Evolutionary Robotics: Exploring New Horizons

Stéphane Doncieux, Jean-Baptiste Mouret, Nicolas Bredeche, Vincent Padois

Chapter 1
Evolutionary Robotics: Exploring New Horizons

Stéphane Doncieux, Jean-Baptiste Mouret, Nicolas Bredeche, and Vincent Padois

Abstract. This paper considers the field of Evolutionary Robotics (ER) from the perspective of its potential users: roboticists. The core hypothesis motivating this field of research is discussed, as well as the potential use of ER in a robot design process. Four main aspects of ER are presented: (a) ER as an automatic parameter tuning procedure, which is the most mature application and is used to solve real robotics problem, (b) evolutionary-aided design, which may benefit the designer as an efficient tool to build robotic systems (c) ER for online adaptation, i.e. continuous adaptation to changing environment or robot features and (d) automatic synthesis, which corresponds to the automatic design of a mechatronic device and its control system. Critical issues are also presented as well as current trends and pespectives in ER. A section is devoted to a roboticist's point of view and the last section discusses the current status of the field and makes some suggestions to increase its maturity.

1.1 Introduction

The advent of genetic algorithms in the sixties, as a computational abstraction of Darwin's theory of evolution, promised to transfer the richness and efficiency of living organisms to artificial agents, such as robotic systems. This envisioned future inspired a whole field of research, now called *Evolutionary Robotics* (ER) [28, 70, 80], in which researchers create evolutionary algorithms to design robots, or some part of robots such as their "artificial brain". The long-term goal of this field is to

Stéphane Doncieux · Jean-Baptiste Mouret · Vincent Padois
ISIR Pierre and Marie Curie University, CNRS Pyramide Tour 55 Boite courrier 173
4, place Jussieu 75252 Paris cedex 05 France
e-mail: stephane.doncieux@isir.upmc.fr,
jean-baptiste.mouret@isir.upmc.fr,
vincent.padois@isir.upmc.fr
Nicolas Bredeche
TAO - University Paris-Sud, INRIA, CNRS LRI, Bat. 490, 91405 Orsay, France
e-mail: nicolas.bredeche@lri.fr

S. Doncieux et al. (Eds.): New Horizons in Evolutionary Robotics, SCI 341, pp. 3–25.
springerlink.com

obtain an automatic process able to design, and even build, an optimal robot given only the specification of a task; the main underlying hypothesis is that Darwin's theory of evolution is the best source of inspiration, in particular because Nature demonstrated its efficiency; the main hope is to obtain machines that fully and robustly exploit the non-linear dynamics offered by their structure and their environment without having to model them explicitly.

After almost twenty years of ER research, simple crawling robots have been automatically designed then manufactured [64]; neural networks have been evolved to allow wheeled robot to avoid obstacles then autonomously charge their battery [29]; neural networks have also been evolved to drive walking [50, 55] and flying [78, 90] robots, as well as self-organizing swarm of robots [5, 38].

These results demonstrate that it is *possible* to automatically design robots or parts of robots with evolutionary algorithms. However, most evolved robots or controllers are not yet competitive with human-designed solutions. What was seen as complex challenges for robotics twenty years ago (walking robots with many degrees of freedom, non-linear control, simple but emergent reactive behaviors, ...) has now been widely investigated in robotics and many efficient solutions have been proposed.

Concurrently with the advances in robotics, evolutionary robotics matured too, both with regards to the basis of evolutionary computation and to its application, and it may be time to reconsider its place with regards to the robotics field. Consequent to this analysis, this paper tackles the simple question: how current evolutionary algorithms can be used in current robotics? After a short reminder of Evolutionary Algorithms (EA) (section (1.2)), we describe the conditions of EA applicability (section 1.3), i.e. when ER should be taken into consideration. We then review the main techniques developed in the ER field by dividing them into mature techniques (section 1.4.1), current trends (section 1.4.2 and 1.4.3) and long-term research (section 1.4.4). We discuss the current challenges of ER and the corresponding perspectives (section 1.5). The point of view of a roboticist is presented in section 1.6 and a discussion on ER as a scientific field together with suggestions to make it more mature end the paper.

1.2 A brief Introduction to Evolutionary Computation

Evolutionary Computation (EC) has been investigated for more than 40 years, with pioneering works both in Computer Sciences (Genetic Algorithms [48]) and Applied Mathematics (Evolution Strategies [85, 89]). Today, the term includes several sub-branches that share the common background of taking a more or less loose inspiration from Darwin's principles of natural selection and blind variations [17]. Moreover, the field has made great progress both considering fundamental concepts (e.g. with the advent of developmental representations [95]) as well as theoretical grounding, from Evolution Strategies [8] to symbolic regression in Genetic Programming [2]. Tools from Evolutionary Computation are now widely accepted within the engineer's meta-heuristic toolbox, and have been successfully applied to

several domains, such as automatic design of NASA satellite antennae [65][66][67], chairs [41, 42], electronic circuits [37], photonic crystals [83], polymer optical fibers [68], and real world crawling robots for locomotion [64, 82] to cite a few. What these works have in common is the fact that the objective function is defined such that it gives minimal information on the performance of the evaluated design (e.g. travelled distance, radio signal strength, structure stability). However, the relative freedom in the design definition made it possible to achieve impressive results: satellite antennae from [66] actually ended up being more efficient and more compact than human designed alternatives, and were integrated in the design process of a satellite and sent to space.

From a practical viewpoint, Evolutionary Algorithms are population-based metaheuristics that provide the human engineer with a set of tools to address particular optimization problems[1]. The core principles are built upon two complementary mechanisms, inspired from Darwin's original principles: blind variations (favoring the introduction of new candidates) and survival of the fittest (favoring pressure towards the best individuals). Figure 1.1 illustrates this process with the example of offline behavior optimization of an autonomous agent. The left part of the image illustrates the evolutionary loop: an initial population of random individuals is generated randomly, each individual corresponding to a genome (e.g. a set of parameters or specifications) that defines a particular configuration of robot. Individuals are ranked according to their performance in order to select a subset of these individuals. These "parents" will then be used to generate new "children" individuals, whose genomes are created using stochastic variations, either by recombining several parent genomes and/or mutating a specific parent. This optimization process is termed iterative as it goes on until a pre-defined criterion is reached (e.g. maximum number of evaluations, desired performance, etc.). The right part of the image gives an example of a navigation task (i.e., a two-wheel robot should explore a maze). In this example, deliberately simplified, the aim is to design an automatic control architecture allowing an autonomous mobile robot to explore a maze. On the one side, the genome encodes the parameters of an artificial neural network connecting sensory inputs to motor outputs. On the other side, the performance of this genome is assessed by the behavior of the autonomous robot in the environment. An important remark is that the nature of the evaluation process is completely independent from the viewpoint of evolution in the offline setting, and only results in fitness values to be used for further ranking and selection.

However, the actual expertise of the human engineer is crucial to the success of such algorithms, both with regards to representation issues and evolutionary mechanisms. In fact, several practical questions must be answered before actually launching the evolutionary design process: how to describe a candidate solution? how to explore new candidate solutions? what is the structure of the problem? Moreover, fundamental issues should also be addressed related to the nature of the design process and the viability of the solution, especially regarding *robustness* and *scalability*.

