32-member population which evolved over 500 generations in 9 hours on a standard personal computer.

Table VII. List of co-evolving creatures or robots.

Author/s	Application	Type of algorithm		Year
		Evolution	Controller	
Sims [48]	Creatures for swimming, walking and jumping	GA	ANN	1994
Lee *et al.* [60]	Mobile robot with obstacle avoidance	GA, GP	ANN	1996
Komosiński *et al.* [76]	Creatures for walking and swimming	EA	ANN	1999
Mautner *et al.* [43]	Mobile robot with obstacle avoidance	GA	ANN	2000
Pollack *et al.* [73]	Locomotion with linear elements	EA	ANN	2000
Lee [61]	Mobile robot with obstacle avoidance	GA, GP	ANN	2000
Endo *et al.* [79]	Multi-linked 2D robot for locomotion and hill climbing	GP	ANN	2001
Taylor *et al.* [9]	Creatures for swimming, walking and jumping	GA	ANN	2001
Lund [63]	Line follower robot with *LEGO*	GA, GP	ANN	2003
Pollack *et al.* [89]	*Genobots* for locomotion	EA	ANN	2003
Lee [101]	Straight locomotion with obstacle avoidance	GA, GP	ANN	2003
Endo *et al.* [90]	Humanoid body design	GA	ANN	2003
Shim *et al.* [69]	Wing structure design	EA	ANN	2004
O'Kelly *et al.* [53]	Creatures for combat	GA	ANN	2004
Macinnes *et al.* [59]	Locomotion with LEGO bricks	EA	CTRNN	2004
Miconi *et al.* [88]	Creatures with multiple locomotion modes	GA	ANN	2005
Lassabe *et al.* [97]	Creatures for multi-surface locomotion	GA	ANN	2007
Chaumont *et al.* [52]	Creatures for walking/block throwing	GA	ANN	2007
Parker *et al.* [98]	Sensor position and gait design	GA	Reactive controller	2007
Chocron [85]	Serial manipulator design	TGA, M/AMEA	Inverse kinematic	2007
Chocron [86]	Mobile robot for rough terrains	EA	Open-loop	2007
Miconi *et al.* [58]	Fighting creatures	GA with LEO	ANN	2008
Heinen *et al.* [83]	Four-legged walking robot	GA	FSM & ANN	2009
Mazzapoida *et al.* [82]	Creatures for irregular surface locomotion	EA	ANN	2009
Rommerman *et al.* [47]	Six-legged robot design	CMA-ES	Inverse kinematic controller	2009
Bongard [44]	Articulated robot design	MOEA	CTRNN	2010
Rubrecht *et al.* [81]	Serial manipulator design	NSGA-II	CCC	2011
Azarbadegan *et al.* [51]	Biped walking creatures	GA	ANN	2011
Gregor *et al.* [57]	Creatures for locomotion	GP	ANN	2012
Larpin *et al.* [40]	Quadrupedal robot design	MOEA	CPG	2012
Moore *et al.* [102, 36]	Amphibious robot design	EA	Sinusoidal controller	2012
Pilat *et al.* [50]	Food consuming creatures	Steady State GA	Recurrent ANN	2012
Auerbach *et al.* [78]	Creatures for locomotion	CPPN-NEAT	CTRNN	2013
Samuelsen *et al.* [34]	Robots for locomotion	NSGA-II	CPG	2013
Risi *et al.* [64]	Robots for walking	NEAT	CPPN	2013
Lessin *et al.* [49]	Light following/fighting creatures	EA	ESP-ANN	2014
Digumarti *et al.* [77]	Legged robot design	CMA-ES	Inverted pendulum based	2014
Auerbach *et al.* [56]	Racing/chasing robot design	GA	Recurrent ANN/Hyper-NEAT	2014
Corucci *et al.* [70]	Underwater robot design	GA	Open-loop control	2015
Faiña *et al.* [54]	Mobile robot design	Multiple EAs	Sinusoidal control	2015
Brodbeck *et al.* [72]	Mobile robot design	EA	Amplitude & phase shift based	2015

Evolved robot morphology for creatures that moved towards a goal with an innovative *hox* gene inspired historical marking strategy for tracking ancestors was shown applied by Samuelsen *et al.* [34]. In a unique application of EAs in robot design, the appearance of a Social Robot was evolved with human feedback acting as the fitness function [66].

4.2. Co-evolution of morphology and control

In this section, the research reported can be primarily classified into the evolution of virtual creatures and evolution of virtual but physically realisable robots.

4.2.1. Virtual Creatures or Robots. The first reported application of EAs in morphology evolution was for configuring sensor position on a mobile robot in 1993 [103]. However, as this paper deals with the evolution of stationary sensory element positions and not the core mechanical aspects of the robots, such works are not covered. The pivotal paper that reported the evolution of virtual creatures was by Sims in 1994 [48, 91] where body and brain of creatures for swimming, walking and jumping were evolved. Here, 3D blocks with imaginary joints were allowed to freely develop in simulation. Subsequent works were inspired by Sim's virtual creatures.

For instance, the work was later replicated by Taylor *et al.* [9] in the *MathEngine* platform. Again, virtual robot combat pairs with spherical links and motors were evolved in a period of few days in [53]. Further, a simple steady state GA was reported in [88] for imitating Sims's work. In the same direction, creatures for walking, climbing and skating on multiple surface types were evolved by Lassabe *et al.* [97]. Likewise, walking or block throwing Sims's creatures were developed on *ODE* by Chaumont *et al.* [52]. In a related work, Miconi [58] used Sims's creatures for fighting through the LEO algorithm. In [82], to evolve creatures that could transverse flat and irregular surfaces, a tub with round ends acted as the building block with possibilities of attaching to others or modifying its length. Yet again following Sims's work, robots with biped morphology were designed in [51].

A cuboidal body with hinge joints were the building blocks for evolving creatures aimed at consuming virtual food in [50]. The evolution of a creature called *Framestick* with sticks, muscles and sensors for orientation, smell and touch for walking and swimming is explored in [76] (Fig. 1(b)). A *Braitenberg* type vehicle design with the aim of reaching a goal while avoiding obstacles is described by Mautner *et al.* [43]. A four-legged virtual dog-like creature had its body dimensions evolve in the *LeGen* simulator [83].

Multi-linked robots were designed for 2D locomotion and hill climbing by Endo *et al.* [79]. They constrained the evolution process with two to seven links for the robot. Later, Bongard developed the body plan design of an articulated robot for grasping and lifting various objects [44]. Context blocks along with GP were used for evolution of body plan only and not the structure of creatures with the help of *Ella* software package by Gregor *et al.* [57]. Sphere shaped parts with embedded sensors for distance moved, touch, proprioception and time were evolved for covering maximum displacement in [78] (Fig. 1(c)). Risi *et al.* [64] evolved robots that moved straight made with 3D printed ribbons and embedded motors. On *NVIDIA PhysX*, creatures with muscle (spring), actuators and photoreceptors were evolved which can adapt its morphology as per the chosen task [49].

4.2.2. Physically realisable robots. The papers in the previous section covered the creatures or robots with virtually co-evolved body and controller, and they all suffered from a major drawback of not being able to be built physically. This section and Table VIII are dedicated to those works that report buildable robots.

Lee *et al.* [60], kept a fixed robot shape while evolving the structural parameters of a mobile robot for locomotion with obstacle avoidance. In a similar work, Lee [61] evolved parameters for obstacle avoidance on a three-wheeled robot. A robot to move in a straight line while avoiding obstacles was designed in [101] with its body parameters and controller evolving simultaneously.

The morphology and walking pattern of a six-legged robot for space missions were simultaneously optimised by Rommerman *et al.* [47] in the MARS (Machina Arte Robotum Simulans) simulation platform. The robot was simulated in three scenarios during its evaluation. A quadrupedal robot body parameters were evolved for maximum distance travelled using minimum energy in [40]. Another quadrupedal evolved using modular components for rough terrain exploration is presented by Chocron [86]. Using an innovative generative encoding, [65, 89, 104] created *Genobots* with bars and joints (actuated and non-

actuated) for locomotion. The GOLEM project at the Brandeis University evolved robots with linear bars and linear actuators and with the help of rapid prototyping tested the evolved prototypes [73].

Table VIII. Buildable co-evolved robots.