[1] The interested reader is referred to [25] for a complete introduction to Evolutionary Computation and to [80] for an in-depth introduction to Evolutionary Robotics.

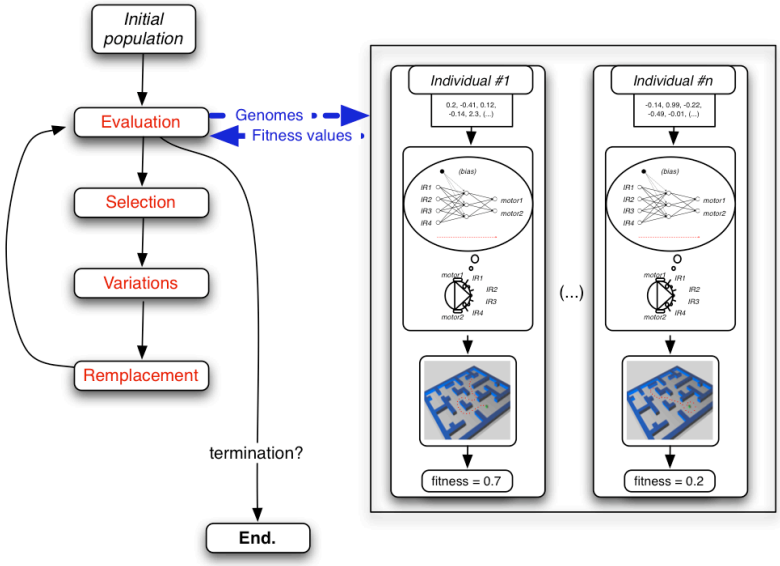

Fig. 1.1 A typical scheme of Evolutionary Algorithm for Autonomous Robot Control Architecture Optimization. **Left:** Evolutionary Process: starting from a population of randomly generated individuals, each individual is evaluated. Based on the outcome of this evaluation, individuals are selected depending on their performance. Then, a new population is generated using two kinds of variation operators: mutation (i.e. a new individual is created as a modified clone of a previous one) and recombination (i.e. a new individual is created by merging several individuals of the previous generation). The evolutionary process goes on until a stopping criterion is reached. **Right:** from the evolutionary algorithm viewpoint, the evaluation operator is simply seen as a blackbox function that maps a set of parameters or structures (i.e. the genome values) to a real value (the "fitness" value of this particular genome). In the particular case of evolutionary robotics, evaluation also encompasses a set of transformation, from the genome values to the actual phenotypic representation of a candidate solution (e.g. a robot with a specific morphology and controller), and then to the fitness value, which is the result of the behavior produced by a particular robot. It should be noted that this terminology is sometimes used in a different fashion in other application domains within EC (merging phenotype and behavior, as in most case there is no temporal aspect in the evaluation process).

1.3 When to Use ER Methods?

Despite the large amount of papers about ER, the question of the underlying hypothesis of this approach is seldom discussed.

While there exists some active research providing sounded theoretical basis of Evolutionary Algorithm [7], the practical use of such methods does not require strong mathematical know-how so as to be efficient in any context. This section attempts to provide an overview of some critical aspects of using Evolutionary Algorithm in the context of Robotics.

1.3.1 Absence of "Optimal" Method

The first and foremost remark concerns the relevance of applying Evolutionary Algorithms rather than another existing methods to solve a given problem. Evolutionary Algorithms do not guarantee convergence towards a global optima, but merely provide an efficient way to address problems that are usually left aside because of their intrinsic difficulties (ill-defined, poorly-defined, implying complex dynamics, etc.). In this scope, ER results from a compromise between applying an iterative algorithm, that may be very slow compared to analytical method, and obtaining approximated solutions rather than no solution at all. Moreover, a key advantage of Evolutionary Robotics is its anytime nature, i.e. the ability to provide one or several solutions, more or less valid, whenever the algorithm is stopped.

1.3.2 Knowledge of Fitness Function Primitives

EA principles consist in producing some diversity and then applying a selective pressure to, statistically, keep the best solutions and discard the others. The key question is that of defining what makes a solution better than the others? The behavior of solutions needs to be quantitatively described. To this end, descriptors of the behavior have to be defined and measured during an evaluation. Such descriptors are the fitness function primitives that should lead the search process towards interesting solutions.

There is no handbook to guide the design of such functions. It is often easy to define objectives able to discriminate between individuals that solve the problem – the preference going to those solving it faster or more efficiently – and likewise it is trivial to discriminate between individuals solving the task and those who don't solve it at all. The most difficult part of a fitness function design comes when individuals not solving the task at all have to be discriminated. For the algorithm to work, this discrimination should lead towards interesting solutions, but naive fitness functions often lead to local extrema, far from interesting solutions. Examples of such cases are numerous, the most famous probably being the obstacle avoidance problem. If simply defined as a count of collisions to be minimized, then the best way to minimize it is ... not to move at all ! Even if the robot is forced to move, it is simpler to find a way to turn round in a safe area, rather than taking the risk of coming close to obstacles and then of learning to use sensors.

1.3.3 Knowledge of Phenotype Primitives

The phenotype is the system to be designed by evolution. In Evolutionary Robotics, it may be the morphology of a robot, its control system or both. The goal is to find a design that best answers to the requirements on the exhibited behavior, requirements quantitatively described in the fitness function.

Evolutionary algorithms can do more than mere numerical optimization, it can also design complex structures like graphs (neural networks, for instance), set of

rules, etc. Actually, EA may both assemble and parameterize sets of primitive elements and explore open search spaces, like the space of graphs, for instance. Solving a problem other than parameter optimization with an ER approach implies to find appropriate phenotype primitives and their corresponding genotype primitives that will be assembled or modified by the genetic operators.

For the search to be efficient (and at least more efficient than a pure random search), the encoding, i.e. the way a phenotype or solution is represented in the genotype space, must be carefully chosen. [86] presents a survey of encoding related issues. If strings of symbols are used, the building block hypothesis, direct consequence of the schemata theorem of genetic algorithms, states that alphabet should be minimal and building blocks as small and independent as possible [35]. Schemas have been extended to the concept of forma [84], defined on the basis of an equivalence relation between genomes; this concept can be used to define desired properties of the crossover operator. Likewise, the genotype-phenotype mapping can be studied in order to determine how it changes the difficulty of the problem [87]. When used to generate structures, other rules have been formulated [39], see [53] for a review.

1.4 Where and How to Use EA in the Robot Design Process?

We will distinguish four different uses of EA in a robot design process:

- parameter tuning
- evolutionary aided design
- online evolutionary adaptation
- automatic synthesis

All of them do not have the same maturity. Parameter tuning consists (figure 1.2 (a) and (b)) in using EA as an optimization tool, this is their most frequent use, for which very efficient algorithms now exist, like CMA-ES [44], for instance, or NSGA-II for multi-objective problems [19]. Evolutionary aided design is a more recent trend that differs from parameter tuning in the use of the results. Whereas in parameter tuning, finding optimized parameters is the goal and generally comes at the end of the design process, in evolutionary-aided design, these optimized parameters are to be analyzed by experts to get a better understanding of the problem. Experts will then be able to propose new solutions[2] in a further step. Embodied evolution consists in using EA not only during the design step, but also during robot lifetime, in order to allow it to adapt on-line to drastically changing situations. Lastly, one promising use of EA is evolutionary synthesis. Evolutionary Synthesis is indeed the original motivation behind ER, i.e. building from scratch an autonomous agent by taking some inspiration from the actual evolution mechanisms with the goal to better exploit robot features and environment than what an engineer would do. However, due to its challenging goal, it is also the less mature use of ER as many issues remain to be studied.