Author/s	Fixed design	Fixed parts library	Fixed part shape with variable size	Locomotion (Wheeled(W)/Legged(L))	Sensor feedback used during evolution	Controller parameter only evolution	Subtasks
Lee *et al.* [60]	✓			W	✓		✓
Pollack *et al.* [73]		✓	✓	L			
Lee [61]	✓			W	✓		✓
Lund [63]	✓			W	✓		✓
Pollack *et al.* [89]		✓	✓	L			
Lee [101]	✓			W	✓		✓
Chocron [86]		✓		W&/L		✓	✓
Endo *et al.* [90]	✓			L	✓	✓	
Shim *et al.* [69]		✓	✓	Winged	✓	✓	
Macinnes *et al.* [59]		✓		L	✓		
Rommerman *et al.* [47]	✓			L	✓	✓	
Rubrecht *et al.* [81]		✓	✓	^	✓		
Larpin *et al.* [40]	✓			L		✓	
Moore *et al.* [102, 36]	✓			L[$]		✓	
Samuelsen *et al.* [34]		✓	✓	L		✓	
Digumarti *et al.* [77]	✓			L		✓	
Auerbach *et al.* [56]		✓		W&/L	✓		✓
Corucci *et al.* [70]	✓			L[$]		✓	
Faiña *et al.* [54]		✓		L		✓	✓
Brodbeck *et al.* [72]		✓		L	✓	✓	

[$] Underwater fin based.
^ Not a mobile robot.

A line follower was evolved by Lund [63] using *LEGO Mindstorms* where the morphological shape was fixed and allowed the evolution of all other parameters. From a library of pre-existing parts comprising of *LEGO* bricks, Macinnes *et al.* [59] evolved robots for locomotion. Servo motors and position feedback sensors used in the assembly of the final robot after evolution.

A number of variables of a fixed topology biped humanoid robot along with it controller for gait design were evolved in [90]. *ODE* found its application in developing the body dimensions and control parameters of a four-legged robot for optimal speed and predefined gait [77]. The wing structure and low-level controllers of a robot for flying along a straight line were evolved by Shim *et al.* [69]. The frequently used *ODE* platform appeared again in [36, 102] for evolving body parameters of an amphibious robot and the final model there was 3D printed to confirm the simulation.

Along similar lines, existing design parameters of an underwater robot *PoseiDRONE* was modified using a novelty search based GA in *MATLAB* by Corucci *et al.* [70]. Chocron [85] applied two level GA similar to [37] for evolving the end effector pose and orientation with other physical parameters of a serial manipulator with multiple chromosome genotypes. A serial manipulator for highly constrained space like the inside of a tunnel boring machine is elaborated in [81]. There, each robot was made of segments which comprised of links with revolute or prismatic joints for a maximum of one-DOF.

RoboGen [56] is an open-source platform which generates robots for racing or chasing activities with a set of standard parts including servo motor actuators, IR, light and IMU sensors and *Arduino* controller. The package is capable of evolving morphology and controller and generates 3D printable models which can be physically assembled. A virtual and corresponding 3D printed evolved mobile robot is shown in Figures 1(e) and (f), respectively.

Faiña *et al.* [54] developed a system for applications involving a list of subtasks such as painting, carrying and cleaning executed through a variety of locomotion modes like climbing, walking, rolling and crawling. The system was called *EDHMoR* [105] with the core module containing encoder and accelerometer and there were four possible options for actuators or special sensors that could be connected on empty slots of the core module.

In all the works above, the evolution was performed in a software platform, and best designs were implemented in reality. However, in the paper by Brobeck *et al.* [72], a 6-DOF serial mother robot built the solution population and tested their speed of locomotion and created better solutions with the help of an EA. The solution population was constructed from active and passive modules. Servo motor, *Bluetooth* module and *Arduino* controller were used in the module while the evolution process was offloaded to *MATLAB* running on a PC which was interfaced with the mother robot. The end-effector of mother robot building robots is shown in Fig. 1(d). In a step further, Weel *et al.* [39] explained a futuristic concept of online co-evolution as a proof of concept with no central evolution process involved.

5. Discussion and Conclusion

In this paper, the general principle of EAs along with the various methods applied for evolving body morphology and controller have been explored. A brief explanation of algorithms for controller development and how they are applied in different scenarios are also covered in the above sections. In the quarter century that evolutionary computing techniques have found applications in robotics, more than 80% of the works reported are in the post-2000 era which could be owing to the advent of fast computing systems as optimisation methods are computationally expensive. This could also be why in most of the applications, the core topology of robots was fixed.

While many researchers applied EAs in robotics as a proof of concept, there were only a handful who tested the evolved robots in reality. Usually, when the design was allowed to evolve freely, the ability to physical realise it appeared to be compromised. When the aim was to build physical robots, the evolution process was confined to the selection of parts from a predefined set, except in a few cases. However, in such situations, any sensory feedback in the system seemed to be missing (Table VIII).