[2] Whose parameters might be further tuned with an EA.

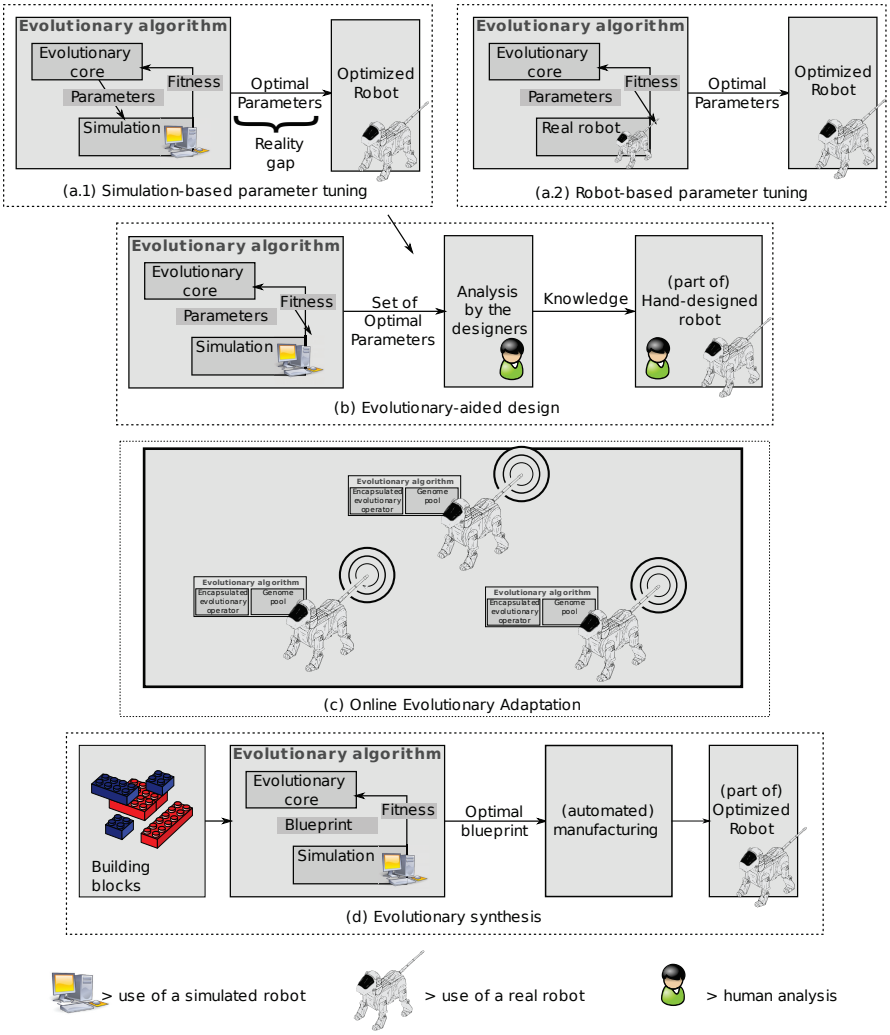

Fig. 1.2 Overview of the different uses of evolutionary algorithms in robotics. On this figure, "evolutionary core" denotes the basic evolutionary loop (see section 1.2) excluding fitness evaluation. (a.1) Parameter tuning based on a simulation then a transfer to the real robots; (a.2) Parameter tuning that uses the real robot to evaluate the fitness; (b) Evolutionary-aided design (e.g innovization); (c) Online Evolutionary Adaptation (e.g. with Embodied Evolution); (d) Evolutionary synthesis (building blocks can be neurons, physical blocks, ...).

1.4.1 Mature Techniques: Parameter Tuning

Evolutionary algorithms, and especially modern evolution strategies [43], are now mature tools for black-box optimization. As they don't impose any constraint on the objective function(s), they can be employed to tune some parameters (constants

used in control laws, width of some parts, lengths, ...) of a robot with regards to a set of defined objectives. Typical applications work from a dozen to one hundred real parameters; they involve one to four objectives [18].

One of the easiest setup is to use a robot simulator combined with an EA to find the optimal parameters of a control law [27]. For instance, Kwok and Sheng [59] optimized the parameters of PID controllers for a 6-DOF robot arm with a genetic algorithm. The fitness function was the integral of sum of squared errors of joints, evaluated with a dynamic simulation of the robot. In addition to the many papers that propose to optimize classical control laws, a substantial litterature employed EAs to find optimal parameters of neural networks or fuzzy controllers (see [27] and [28] for some overviews), especially because such controllers are difficult to tune by hand.

Since simulators are never 100% realistic, results obtained in simulation often face what is called "the reality gap": the optimal parameters obtained in simulation may not be optimal on the real robot; in many cases, the optimized controller may even rely on so badly simulated behaviors that it does not work at all on the real robot. The potential solutions to bridge this reality gap will be described in section 1.5.1.

1.4.2 Current Trend: Evolutionary Aided Design

A growing trend in evolutionary robotics is to use evolutionary algorithms for analysis and exploration tool instead of optimization. Hence, the main goal is not to find an optimal set of parameters but to answer questions such as:

- is it possible to solve a given problem using the system parameterized for another problem?
- what efficiency is to be expected if a given choice is made?
- given a set of different objectives, how antagonistic are they? Can we find a solution that is optimal with regards to all these objectives?
- does some regularities exist between optimal solutions?
- what are the critical parameters?

The typical process is divided into three steps: (1) run an evolutionary algorithm (typically with a simulated system to evaluate the fitness); (2) analyze the results to have a better understanding of the studied system; (3) implement a solution on the real robot with classic (non-evolutionary) techniques but by exploiting the new knowledge to improve the design.

Such an approach was followed by Hauert et al. [47] to evolve decentralized controllers for swarms of unmanned aerial vehicles (UAV). They first evolved neural networks to automatically discover original and efficient strategies. In a second step, they reverse-engineered the obtained controllers to hand-design controllers which capture the simplicity and efficiency of evolved controllers. The hand-design step allows to check the generality of the controllers and to use well established methods – to guarantee the stability of controllers, for instance – while taking advantage of the potential innovations brought by the evolutionary process.

Deb and Srinivasan recently demonstrated how multiobjective evolutionary algorithms (see [18]) can bring knowledge of a given system through the analysis of Pareto-optimal solutions, a process called *innovization* [20, 21]. The first step consists in selecting two antagonistic objectives (or more); an evolutionary algorithm is then employed to find the best possible approximation of the Pareto Front; last, Pareto-optimal solutions are analyzed, for instance to find relations between parameters. Typical conclusions are:

- A given parameter is constant for all the Pareto-optimal solutions;
- A given parameter can be computed as a function of another, well identified, parameter;
- A given parameter is stated to be critical;
- Performance seems limited by the range of authorized values for a specific parameter;

This analysis can then be employed to reduce the number of parameters and/or to hand-design some efficient solutions. This approach has been successfully employed to design motors [21] and controllers of a flapping-wing robot [22].