In buildable systems, the general trend of application area appeared to be around locomotion and serial manipulation. The cause for such constrained applications might be due to several reasons as follows: The long time required for evolution on the current general-purpose computing systems, even for slightly complex tasks. The reality gap between the simulation and real systems. Difficulty in designing effective fitness functions, and methodological problems such as biasing and premature convergence.

The reality gap could also be contributing to discouraging physical realisation. Even though, there has been several reported works addressing the reality gap in ER, they mainly focus on controller only evolution, this suggests the need for further research specific to co-evolution process. A similar trend is also observed in designing fitness functions where functions for works related to controller only evolution show multiple types being applied, such as aggregate, competitive, environmental, behavioural, incremental, tailored and training data based fitness functions [17]. It has also been demonstrated that EAs could help in the latest thrust areas of online lifelong learning where neural networks evolve though robot's lifetime, perform diagnostic/repair functions automatically and even perform co-operative actions.

The review also suggests the need for software packages integrating multiple areas covering physics simulator, CAD modelling, controller development and evolver as only an interdisciplinary team is restricted to using the full advantage of ER. This highlights an immediate need for developing a stable high-level ER environment. There is also a need to consolidate the generated knowledge and develop standards including benchmarking methods to aid transferability and to move away from *ad hoc* practices. Further, formal conventional methods for morphology and controller designs guarantee convergence and discovery of suitable solutions. All these reasons collectively could be why ER is not given sufficient weight in robotics or discourage roboticists to delve into the area. However, it must not be forgotten that humans are a result of evolution by natural selection and we are yet to replicate the grace and sophistication exhibited by its creations from the traditional approaches.

On a final note, evolutionary approaches have found successful application in robot

morphology design and simultaneous evolution of controller and structure among other application areas. It is reckoned that the technology has still not reached its peak and will continue to evolve towards fully *automatic synthesis* [4] of robots in the future or towards the *evolution of things* [32]. Even though ER is inspired by biological evolution which has evolved over trillions of organisms, we may still be far away from conducting evolution at that scale. Nevertheless, this should gradually change as our understanding of the biological evolution deepens on one side and the technology matures on the other.

References

1. C. Darwin, *The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life* (John Murray. London, 1859).
2. G. Mendel, "Experiments in plant hybridization (1865)," **In:** Classic papers in genetics, (1), 1–19, (Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1959).
3. J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence* (MIT Press. Cambridge, MA, USA, 1992).
4. S. Doncieux, J.B. Mouret, N. Bredeche, and V. Padois, "Evolutionary Robotics: Exploring New Horizons," **In:** *New Horizons in Evolutionary Robotics*, **341**(1), 3–25, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2011).
5. S. Nolfi and D. Floreano, *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines* (MIT Press. Cambridge, MA, USA, 2000).
6. S. Nichele, "The coevolution of robot controllers (" brains") and morphologies (" bodies")–challenges and opportunities," online, 2015. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.2391&rep=rep1&type=pdf. Accessed Mar- 20, 2017.
7. P. A. Vargas, E. A. Di Paolo, I. Harvey, and P. Husbands, *The horizons of evolutionary robotics* (MIT Press. Cambridge, MA, USA, 2014).
8. J. C. Bongard, "Evolutionary robotics," *Communications of the ACM* **56**(8), 74–83 (2013).
9. T. Taylor and C. Massey, "Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creatures," *Artif. Life* **7**(1), 77–87 (2001).
10. D. Floreano, F. Mondada, A. Perez-Uribe, and D. Roggen, "Evolution of Embodied Intelligence," **In:** *Embodied Artificial Intelligence*, **3139**(23), 293–311, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2004).
11. H. Lipson, "Evolutionary robotics and open-ended design automation," *Biomimetics* **17**(9), 139–155 (2005).
12. R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics.," *Science* **318**(5853), 1088–1093 (2007).
13. A. F. T. Winfield and J. Timmis, "Evolvable Robot Hardware," **In:** *Evolvable Hardware*, (13), 331–348, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2015).
14. S. Nolfi, J. Bongard, P. Husbands, and D. Floreano, "Evolutionary Robotics," **In:** *Springer Handbook of Robotics*, 2nd ed.**236**(76), 2035–2068, (Springer International Publishing. Cham, 2016).
15. A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things," *Nature* **521**(7553), 476–482 (2015).
16. S. Gupta and E. Singla, "Evolutionary robotics in two decades: A review," *Sadhana* **40**(4), 1169–1184 (2015).
17. A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Rob. Auton. Syst.* **57**(4), 345–370 (2009).
18. J. Walker, S. Garrett, and M. Wilson, "Evolving controllers for real robots: A survey of the literature," *Adapt. Behav.* **11**(3), 179–203 (2003).
19. J. Teo, "Darwin+ robots= evolutionary robotics: Challenges in automatic robot synthesis," **In:** *Proceedings of the International Conference on Artificial Life* (2004) pp. 7–13.
20. R. Kicinger, T. Arciszewski, and K. De Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art," *Comput. Struct.* **83**(23), 1943–1978 (2005).
21. J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica* **31**(3), 345–359 (2013).
22. H. Ahmadzadeh, E. Masehian, and M. Asadpour, "Modular Robotic Systems: Characteristics and Applications," *J. Intell. Rob. Syst.* **81**(3), 317–357 (2016).
23. R. Alattas, "Evolutionary Modular Self-Assembly and Self-Reconfigurable Robotics: Exhaustive Review," *arXiv.org*. 09-Feb-2017.
24. C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. Jin Kim, and S. Kim, "Soft robot review," *Int. J. Control Autom. Syst.* **15**(1), 1–13 (2017).
25. R. Pfeifer, F. Iida, and M. Lungarella, "Cognition from the bottom up: on biological inspiration, body morphology, and soft materials," *Trends Cognit. Sci.* **18**(8), 404–413 (2014).
26. W. Aguilar, G. S. a-Bonfil, T. Froese, and C. Gershenson, "The Past, Present, and Future of