1.4.3 Current Trend: Online Evolutionary Adaptation

Evolutionary Design tools for Robotics are considered as a specific flavor in the Optimization toolbox. Broadly, Evolutionary Design is applied in an off-line manner, prior to the actual use in production of the best solution(s). Whether it is a relevant morphology and/or control architecture, a given solution may or may not feature some kind of generalization capabilities. Indeed, the outcome of the optimization process is still limited to address a specific problem or class of problems, within a limited range of variability, constrained by the experimental setting it was designed in. On the other hand, Online Learning in Machine Learning addresses problem settings where the very definition of the problem is subject to change over time, either slowly or abrutly [10]. In this scope, the goal is to provide a continuously running algorithm providing adaptation in the long run, that is the conception and production phases happen simultaneously. In the scope of ER, Online Evolutionary Adaptation is currently being explored from different perspectives, ranging from endowing robots with some kind of resilient capacity [13] with regards to environmental changes, to adapting known evolutionary algorithms to perform online evolution for single robot or multiple robots [100] or addressing environment-driven evolutionary adaptation [15] (refer to [24] for an overview).

Within Embodied Evolutionary Robotics [100], an online onboard evolutionary algorithm is implemented into one robot or distributed over a population of robots, so as to provide real time adaptation in the environment - An example is shown in figure 1.2-(c). This example illustrates Embodied Evolution in a population of robots: each robot is running an evolutionary algorithm. At time t, only one genome is ”active” and used for robot control. Genomes migrate between robots. Execution of evolutionary operators (variation, selection and replacement) takes places inside the individual robots, but communication and interaction between robots is

possibly required for genome migration. In this setup, the evolutionary algorithm is distributed and is running online, i.e. there is no distinction between the design process and the actual use of solution in a real world situation.

Advantages of this approach include the ability to address a new class of problems (problems that require on-line learning), the parallelization of the adaptation (direct consequence of population-based search) and a natural way to address the reality gap (as design constraints enforce onboard algorithms). However this also comes with a price to pay: the lack of control over the experimental setup, such as the difficulty to reset the starting position of the robots inbetween evaluations, may dramatically slow down the optimization process. However, this field of research looks promising as it naturally addresses the unavailability of human intervention and control over the environment as the algorithm is supposed to be completely autonomous from the start. Indeed, a direct consequence is that most of the works in this context have been conducted on real robots [14, 71, 97, 100, 101], which is sufficiently unusual in ER to be mentionned.

The long term goal of online evolutionary adaptation in ER is to provide continuous online adaptation by combining the ability to address the task specified by the human supervisor (the goal) with a priori unknown environmental constraints – that is constraints that cannot be expressed within the fitness function because of the a priori unpredictable nature of the environment. Hence, this field is at the crossroad of traditional optimization techniques (there is an explicitly defined goal to address), open-ended evolution (the environment particularities are to be taken into account during the course of the adaptation process), and online machine learning (the motivation is to provide an efficient algorithmic solution to solve the problem at hand). Compared to other online learning techniques, evolutionary algorithms rely on the same advantages as for black-box optimization: the ability to provide robust optimization through stochastic operators in the scope of problems with limited expert's domain knowledge.

1.4.4 Long Term Research: Automatic Synthesis

As Nature demonstrates it daily, Darwinian evolution is not solely an optimization tool, it is also a powerful automatic *design* process. The marvels accomplished by evolution inspired many researches with the long term goal of automatically designing and even manufacturing complete robotics "lifeforms" with as little human intervention as possible. From the robotics point of view, such an automatic design process could lead to "morpho-functional machines" [45], i.e. robots that can fully adapt the dynamics that emerge from the interactions between their morphology and their controller in order to optimally solve a task. The challenges raised by the automatic synthesis problems range from the understanding of biological evolution (what is the role of development to evolve complex shapes? how did living organisms evolved to modular systems?) to complex engineering problems (how could a robot be automatically manufactured, including its battery and its actuators?).

In a seminal paper, Sims [93] demonstrated how the morphology and neural systems of artificial creatures can be generated automatically with an evolutionary algorithm. Individuals were described as labeled directed graphs, which were then translated to morphology and artificial "brains". Sims was able to obtain creatures that managed to walk, swim and follow a target in a 3-dimensional dynamics simulator. The Golem project [64] put Sims' work in the robotics field by employing a 3D rapid prototyping machine to build walking robots whose morphology and controller were automatically designed by an evolutionary algorithm.

Despite these stimulating results, obtained creatures are by far many order of magnitudes simpler than any real organism. Many researchers hypothesized that designs have to be encoded using a representation that incorporates the principles of modularity (localization of functions), repetition (multiple use of the same substructure) and hierarchy (recursive composition of sub-structures) [62], three features of most biologically-designed systems but also of most engineered artifacts. Such principles led to several generative evolutionary processes that evolve programs that, once executed, generate a blueprint for a robot [49] or a neural network [40, 74]. Abstractions of the development process based on chemical gradients are also investigated [16, 32] and mostly employed to evolve neural networks. However, it has been found that these principles could need to be linked to appropriate selective pressures to be fully exploited [75], hence emphasizing that the synthesis problem may not be solely an encoding problem.

1.5 Frontiers of ER and Perspectives

ER still has many open issues. Here are several of the most critical:

- how to avoid the reality gap? Or, how to limit the risks of using an imperfect simulation to evaluate the performance of a system within an opportunistic learning scheme;
- how does it scale relative to behavior complexity? This question reveals to be actually tightly linked to fitness landscapes and exploration abilities of the EA. We will consider this question under this point of view;
- genericity of evolved solutions? For CPU time considerations, evaluations are as short as possible, and correspond thus to the behavior of the robot within only a limited set of conditions;

We will briefly discuss them in this section and sketch out current work and perspectives.

1.5.1 Reality Gap

The reality gap problem is clearly the most critical one with regards to practical applications. In theory, the reality gap should not even exist as the optimization process could be achieved directly on the target robotics setup. Several works have actually achieved evolution on real robots, such as for evolving homing behavior for a

mobile robot [29], optimizing the walking gait of an AIBO robot [50], of a pneumatic hexapod robot with complex dynamics [63] or even a humanoid robot [102]. While the optimization on the real robot guarantees the relevance of the obtained solutions, this has several major drawbacks as it can be quite consuming in term of time. As a consequence, only small populations (most of the time less than 30) and few generations (often less than 500) are performed in such a context, therefore limiting the problem that can be addressed to rather simple tasks.

Given that simulation is difficult to avoid in most practical situations, a new question arises regarding how to avoid, or at least limit, the reality gap effect, or, stated differently, how to ensure that the fitness function gives similar results within the simulation and on the robot. As a perfectly accurate simulation is highly unlikely to be available, many works focus on coping with the simulation intrinsic approximations and mistakes. A representive contribution is that of Jakobi [51] with minimal simulations: only the accurately simulated parts of the environment are taken into account and random noise is added to keep the evolutionary process from being mistakenly optimistic. Another approach consists in estimating how well a particular controller transfers to the reality on the basis of a few experiments on the real robot and then use this objective to push towards solutions that transfer well [57].