Artificial Life," *Front. Robot. AI* **1**(3), 4463 (2014).
27. R. Doursat, H. Sayama, and O. Michel, "A review of morphogenetic engineering," *Nat. Comput.* **12**(4), 517–535 (2013).
28. K. O. Stanley, "Why Evolutionary Robotics Will Matter," **In:** *New Horizons in Evolutionary Robotics*, **341**(3), 37–41, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2011).
29. J. A. Meyer, P. Husbands, and I. Harvey, "Evolutionary robotics: A survey of applications and problems," **In:** *Evolutionary Robotics*, **1468**(1), 1–21, (Springer Berlin Heidelberg. Berlin, Heidelberg, 1998).
30. F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and A. L. Christensen, "Open Issues in Evolutionary Robotics," *Evol. Comput.* **24**(2), 205–236 (2016).
31. J. Bongard, "Why Morphology Matters," **In:** *The horizons of evolutionary robotics*, (6), 125–152, (The MIT Press. Cambridge, MA, USA, 2014).
32. A. E. Eiben and J. E. Smith, "Towards the evolution of things," *SIGEVOlution* **8**(3), 3–6 (2016).
33. A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing* (Springer Berlin Heidelberg. Berlin, Heidelberg, 2015).
34. E. Samuelsen, K. Glette, and J. Torresen, "A hox gene inspired generative approach to evolving robot morphology," **In:** *Proceedings of the Genetic and Evolutionary Computation Conference* (2013) pp. 751–758.
35. G. B. Parker, D. Duzevik, A. S. Anev, and R. Georgescu, "Morphological Evolution of Dynamic Structures in a 3-Dimensional Simulated Environment," **In:** *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation* (2007) pp. 534–540.
36. J. M. Moore and P. K. McKinley, "Evolution of an amphibious robot with passive joints," **In:** *Proceedings of the IEEE Congress on Evolutionary Computation* (2013) pp. 1443–1450.
37. O. Chocron and P. Bidaud, "Genetic design of 3D modular manipulators," **In:** *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation* (1997) pp. 223–228.
38. R. S. Núñez Cruz and J. M. Ibarra Zannatha, "Efficient mechanical design and limit cycle stability for a humanoid robot: An application of genetic algorithms," *Neurocomputing* **233,** 72–80 (2017).
39. B. Weel, E. Crosato, J. Heinerman, E. Haasdijk, and A. E. Eiben, "A robotic ecosystem with evolvable minds and bodies," **In:** *Proceedings of the IEEE International Conference on Evolvable Systems* (2014) pp. 165–172.
40. K. Larpin, S. Pouya, J. van den Kieboom, and A. J. Ijspeert, "Co-evolution of morphology and control of virtual legged robots for a steering task," **In:** *Proceedings of the IEEE International Conference on Robotics and Biomimetics* (2011) pp. 2799–2804.
41. G. Buason, N. Bergfeldt, and T. Ziemke, "Brains, Bodies, and Beyond: Competitive Co-Evolution of Robot Controllers, Morphologies and Environments," *Genet. Program. Evolvable Mach.* **6**(1), 25–51 (2005).
42. K. Endo, T. Maeno, and H. Kitano, "Co-evolution of morphology and walking pattern of biped humanoid robot using evolutionary computation:designing the real robot," **In:** *Proceedings of the IEEE International Conference on Robotics and Automation* (2003) pp. 1362–1367.
43. C. Mautner and R. K. Belew, "Evolving robot morphology and control," *Artif. Life Rob.* **4**(3), 130–136 (2000).
44. J. Bongard, "The Utility of Evolving Simulated Robot Morphology Increases with Task Complexity for Object Manipulation," *Artif. Life* **16**(3), 201–223 (2010).
45. P. Chedmail and E. Ramstein, "Robot mechanism synthesis and genetic algorithms," **In:** *Proceedings of the IEEE International Conference on Robotics and Automation* (1996) pp. 3466–3471.
46. P. S. Shiakolas, D. Koladiya, and J. Kebrle, "Optimum Robot Design Based on Task Specifications Using Evolutionary Techniques and Kinematic, Dynamic, and Structural Constraints," **In:** *Proceedings of the International Mechanical Engineering Congress and Exposition* (2002) **2002** pp. 825–832.
47. M. Rommerman, D. Kuhn, and F. Kirchner, "Robot design for space missions using evolutionary computation," **In:** *Proceedings of the IEEE Congress on Evolutionary Computation* (2009) pp. 2098–2105.
48. K. Sims, "Evolving virtual creatures," **In:** *Proceedings of the Annual Conference on Computer Graphics* (1994) pp. 15–22.
49. D. Lessin, D. Fussell, and R. Miikkulainen, "Adopting Morphology to Multiple Tasks in Evolved Virtual Creatures," **In:** *Proceedings of the International Conference on Simulation and Synthesis of Living Systems* (2014) pp. 247–254.
50. M. L. Pilat, T. Ito, R. Suzuki, and T. Arita, "Evolution of virtual creature foraging in a physical environment," **In:** *Proceedings of the International Conference on Simulation and Synthesis of Living Systems* (2012) pp. 423–430.
51. A. Azarbadegan, F. Broz, and C. L. Nehaniv, "Evolving Sims's creatures for bipedal gait," **In:** *Proceedings of the IEEE Symposium On Artificial Life* (2011) pp. 218–224.
52. N. Chaumont, R. Egli, and C. Adami, "Evolving virtual creatures and catapults," *Artif. Life* **13**(2),

139–157 (2007).
53. M. O'kelly and K. Hsiao, "Evolving Simulated Mutually Perceptive Creatures for Combat," **In:** *Proceedings of the International Conference on Simulation and Synthesis of Living Systems* (2004) pp. 113–118.
54. A. Faiña, F. Bellas, F. Orjales, D. Souto, and R. J. Duro, "An evolution friendly modular architecture to produce feasible robots," *Rob. Auton. Syst.* **63,** 195–205 (2015).
55. H. Lipson, "Evolutionary synthesis of kinematic mechanisms," *Artif. Intell. Eng. Des. Anal. Manuf.* **22**(3), 195–205 (2008).
56. J. Auerbach, D. Aydin, A. Maesani, P. Kornatowski, T. Cieslewski, G. Heitz, P. Fernando, I. Loshchilov, L. Daler, and D. Floreano, "RoboGen: Robot Generation through Artificial Evolution," **In:** *Proceedings of the Artificial Life 14: International Conference on the Synthesis and Simulation of Living Systems* (2014) pp. 136–137.
57. M. Gregor, J. Spalek, and J. Capák, "Use of context blocks in genetic programming for evolution of robot morphology," **In:** *Proceedings of the International Conference ELEKTRO* (2012) pp. 286–291.
58. T. Miconi, "In Silicon No One Can Hear You Scream: Evolving Fighting Creatures," **In:** *Genetic Programming*, **4971**(3), 25–36, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2008).
59. I. Macinnes and E. Di Paolo, "Crawling out of the simulation: Evolving real robot morphologies using cheap reusable modules," **In:** *Proceedings of the International Conference on Simulation and Synthesis of Living Systems* (2004) pp. 94–99.
60. W. P. Lee, J. Hallam, and H. H. Lund, "A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks," **In:** *Proceedings of the IEEE International Conference on Evolutionary Computation* (1996) pp. 384–389.
61. W. P. Lee, "Evolving Autonomous Robot: From Controller to Morphology," *IEICE Trans. Inf. Syst.* **83**(2), 200–210 (2000).
62. D. Lessin, D. Fussell, and R. Miikkulainen, "Open-ended behavioral complexity for evolved virtual creatures," **In:** *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2013) pp. 335–342.
63. H. H. Lund, "Co-evolving Control and Morphology with LEGO Robots," **In:** *Morpho-functional Machines: The New Species*, (4), 59–79, (Springer Japan. Tokyo, 2003).
64. S. Risi, D. Cellucci, and H. Lipson, "Ribosomal robots: Evolved designs inspired by protein folding," **In:** *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2013) pp. 263–270.
65. G. S. Hornby and J. B. Pollack, "Body-brain co-evolution using L-systems as a generative encoding," **In:** *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2001) pp. 868–875.
66. A. De Beir and B. Vanderborght, "Evolutionary method for robot morphology: Case study of social robot probo," **In:** *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction* (2016) pp. 609–610.
67. S. H. Lim and J. Teo, "Design, Optimization and Fabrication of a Climbing Six Articulated-Wheeled Robot Using Artificial Evolution and 3D Printing," *BR. J. Math. Comput. Sci.* **10**(2), 1–21 (2015).
68. A. J. Clark, J. M. Moore, J. Wang, and X. Tan, "Evolutionary design and experimental validation of a flexible caudal fin for robotic fish," **In:** *Proceedings of the International Conference on Simulation and Synthesis of Living Systems* (2012) pp. 325–332.
69. Y. S. Shim, S. J. Kim, and C. H. Kim, "Evolving flying creatures with path-following behavior," **In:** *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation* (2004) pp. 125–132.
70. F. Corucci, M. Calisti, H. Hauser, and C. Laschi, "Novelty-Based Evolutionary Design of Morphing Underwater Robots," **In:** *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2015) pp. 145–152.
71. G. S. Hornby, *Generative representations for evolutionary design automation*. Ph.D. Thesis (Brandeis University. Waltham, Massachusetts, USA, 2003).
72. L. Brodbeck, S. Hauser, and F. Iida, "Morphological Evolution of Physical Robots through Model-Free Phenotype Development," *PLoS ONE* **10**(6), e0128444 (2015).
73. J. B. Pollack and H. Lipson, "The GOLEM project: Evolving hardware bodies and brains," **In:** *Proceedings of the NASA/DoD Workshop on Evolvable Hardware* (2000) pp. 37–42.
74. B. G. R. Smith, C. M. Saaj, and E. Allouis, "Evolving legged robots using biologically inspired optimization strategies," **In:** *Proceedings of the IEEE International Conference on Robotics and Biomimetics* (2010) pp. 1335–1340.
75. D. Cliff and G. F. Miller, "Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations," **In:** Advances in Artificial Life, **929**(16), 200–218, (Springer Berlin Heidelberg. Berlin, Heidelberg, 1995).
76. M. Komosiński and S. Ulatowski, "Framsticks: Towards a Simulation of a Nature-Like World, Creatures and Evolution," **In:** Applications of Evolutionary Computation, **1674**(33), 261–265, (Springer Berlin Heidelberg. Berlin, Heidelberg, 1999).