Instead of learning behaviors, ER techniques may be used to directly learn a model of a real mechanical device [11, 12, 56, 88]. Learning techniques can even be used to correct model errors online [33] or even to learn a complete model of the robot in action [13], thus opening the way towards robots able to adapt to motor failures in an online evolution scheme.

1.5.2 Fitness Landscape and Exploration

While Evolutionary Robotics has long been intended to address challenging problems, most of the achievements so far concern quite simply defined robotics problems: wall avoidance, food gathering, walking distance maximization, and other simple navigation tasks [80]. One of the major pitfalls is that the difficulty of a problem often arises with the complexity of the fitness landscape: while a smooth, convex fitness landscape with no noise will be quite easy to deal with, most of the problems from the real world often comes with multimodal, noisy fitness landscapes that feature neutrality regions. The direct consequence is that search may often get stalled, would it be at the very beginning of the algorithm execution (i.e. a boostrap problem) or during the course of evolution (i.e. premature convergence), with no hint on how to escape a local optimum or on how to direct the search within a region where all neighboring candidate solutions are equally rewarded.

Exploiting expert knowledge is a good way to escape from local optima, but as it is not always available, several solutions have been considered, the most prominent ones are listed here:

- decomposing the problem into sub-problems, each of them being solved separately, either implemented manually or learned. The resulting behaviors can then

be combined through an action-selection mechanism, that may itself eventually be tuned through evolution [34, 54, 99];
- reformulating the target objective into an incremental problem, where the problem is decomposed into possibly simpler fitness functions of gradually increasing difficulties, ultimately leading to what is reffered to as incremental evolution [36];
- reformulating the target objective into a set of fitnesses optimized independantly in a multi-objective context [73]. As opposed to the previous point, a multi-objective formulation of the problem makes it possible to avoid ranking sub-fitnesses difficulties, which is often a tricky issue;
- using co-evolution to build a dynamically changing evaluation difficulty in competitive tasks [79, 96];
- changing the evaluation during evolution to focus first on simpler problems and make the robot face progressively more difficult versions of the same task[4];
- likewise exploring solutions of increasing complexity with mechanisms protecting innovation to give new solutions a chance to prove their value [94];
- searching for novelty of behavior instead of efficiency [60, 61]. This avoids getting trapped in local optima while enhancing the search ability over robot behaviors;
- in a multi-objective scheme, adding an objective that explicitely rewards the novelty or diversity of behaviors [23, 72, 76, 77];
- putting the human into the loop. For instance, this is the kind of approach that has previously been called “innovization” [20], where the search algorithm is used to provide a basis for the expert to refine the optimization process and to provide original solutions.

1.5.3 Genericity of Evolved Solutions

One major requirement of optimization in the context of ill or poorly defined problems is to provide solutions capable of generalization, or robustness. It may indeed be very difficult to grasp all the aspects of a problem during the conception phase as the combinatorial explosion makes it impossible to generate all possible test cases. A typical example is that of a walking robot where all inclinations or textures of the ground cannot be generated during optimization, but where generalization is possible over examples. In this setup, both the experimental setup and the representation formalism are of the utmost importance. For example, relying on a test case generator or adding noise during the course of evaluation is an efficient way to enforce generalization [46]. Also, some specific representations are more fitted for generalization: artificial neural network, for example, are naturally biased towards generalization. Anyway, the actual robustness of evolved solutions remains an open question that has been seldom studied.

1.6 A Roboticist Point of View

From the Robotics point of view, control parameters tuning is a critical task since the resulting controllers dictate the behaviours of the robots. Considering the parametric identification of the dynamics model of a robot, it is important to recall that the considered model is an approximation of the real system dynamics. Thus, the identified parameters are a compromise that captures some physical properties of the system as well as some of the unknown or unmodelled dynamics. Equivalently, when tuning a PID controller for the control of a joint at the position level, the goal is to find a compromise that will best reject perturbations, most of them being hard to model (friction, backlash). In these two examples, parameters tuning is, by essence, meant to be achieved on the real system in order to best capture properties which cannot be accounted for a priori. In fact, this approach is often retained in Robotics and parametric identification and PID control tuning are then widely covered subjects in Robotics textbooks [52], [91].

One may thus argue that the use of EA in such contexts is probably not appropriate nor needed. This is only partially true. In fact, these ”roboticists” methods are well suited for problems where robots do not physically interact much with their environments. When this is not the case, either the interactions are restricted to a specific context and can be modelled using simple representation of the environment or they are not restricted to specific objects or modes of contact and in that case it is hard to tune a parametric controller that will fit a wide variety of situations. The latter case can actually not be tackled with ”low-level” controllers only and higher level decision making is often required. That is where EA may contribute: either by helping to understand what are the important physical parameters to consider within the context of a complex interaction between a robot and its environment or by tuning higher level decison making controllers that cannot be tuned using physics-based approaches only. From a more general point of view, challenges in Robotics are at the interface between high level planning methods and lower level control. This is probably the space where learning based approaches [92] and EA can best contribute in current Robotics problems.

However, EA may also still bring strong contributions in the domain of robots design. As a matter of fact, the emergence of service Robotics raises the problems of energetic efficiency and physical compliance (which is one of the prerequisite to safety) at a level such that the design problem can no longer be considered from the sole structural perspective [98], [103]. In fact, when trying to design energy efficient and compliant robots, one should consider the design and control problems as a whole. New control modes have to be explored together with new types of actuators and transmissions and EA could be one of the tools used to tackle this vast exploration problem.

1.7 Discussion

Evolutionary Robotics is a young field of research that needs time to mature and to identify its place in the engineering and science ecosystems. From a general

viewpoint, there is a need of positionning ER with related research fields. On the one hand, *Machine learning* has successfully proved its strong potential impact on robotics current challenges as demonstrated in [1] and the emergence of a new research field at the frontier between both domains is illustrated by recent publications (e.g. [92]). Although ER may be included in a weak definition of Machine Learning, there is a definitly stronger emphasis on structural design and weakly formalized open-ended problems. On the other hand, *Developmental robotics*[3] [3, 69, 81] also shares many common concerns with ER. Developmental robotics takes its inspirations from Developmental Psychology rather than Evolutionary Biology, and is concerned with learning of efficient behavior. The constraint of realism with respect to the developmental psychology ideas and models is at the core of this approach: the motivation is to provide efficient solutions as well as to validate models actually observed in Nature, while ER takes loose inspiration from it.

Up to now, the vast majority of ER published papers are proofs of concept that demonstrate, usually with a small set of experiments, that a given technique has enough potential to be further investigated. For instance, Floreano et al. [30] showed in several papers that the weights of a fully-connected continuous time recurrent neural network can be evolved to make a real robot avoid obstacles. This experiment showed that evolution *can* automatically design a controller for such a simple but useful behavior. It didn't show (and didn't aim at showing) that this method was the best to implement obstacle avoidance on a real robot.

Proofs of concept are undoubtedly important to explore new ideas. When Bongard et al. introduced resilient robotics with evolutionary algorithms [13], they did the spadework for new robotic abilities. Their work opened a previously almost not explored area of research, hence emphasizing the key role of proof of concepts. Nevertheless, such proofs of concept are only the first — often the easiest — step of an original scientific work. In an applicative context, using a given method requires to be convinced that it is one of the best method available or, at least, that this method will work with a large probability of success. This requires more than a proof of concept: strength and limitations need to be well understood, the alternatives methods should be extensively compared, the success on a large set of problems should be demonstrated. The difficulties are similar to include a given method in a larger scientific work: each brick must be strong enough to support the higher-level bricks. The abundance of proof of concepts in ER (instead of more solid knowledge building) may be the main reason for which the field is not improving faster: almost no paper re-use the results from previous papers by other authors; most of the time, researchers create a new system from scratch.