77. K. Digumarti, "Concurrent optimization of mechanical design and locomotion control of a legged robot," **In:** *Proceedings of the Mobile Service Robotics: Proceedings of the 17th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines* (2014) pp. 315–323.
78. J. E. Auerbach and J. C. Bongard, "Evolving complete robots with CPPN-NEAT: the utility of recurrent connections," **In:** *Proceedings of the Annual Conference on Genetic and Evolutionary Computation* (2011) pp. 1475–1482.
79. K. Endo and T. Maeno, "Simultaneous design of morphology of body, neural systems and adaptability to environment of multi-link-type locomotive robots using genetic programming," **In:** *Proceedings of the RSJ/IEEE International Conference on Intelligent Robots and Systems* (2001) pp. 2282–2287.
80. K. Endo, T. Maeno, and H. Kitano, "Co-evolution of morphology and walking pattern of biped humanoid robot using evolutionary computation. Consideration of characteristic of the servomotors," **In:** *Proceedings of the RSJ/IEEE International Conference on Intelligent Robots and Systems* (2002) pp. 2678–2683.
81. S. Rubrecht, E. Singla, V. Padois, P. Bidaud, and M. de Broissia, "Evolutionary Design of a Robotic Manipulator for a Highly Constrained Environment," **In:** *New Horizons in Evolutionary Robotics*, **341**(8), 109–121, (Springer Berlin Heidelberg. Berlin, Heidelberg, 2011).
82. M. Mazzapioda, A. Cangelosi, and S. Nolfi, "Evolving morphology and control: A distributed approach," **In:** *Proceedings of the IEEE Congress on Evolutionary Computation* (2009) pp. 2217–2224.
83. M. R. Heinen and F. S. Osório, "Evolving morphologies and gaits of physically realistic simulated robots," **In:** *Proceedings of the ACM symposium on Applied Computing* (2009) pp. 1161–1165.
84. S. Farritor and S. Dubowsky, "On Modular Design of Field Robotic Systems," *Auton. Robot.* **10**(1), 57–65 (2001).
85. O. Chocron, "Evolutionary design of modular robotic arms," *Robotica* **26**(3), 323–330 (2007).
86. O. Chocron, "Evolving Modular robots for rough terrain exploration", In: *Mobile Robots: The Evolutionary Approach*, 50(2), 23–46 (Springer-Verlag, Berlin, Heidelberg, 2007).
87. W. K. Chung, Jeongheon Han, Y. Youm, and S. H. Kim, "Task based design of modular robot manipulator using efficient genetic algorithm," **In:** *Proceedings of the International Conference on Robotics and Automation* (1997) pp. 507–512.
88. T. Miconi and A. Channon, "A virtual creatures model for studies in artificial evolution," **In:** *Proceedings of the IEEE Congress on Evolutionary Computation* (2005) pp. 565–572.
89. J. B. Pollack, G. S. Hornby, H. Lipson, and P. Funes, "Computer creativity in the automatic design of robots," *Leonardo* **36**(2), 115–121 (2003).
90. K. Endo, T. Maeno, and H. Kitano, "Co-evolution of morphology and walking pattern of biped humanoid robot using evolutionary computation - evolutionary designing method and its evaluation," **In:** *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (2003) pp. 340–345.
91. K. Sims, "Evolving 3D Morphology and Behavior by Competition," *Artif. Life* **1**(4), 353–372 (1994).
92. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Computat.* **6**(2), 182–197 (2002).
93. T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms* (Oxford University Press. New York, 1996).
94. K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evol. Comput.* **10**(2), 99–127 (2002).
95. N. Hansen, "The CMA Evolution Strategy: A Tutorial," *arXiv.org*, **1604**. arXiv:1604.00772, 2016.
96. P. Caamaño, R. Tedín, A. Paz-Lopez, and J. A. Becerra, "JEAF: A Java Evolutionary Algorithm Framework," **In:** *Proceedings of the IEEE Congress on Evolutionary Computation* (2010) pp. 1–8.
97. N. Lassabe, H. Luga, and Y. Duthen, "A new step for artificial creatures," **In:** *Proceedings of the IEEE Symposium on Artificial Life* (2007) pp. 243–250.
98. G. B. Parker and P. J. Nathan, "Co-evolution of sensor morphology and control on a simulated legged robot," **In:** *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation* (2007) pp. 516–521.
99. N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," **In:** Advances in Artificial Life, **929**(53), 704–720, (Springer, Berlin, Heidelberg. Berlin, Heidelberg, 1995).
100. A. L. Nelson and E. Grant "Aggregate selection in evolutionary robotics", In: *Mobile Robots: The Evolutionary Approach*, **50**(4), 63–88 (Springer-Verlag, Berlin, Heidelberg, 2007).
101. W. P. Lee, "Evolving robot brains and bodies together: An experimental investigation," *J. Chin. Inst. Eng.* **26**(2), 125–132 (2003).
102. J. M. Moore and P. K. McKinley, "Evolving flexible joint morphologies," **In:** *Proceedings of the*

Annual Conference on Genetic and Evolutionary Computation (2012) pp. 145–152.
103. D. Cliff, P. Husbands, and I. Harvey, "Explorations in Evolutionary Robotics," *Adapt. Behav.* **2**(1), 73–110 (1993).
104. H. Lipson and J. Pollack, "Evolving physical creatures," **In:** *Proceedings of the International Conference on Artificial Life* (2006) pp. 282–287.
105. A. Faíña, F. Bellas, F. López-Peña, and R. J. Duro, "EDHMoR: Evolutionary designer of heterogeneous modular robots," *Eng. Appl. Artif. Intell.* **26**(10), 2408–2423 (2013).