This line of thought lead us to the following conclusion: to mature, *ER need less proofs of concepts and more solid results*. To our opinion, ER has one foot in robotic engineering and one foot in experimental sciences. It therefore has much to gain from importing the best practices from these two fields.

[3] Sometimes also referred to as *Epigenetic robotics.*

1.7.1 Good Robotic Engineering Practices

To be useful in engineering, ER needs to show that it can solve important problems in robotics better than other methods. The only approach available to reach this goal is to draw extensive comparisons in which ER methods are compared to state of the art methods. The comparison should be fair and thus include at least a discussion on the most important aspects of the considered methodologies:

- the main properties of the methodology and the initial goals of the designers;
- the relative efficiency of each methodology as it is classically measured by roboticists;
- the knowledge required to apply each methodology: what should be known from the problem? What has to be done to apply each methodology?
- the constraints for the methodologies to be applied;
- the running time or the required CPU power (this is critical in the case of online ER with mobile robots).

A common pitfall is to ignore the constraints that may have driven the development of a particular methodology. In this case, the comparison isn't fair.

Another good engineering practice is to avoid to re-invent new solutions to already solved problems. The universality of the Darwinian principles suggests that everything could be designed by evolutionary algorithms. As a consequence, many ER papers deal with simple problems, for instance reactive obstacle avoidance, with the ambition that the algorithm should scale up to interesting problems, once refined. While this kind of simple tests is a way to validate the basic feature of an algorithm, this is at best a waste of computational power from a practical point of view. Smart humans spent years to develop efficient, if not optimal, approaches to solve many problems. It is currently pretentious to hope that even the best evolutionary algorithm could surpass them in a few hours (or days) of computation. On the other side, some problems (see section 1.6) are open from a roboticist point of view and ER could significantly contribute to their solution. Put differently, ER researchers should start with the state of the art for the studied problem and improve it, instead of trying to reach it from scratch, at least when then intend to show the potential of ER approaches.

Our last point is a famous engineering slogan that may seem obvious: keep it simple and stupid[4]. Many current projects are so intricate that it is impossible to replicate them in a slightly different setup. Moreover, they often are a combination of weak bricks that are not well understood. In other words, making ER a mature science requires to simplify the methods to identify the essential parts and discard everything else. If two non-critical algorithms are available to perform a sub-task, the simpler one should be chosen, even if it is less efficient; it will be easier to re-implement and to understand and will not restrict the conclusions.

[4] KISS, see `http://en.wikipedia.org/wiki/KISS_principle`

1.7.2 Good Experimental Sciences Practices

Altough the theory of evolutionary computation is improving, the evolutionary algorithms employed in ER rely on complex fitness functions and, most of the time, complex genotypes and phenotypes (e.g. neural networks). ER consequently mostly depends on empirical proofs and not on theorems. This is a significant departure from computer science, in which complexity and proofs of convergence reign, and makes ER closer to experimental sciences such as biology.

The first and foremost lesson from experimental sciences is the use of statistical tests. A single run is not sufficient to conclude anything except that there exists a solution. Comparing two sets of experiments without checking the statistical significance of the comparison is also meaningless. Additionally, comparing two methods on a single benchmark prevents the authors to conclude anything about the generality of the introduced approach. Most of the time, a Student T-test is employed to compare experiments. This statistical test assumes that the results follows a Gaussian distribution. This is often false in ER and especially if several problems are used [31]. As described in [31] for evolutionary computation, non-parametric tests, such as the Wilcoxon signed-rank test, appear more adapted to ER than parametric tests. Additionally, an experimental methodology is still lacking in ER. For instance, how to guarantee that the optimal parameters were used during a comparison? Some progress have been recently accomplished to transfer the "design of experiments" [26] approach to evolutionary computation [6, 58]. Some ideas can also be borrowed from the machine learning literature [9]. This work could be a starting point for more a rigorous design of ER experiments.

The second practice in experimental sciences that must be imported into ER is the habit of reproducing experiments. Most ER experiments are *never* reproduced by independant researchers. Contrarily to theoretical work, it is difficult to rely on experiments that have never been reproduced. However, it is often difficult to reproduce ER experiments because of the intricacy of many ER systems and the large number of parameters.

ER experiments are most of the time done in simulation and therefore internet provides a simple solution to this problem: distributing the source code, which contains every details of the algorithms. Although the solution is simple, it is not that often used. It must be emphasized that the primary goal of distributing the source code of an experiment is to let people have access to every detail of the experiment: without the code some important data may lack to reproduce the experiment. It is not to distribute a well polished and documented code (sadly, no researcher has time to do it). Furthermore, in front of the huge number of parameters, it would be worth sharing the experience on one's work reproduction to better understand what is important, what is not and how robust a particular algorithm is[5].

[5] The EvoRob_Db web site (`http://www.isir.fr/evorob_db`) aims at facilitating such exchanges.

References

1. Abbeel, P., Coates, A., Quigley, M., Ng, A.: An application of reinforcement learning to aerobatic helicopter flight. In: Advances in Neural Information Processing Systems (NIPS), vol. 19. MIT Press, Cambridge (2007)
2. Amil, M., Bredeche, N., Gagné, C., Gelly, S., Schoenauer, M., Teytaud, O.: A statistical learning perspective of genetic programming. In: Proceedings of the 12th European Conference on Genetic Programming at Evostar 2009 (2009)
3. Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., Yoshida, C.: Cognitive developmental robotics: a survey. IEEE Transactions on Autonomous Mental Development 1(1), 12–34 (2009)
4. Auerbach, J., Bongard, J.: How Robot Morphology and Training Order Affect the Learning of Multiple Behaviors. In: Proceedings of the IEEE Congress on Evolutionary Computation (2009)
5. Baele, G., Bredeche, N., Haasdijk, E., Maere, S., Michiels, N., van de Peer, Y., Schmickl, T., Schwarzer, C., Thenius, R.: Open-ended on-board evolutionary robotics for robot swarms. In: IEEE Congress on Evolutionary Computation, CEC 2009 (2009)
6. Bartz-Beielstein, T., Preuss, M.: Experimental research in evolutionary computation. In: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, pp. 3001–3020. ACM, New York (2007)
7. Beyer, H.G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
8. Beyer, H.G., Schwefel, H.P.: Evolution strategies – A comprehensive introduction. Natural Computing 1, 3–52 (2002)
9. Birattari, M., Zlochin, M., Dorigo, M.: Towards a theory of practice in metaheuristics design: A machine learning perspective. RAIRO–Theoretical Informatics and Applications 40, 353–369 (2006)
10. Blum, A.: On-line algorithms in machine learning. In: Proceedings of the Workshop on On-Line Algorithms, Dagstuhl, pp. 306–325. Springer, Heidelberg (1996)
11. Bongard, J., Lipson, H.: Nonlinear system identification using coevolution of models and tests. IEEE Transactions on Evolutionary Computation 9(4), 361–384 (2005)
12. Bongard, J., Lipson, H.: Automated reverse engineering of nonlinear dynamical systems. Proceedings of the National Academy of Sciences 104(24), 9943–9948 (2007)
13. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. Science 314(5802), 1118–1121 (2006)
14. Bredeche, N., Haasdijk, E., Eiben, A.: On-line, On-board Evolution of Robot Controllers. In: Evolution Artificielle / Artificial Evolution. Strasbourg France (2009)
15. Bredeche, N., Montanier, J.-M.: Environment-driven Embodied Evolution in a Population of Autonomous Agents. In: Schaefer, R., et al. (eds.) PPSN XI. LNCS, vol. 6239, pp. 290–299. Springer, Heidelberg (2010)
16. D'Ambrosio, D.B., Stanley, K.O.: A novel generative encoding for exploiting neural network sensor and output geometry. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007 (2007)
17. Darwin, C.: On the Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life. John Murray, London (1859)
18. Deb, K.: Multi-objectives optimization using evolutionnary algorithms. Wiley, Chichester (2001)
19. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

20. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1629–1636. ACM, New York (2006)
21. Deb, K., Srinivasan, A.: INNOVIZATION: Discovery of Innovative Design Principles Through Multiobjective Evolutionary Optimization. In: Multiobjective Problem Solving from Nature: From Concepts to Applications, pp. 243–262 (2007)
22. Doncieux, S., Hamdaoui, M.: Evolutionary Algorithms to Analyse and Design a Controller for a Flapping Wings Aircraft. In: New Horizons in Evolutionary Robotics: Post-Proceedings of the 2009 EvoDeRob Workshop. Springer, Heidelberg (2010)
23. Doncieux, S., Mouret, J.B.: Behavioral diversity measures for evolutionary robotics. In: IEEE Congress on Evolutionary Computation, CEC 2010 (to appear, 2010)
24. Eiben, A., Haasdijk, E., Bredeche, N.: Embodied, on-line, on-board evolution for autonomous robotics. In: Levi, P., Kernbach, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, Cognitive Systems Monographs, vol. 7, pp. 361–382. Springer, Heidelberg (2010)
25. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series. Springer, Heidelberg (2003)
26. Fisher, R.: Design of Experiments. British Medical Journal 1(3923), 554 (1936)
27. Fleming, P.J., Purshouse, R.C.: Evolutionary algorithms in control systems engineering: a survey. Control Engineering Practice 10(11), 1223–1241 (2002)
28. Floreano, D., Mattiussi, C.: Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies. In: Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge (2008)
29. Floreano, D., Mondada, F.: Evolution of homing navigation in a real mobile robot. IEEE Transactions on Systems, Man, and Cybernetics-Part B (1996)
30. Floreano, D., Mondada, F.: Evolutionary neurocontrollers for autonomous mobile robots. Neural Networks 11, 1461–1478 (1998)
31. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 special session on real parameter optimization. Journal of Heuristics 15(6), 617–644 (2009)
32. Gauci, J.J., Stanley, K.O.: Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2007 (2007)
33. Gloye, A., Wiesel, F., Tenchio, O., Simon, M.: Reinforcing the driving quality of soccer playing robots by anticipation (verbesserung der fahreigenschaften von fu ballspielenden robotern durch antizipation). IT - Information Technology 47, 250–257 (2005)
34. Godzik, N., Schoenauer, M., Sebag, M.: Evolving symbolic controllers. In: Evo Workshops, pp. 638–650 (2003)
35. Goldberg, D.: Genetic Algorithms in Search and Optimization. Addison-Wesley, Reading (1989)
36. Gomez, F., Miikkulainen, R.: Incremental evolution of complex general behavior. Adaptive Behavior 5(3-4), 317–342 (1997)
37. Greenwood, G.W., Tyrrell, A.M.: Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems. Wiley-IEEE Press (2006)
38. Gross, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. IEEE Transactions on Robotics 22(6), 1115–1130 (2006)
39. Gruau, F.: Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm. Ph.D. thesis, Claude Bernard-Lyon I University (1994)

40. Gruau, F.: Automatic definition of modular neural networks. Adaptive Behaviour 3(2), 151–183 (1995)
41. Hamda, H., Jouve, F., Lutton, E., Schoenauer, M., Sebag, M.: Compact unstructured representations in evolutionary topological optimum design. Applied Intelligence 16, 139–155 (2002)
42. Hamda, H., Schoenauer, M.: Adaptive techniques for evolutionary topological optimum design. In: Parmee, I. (ed.) Evolutionary Design and Manufacture, pp. 123–136. Springer, Heidelberg (2000)
43. Hansen, N., Muller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
44. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9(2), 159–195 (2001)
45. Hara, F., Pfeifer, R.: Morpho-Functional Machines: The New Species: Designing Embodied Intelligence. Springer, Heidelberg (2003)
46. Hartland, C., Bredeche, N., Sebag, M.: Memory-enhanced evolutionary robotics. In: IEEE Congress on Evolutionary Computation (2009)
47. Hauert, S., Zufferey, J.C., Floreano, D.: Reverse-engineering of Artificially Evolved Controllers for Swarms of Robots. In: IEEE Congress on Evolutionary Computation (2009)
48. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor (1975)
49. Hornby, G.S.: Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1729–1736 (2005)
50. Hornby, G.S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., Fujita, M.: Evolving robust gaits with aibo. In: IEEE International Conference on Robotics and Automation, pp. 3040–3045 (2000)
51. Jakobi, N.: Evolutionary robotics and the radical envelope-of-noise hypothesis. Adaptive Behavior 6(2), 325–368 (1997)
52. Khalil, W., Dombre, E.: Modeling, Identification and Control of Robots, 3rd edn. Taylor & Francis, Inc., Abington (2002)
53. Kicinger, R., Arciszewski, T., Jong, K.: Evolutionary computation and structural design: A survey of the state-of-the-art. Computers & Structures 83(23-24), 1943–1978 (2005)
54. Kim, K.J., Cho, S.B.: Robot Action Selection for Higher Behaviors with CAM-Brain Modules. In: Proceedings of the 32nd ISR (International Symposium on Robotics), vol. 19, p. 21 (2001)
55. Kodjabachian, J., Meyer, J.A.: Evolution and development of neural networks controlling locomotion, gradient-following, and obstacle-avoidance in artificial insects. IEEE Transactions on Neural Networks 9, 796–812 (1997)
56. Koos, S., Mouret, J.B., Doncieux, S.: Automatic system identification based on coevolution of models and tests. In: IEEE Congress on Evolutionary Computation, CEC 2009 (2009)
57. Koos, S., Mouret, J.B., Doncieux, S.: Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, GECCO 2010, ACM, New York (2010)

58. Kramer, O., Gloger, B., Goebels, A.: An experimental analysis of evolution strategies and particle swarm optimisers using design of experiments. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 674–681. ACM, New York (2007)
59. Kwok, D.P., Sheng, F.: Genetic algorithm and simulated annealing for optimal robot arm PID control. In: Proceedings of the First IEEE Conference on IEEE World Congress on Computational Intelligence, pp. 707–713 (1994)
60. Lehman, J., Stanley, K.O.: Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. Artificial Life 11, 329 (2008)
61. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search of novelty alone. Evolutionary Computation (2010)
62. Lipson, H.: Principles of Modularity, Regularity, and Hierarchy for Scalable Systems. In: Genetic and Evolutionary Computation Conference (GECCO 2004) Workshop on Modularity, regularity and Hierarchy (2004)
63. Lipson, H., Bongard, J., Zykov, V., Malone, E.: Evolutionary robotics for legged machines: from simulation to physical reality. Intelligent Autonomous Systems 9, 9 (2006)
64. Lipson, H., Pollack, J.B.: Automatic design and manufacture of robotic life forms. Nature 406(406), 974–978 (2000)
65. Lohn, J., Crawford, J., Globus, A., Hornby, G.S., Kraus, W., Larchev, G., Pryor, A., Srivastava, D.: Evolvable systems for space applications. In: International Conference on Space Mission Challenges for Information Technology (2003)
66. Lohn, J., Hornby, G., Linden, D.: An evolved antenna for deployment on NASAs space technology 5 mission. In: Genetic Programming Theory and Practice II, pp. 301–315 (2004)
67. Lohn, J.D., Linden, D.S., Hornby, G.S., Kraus, W.F., Rodriguez-Arroyo, A.: Evolutionary design of an x-band antenna for nasa's space technology 5 mission. In: Proceedings of the 2003 NASA/DoD Conference on Evolvable Hardware, EH 2003, IEEE Computer Society Press, Washington (2003)
68. Manos, S., Large, M.C.J., Poladian, L.: Evolutionary design of single-mode microstructured polymer optical fibres using an artificial embryogeny representation. In: GECCO 2007: Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation, pp. 2549–2556. ACM, New York (2007)
69. Metta, G., Sandini, G., Vernon, D., Natale, L., Nori, F.: The iCub humanoid robot: an open platform for research in embodied cognition. In: Permis: Performance Metrics for Intelligent Systems Workshop. Washington DC, USA (2008)
70. Meyer, J.A., Guillot, A.: Biologically-inspired Robots. In: Handbook of Robotics. Springer, Heidelberg (2008)
71. Montanier, J.M., Bredeche, N.: Embedded evolutionary robotics: The (1+1)-restart-online adaptation algorithm. In: Proceedings of IROS Workshop Exploring New Horizons in the Evolutionary Design of Robots (2009)
72. Mouret, J.B.: Novelty-based multiobjectivization. In: Proceedings of IROS Workshop Exploring New Horizons in the Evolutionary Design of Robots (2009)
73. Mouret, J.B., Doncieux, S.: Incremental evolution of animats' behaviors as a multi-objective optimization. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS (LNAI), vol. 5040, pp. 210–219. Springer, Heidelberg (2008)
74. Mouret, J.B., Doncieux, S.: MENNAG: a modular, regular and hierarchical encoding for neural-networks based on attribute grammars. Evolutionary Intelligence 1(3), 187–207 (2008)

75. Mouret, J.B., Doncieux, S.: Evolving modular neural-networks through exaptation. In: IEEE Congress on Evolutionary Computation, CEC 2009 (2009)
76. Mouret, J.B., Doncieux, S.: Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In: IEEE Congress on Evolutionary Computation, CEC 2009 (2009)
77. Mouret, J.B., Doncieux, S.: Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In: GECCO 2009: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. ACM, New York (2009)
78. Mouret, J.B., Doncieux, S., Meyer, J.A.: Incremental evolution of target-following neuro-controllers for flapping-wing animats. In: Nolfi, S., Baldassarre, G., Calabretta, R., Hallam, J.C.T., Marocco, D., Meyer, J.-A., Miglino, O., Parisi, D. (eds.) SAB 2006. LNCS (LNAI), vol. 4095, pp. 606–618. Springer, Heidelberg (2006)
79. Nolfi, S., Floreano, D.: How co-evolution can enhance the adaptive power of artificial evolution: Implications for evolutionary robotics. In: Proceedings of the First European Workshop on Evolutionary Robotics (EvoRobot 1998), pp. 22–38 (1998)
80. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press, Cambridge (2001)
81. Oudeyer, P.Y., Kaplan, F., Hafner, V.: Intrinsic motivation systems for autonomous mental development. IEEE Transactions on Evolutionary Computation 1(11), 265–286 (2007)
82. Pollack, J.B., Lipson, H.: The golem project: Evolving hardware bodies and brains. In: EH 2000: Proceedings of the 2nd NASA/DoD workshop on Evolvable Hardware, p. 37. IEEE Computer Society, Los Alamitos (2000)
83. Preble, S., Lipson, H., Lipson, M.: Two-dimensional photonic crystals designed by evolutionary algorithms. Applied Physics Letters 86 (2005)
84. Radcliffe, N.: The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence 10(4), 339–384 (1994)
85. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
86. Ronald, S.: Robust encodings in genetic algorithms: a survey of encoding issues. In: IEEE International Conference on Evolutionary Computation, pp. 43–48 (1997)
87. Rothlauf, F.: Representations for Genetic And Evolutionary Algorithms. Springer, GmbH & Co. K, Heidelberg (2006)
88. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science 324(5923), 81–85 (2009)
89. Schwefel, H.P.: Numerical Optimization of Computer Models. John Wiley & Sons, Inc., New York (1981)
90. Shim, Y., Husbands, P.: Feathered Flyer: Integrating Morphological Computation and Sensory Reflexes into a Physically Simulated Flapping-Wing Robot for Robust Flight Manoeuvre. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 756–765. Springer, Heidelberg (2007)
91. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics: Modelling, Planning and Control. Springer, Heidelberg (2008)
92. Sigaud, O., Peters, J. (eds.): From Motor Learning to Interaction Learning in Robots. Studies in Computational Intelligence, vol. 264, pp. 1–12. Springer, Heidelberg (2010)
93. Sims, K.: Evolving virtual creatures. In: SIGGRAPH 1994: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 15–22. ACM, New York (1994)

94. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
95. Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2), 93–130 (2003)
96. Stanley, K.O., Miikkulainen, R.: Competitive Coevolution through Evolutionary Complexification. Journal of Artificial Intelligence Research 21, 63–100 (2004)
97. Usui, Y., Arita, T.: Situated and embodied evolution in collective evolutionary robotics. In: Proc. of the 8th International Symposium on Artificial Life and Robotics, pp. 212–215 (2003)
98. Vanderborght, B., Verrelest, B., Van Ham, R., Van Damme, M., Beyl, P., Lefeber, D.: Development of a compliance controller to reduce energy consumption for bipedal robots. Autonomous Robots 24(4), 419–434 (2008)
99. Wahde, M.: A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions. Proceedings of the I MECH E Part I Journal of Systems & Control Engineering 217(4), 249–258 (2003)
100. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In: 1999 Congress on Evolutionary Computation, pp. 335–342 (1999)
101. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems 39(1), 1–18 (2002)
102. Wolff, K., Sandberg, D., Wahde, M.: Evolutionary optimization of a bipedal gait in a physical robot. In: IEEE Congress on Evolutionary Computation, CEC 2008, pp. 440–445 (2008)
103. Zinn, M., Khatib, O., Roth, B., Salisbury, J.: Playing it safe [human-friendly robots]. IEEE Robotics Automation Magazine 11(2), 12–21 (2004), doi:10.1109/MRA.2004.1310938